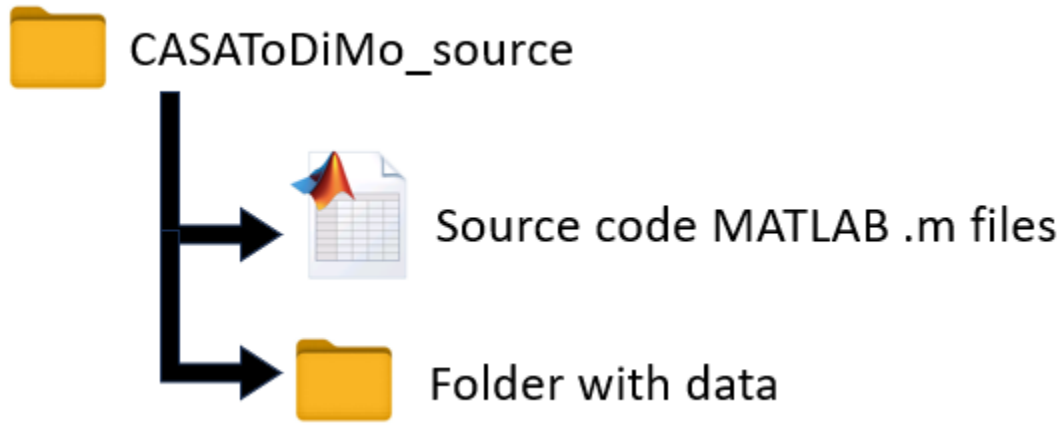
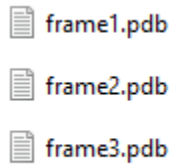
Directory setup (the folder with data just needs to be in the same working directory as the source code MATLAB files):



Example data folder contents, for a structural ensemble of three structures:



The data folder just needs to contain a set of two or more files in pdb (Protein Data Bank) format. They can be RNA, protein, poly(ADP-ribose), or already-coarse-grained. Note that pdbx or cif format may not work, but these can be easily converted into pdb format through other programs.

The hub script is "***classAverageDisordered.m***" – this is the only script that you should need to interact with. Open it in MATLAB. In the first lines of the script after the initial documentation notes there is a set of modifiable input parameters that should be considered and changed accordingly before running the program:

```
folderName: This is the name of the folder within the source code
directory that contains the PDBs that make up the conformational
ensemble to be processed.

rmsdThreshold: Threshold RMSD value (in Angstroms) for two
macromolecules to be considered "connected"; set=0 for program to
determine this value for you via optimization of a graph "Okayness"
metric. It is often good to run first with this set=0 to find a
ballpark rmsdThreshold value and tweak it from there.
```

nKmeans: Number of clusters to use for final K-means on the node/edge graph; set to 0 and the program will do a search between nKmeans=2-20 and choose nKmeans that maximizes the average silhouette

species: Either 'RNA', 'protein', 'PAR' (poly(ADP-ribose)), or 'CG' (already coarse grained from previous program)

spatialVarUpperLim: How high to set the colorbar when visualizing the classed averaged conformers (just a visual thing); 100 by default

PARtrueOrder: Only need to consider this if species='PAR', elsewise this variable does not matter. Sometimes in PAR PDBs the order of the subunits gets scrambled and you need to manually look in the pdb and enter correct subunit order.

rmsdThresholdSearchRange = (1:0.1:20): The range of RMSD values that will be tried if rmsdThreshold=0 (in Anstroms); this does not usually need to be changed unless you fail to find an optimal value during the optimization

Example of what this looks like in the script:

```
%% User-input variables
% Variables that commonly need to be changed
folderName = 'ExamplePool_MixedSequenceSingleStrandedRNA';

rmsdThreshold = 7; % Threshold RMSD value to be considered in the same class, in Angstroms; set to 0 f
nKmeans = 4; % Number of clusters to use for final K-means on the node/edge graph; set to 0 and the pr

species = 'RNA'; % Currently supported: either 'RNA', 'PAR', 'protein', or 'CG' (already coarse graine

spatialVarUpperLim = 100; % how high to set the colorbar when visualizing the classed averaged conform


PARtrueOrder = [1;2;13;16;17;18;19;20;21;22;3;4;5;6;7;8;9;10;11;12;14;15];
%PARtrueOrder = [1;2;8;9;10;11;12;13;14;15;3;4;5;6;7]; % If species='PAR', need to manually look in th
% if species is not 'PAR', this variable does not matter.

rmsdThresholdSearchRange = (1:0.1:20); % range of RMSD values that will be tried if rmsdThreshold=0 (i
```

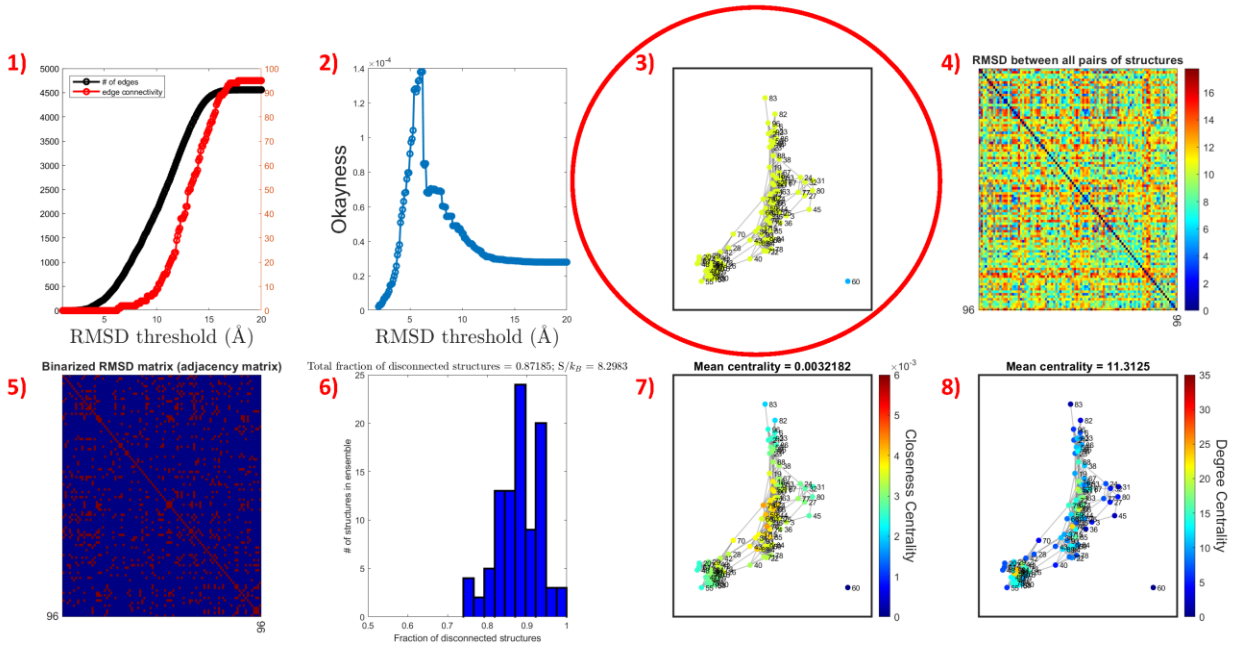
Once you are satisfied with these input variables, RUN the "classAveragingDisordered.m" script (hit the play button at the top when in the “Editor” tab or press F5).

The bulk of the computational time is spent aligning every pair of PDB structures and calculating RMSDs. The computation scales as the number of structures squared - an ensemble of 100 structures (a common magnitude for the size of a disordered macromolecular ensemble) of small-medium sized macromolecules may take ~20 minutes to perform this calculation – in this case 50 structures, for example, would thus take around 5 minutes and 500 structures around 8 hours. This is all by using a single CPU, which by default is whaT MATLAB will utilize, but if you know how to parallelize (for instance with the Parallel Computing Toolbox) then this calculation can be made much faster.

All results will be saved in a subfolder called 'outputs', which the program will create in the working directory.

Example results

Example of the first window, Figure 1, which is saved as 'SpectralAnalysis.png' in the outputs folder:



Description of plots that are output in Figure 1, going left to right starting with the top left:

1) Number of edges and edge connectivity as a function of RMSD threshold

2) Okayness as a function of RMSD threshold (see next page)

3) The spectral clustering result (K-means clustered graph)

4) Heatmap showing pairwise RMSD matrix

5) Binary pairwise RMSD matrix (used as an adjacency matrix to compute the graph)

6) Histogram of the number of disconnections present, or the number of blue squares in 5) (this serves as a relative metrix for the configurational entropy of the structural ensemble)

7) Graph colored by closeness centrality

8) Graph colored by degree centrality

NOTE: If rmsdThreshold =/= 0 and instead is set manually, plots 1) and 2) will not be output and there will be 6 plots instead of 8.

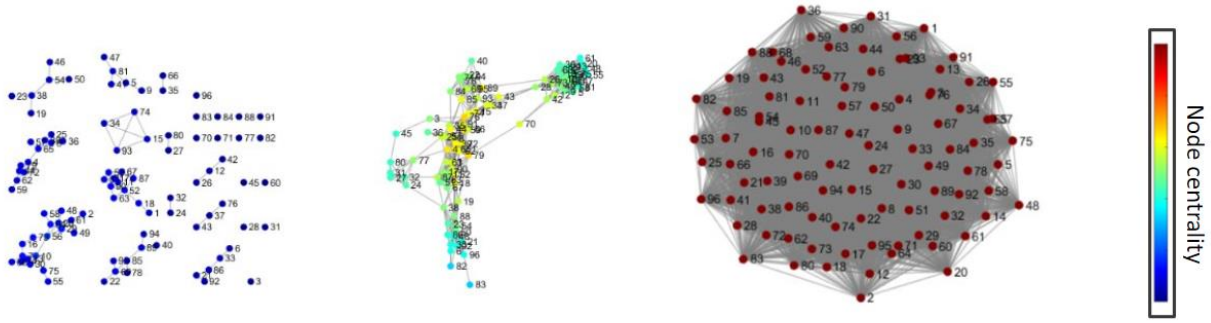
```
Spectral clustering results vary depending on the RMSD threshold and # of K-means clusters declared.
Are you satisfied with the results? Enter 1 for yes, 0 for no (redo spectral clustering).
You do not need to recompute the pairwise RMSD matrix, so subsequent runs will take much less computing time.
Note that if many stray points are present in the graph that are completely disconnected from other points, the next step might not work.
```

At this point the program will pause (in the MATLAB Command Window) and allow you to assess the results. Generally, make sure that the graph (plot 3, circled in red) looks reasonable and is not too disconnected or connected. By setting rmsdThreshold = 0 in the first run, the program can help give a ballpark estimate for what this value should be for a reasonable graph.

rmsdThreshold optimization is conducted by scanning across a series of rmsdThreshold values (by default, 1-20 Angstroms in intervals of 0.1 Angstroms) and maximizing the "Okayness" metric as a function of the rmsdThreshold 'R':

$$\arg \min_{R\in(0,20)} \left\| \frac{1}{\sqrt{N_{edges}(R)}} < C_c(R) > (1 + 0.1C_{edge}(R)) \right\|$$

Where N_{edges} is the number of edges in the graph, C_c is the closeness centrality of the graph, and C_{edgd} is the edge connectivity of the graph. The Okayness metric will tend to be low if the graph is too disconnected or too connected, and helps us find a happy medium. For instance:



RMSD threshold too low
Low Okayness

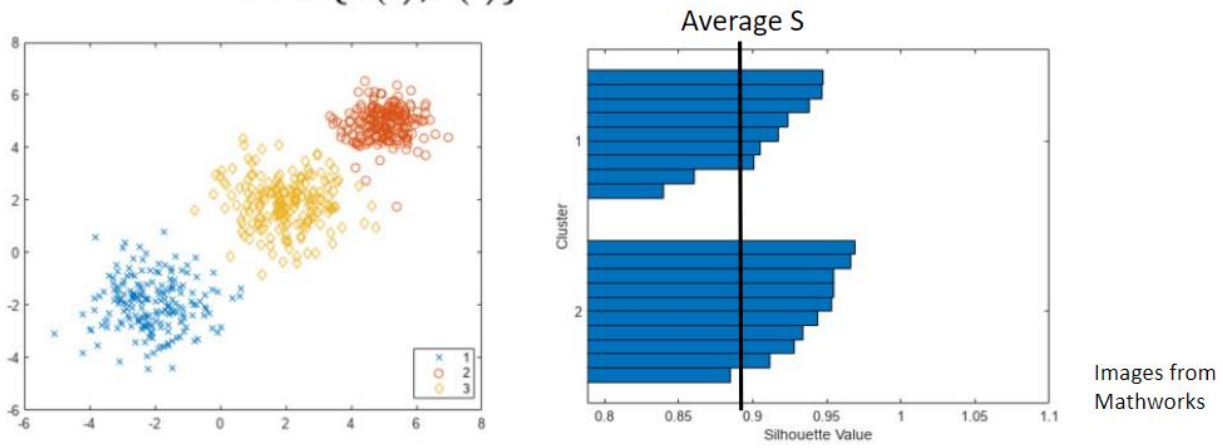
RMSD threshold reasonable
High Okayness

RMSD threshold too high
Low Okayness

The second variable that will vary the result is the number of K-means clusters. If nKmeans = 0 is set initially, the program will make an educated guess at the optimal number of clusters through the well-known method of maximizing the mean silhouette value across a number of nKmeans values, which for all points i is calculated as:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

b: inter-cluster distance
a: intra-cluster distance



Images from Mathworks

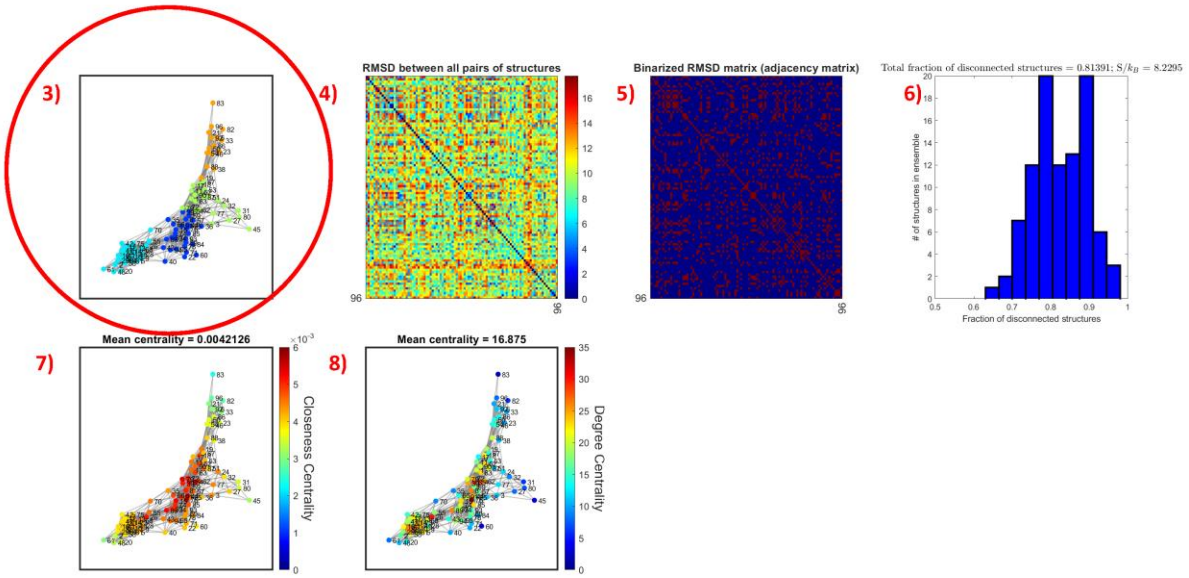
In the first pass (shown above), I let the program try and optimize rmsdThreshold and nKmeans - it output rmsdThreshold=6.2 and nKmeans=2. The graph did not look to my liking. Namely, I saw that there was a stray node in the graph that was disconnected from everything else. I also saw that 2 K-means clusters seemed too little to represent the graph. I typed '0' in the command window prompt:

```
Spectral clustering results vary depending on the RMSD threshold and # of K-means clusters declared.
Are you satisfied with the results? Enter 1 for yes, 0 for no (redo spectral clustering).
You do not need to recompute the pairwise RMSD matrix, so subsequent runs will take much less computing time.
Note that if many stray points are present in the graph that are completely disconnected from other points, the next step might not work.
0
Error using classAverageDisordered
Feel free to change the rmsdThreshold and/or nKmeans and run "classAverageDisordered.m" again.
```

This will terminate the program and allow you to make changes to the input variables before running again. To assimilate the outlier point with the rest of the graph, I increased the rmsdThreshold value to 7, making it a bit more generous which points are considered to be connected. I also tuned nKmeans to 4, making an empirical judgement that this number of clusters seems reasonable for the graph.

```
rmsdThreshold = 7; % Threshold RMSD value to be considered
nKmeans = 4; % Number of clusters to use for final K-means
```

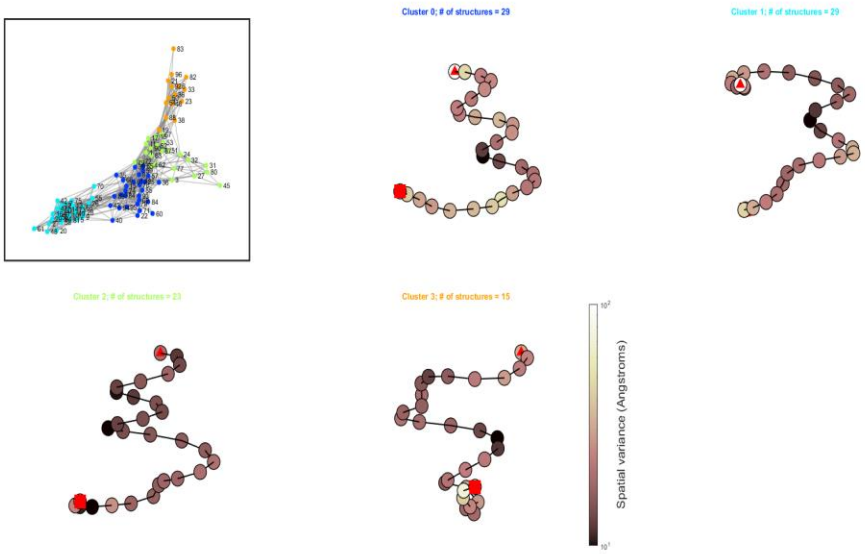
I then ran the "classAveragingDisordered.m" script again, and this was the second pass output:



The graph looks more reasonable now, with no outlying nodes and a number of K-means clusters that covers all regions reasonably.

```
Spectral clustering results vary depending on the RMSD threshold and # of K-means clusters declared.
Are you satisfied with the results? Enter 1 for yes, 0 for no (redo spectral clustering).
You do not need to recompute the pairwise RMSD matrix, so subsequent runs will take much less computing time.
Note that if many stray points are present in the graph that are completely disconnected from other points, the next step might not work.
1
Nice! Saving outputs...
Writing to log...
The program will now compute the class averaged conformers and display them when done...
------------------------------------------------------------------------------------
Generating and saving figures...
The class averaging result is saved in .../outputs.
You may now look at each individual conformer with the MATLAB figure viewer and scale/rotate to your liking.
```

So I entered '1' to advance the program to the next step, where the program accepts the spectral clustering result shown in Figure 1, saves the clustering information, and goes on to group the PDBs into separate subfolders corresponding to each identified cluster. For each of these clusters, the program then coarse grains the backbone and computes an average conformation:



This is Figure 2, which contains the resulting graph from Figure 1 along with the averaged conformation of each cluster, with titles colored according to their position in the spectrally clustered graph. This is saved as 'ClassAveragedEnsemble.fig', and you can use the MATLAB plot tools to rotate the models around, zoom in/out, etc.

The results that are saved in the ”outputs” folder in the working directory: