```
In [2]: import numpy as np
```

Accelerated agglomerative hierachical clustering with multicomparison

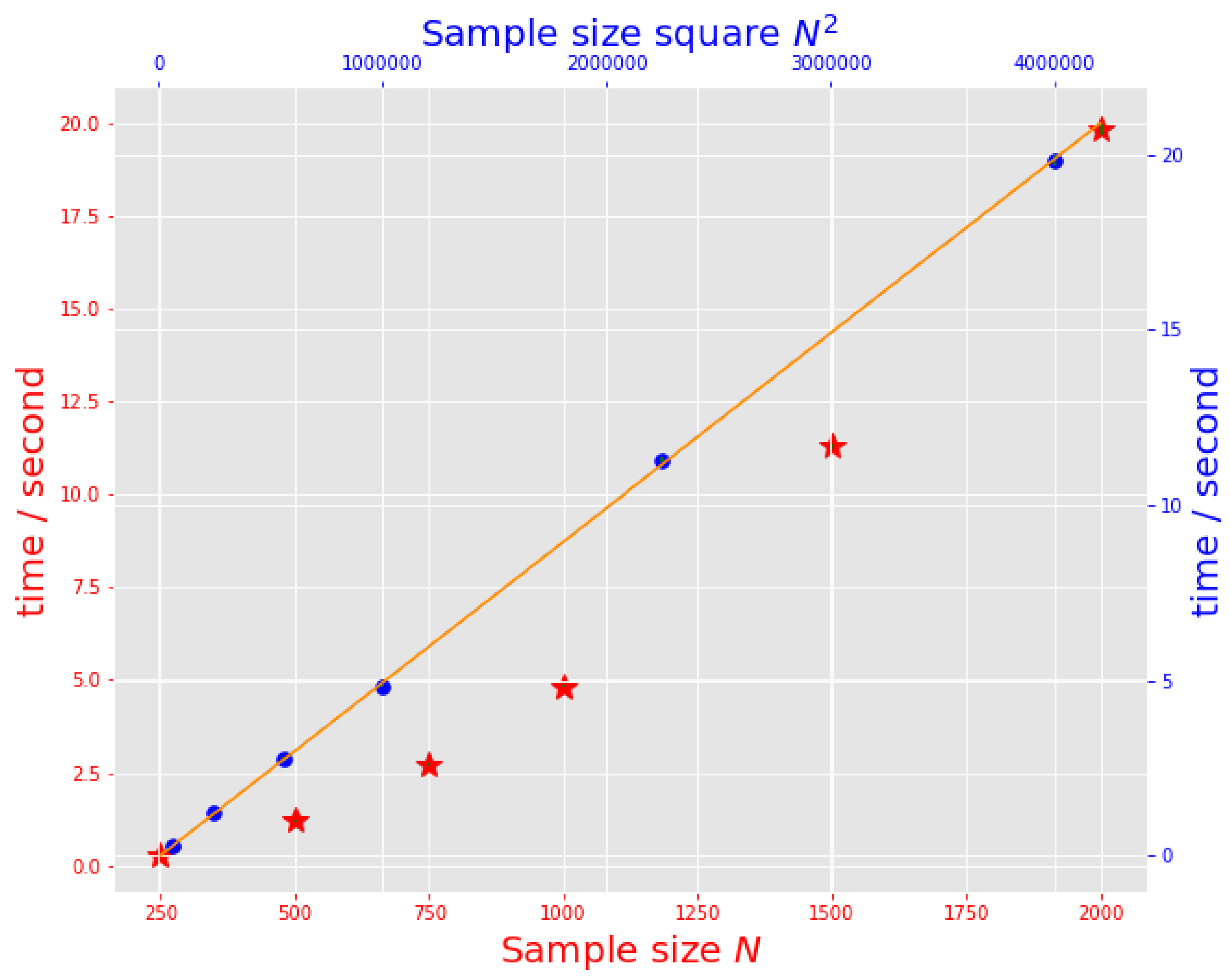
Implementation of agglomerative hierachical clustering:

- n samples from molecular dynamics simulation.
- convert each frame into contact fingerprint then forming input with size (n,m), where n is sample number and m is the length of each fingerprint.
- initialization of binary similarity matrix by comparing each two fingerprints then forming output binary similarity matrix with size (n,n) and upper diagonal 0. **e.g.** result from 1000 fingerprints binary comparison with the similarity defined by sum(a,d), where a is the number of coincident 1's and b is the number of coincident 0's.

```
In [3]: np.load('test_1000samples_simi_matrix.npy')
```

```
Out[3]: array([[  0.,   0.,   0., ...,   0.,   0.,   0.],
               [724.,   0.,   0., ...,   0.,   0.,   0.],
               [714., 702.,   0., ...,   0.,   0.,   0.],
               ...,
               [680., 684., 666., ...,   0.,   0.,   0.],
               [684., 680., 674., ..., 706.,   0.,   0.],
               [684., 680., 680., ..., 698., 734.,   0.]])
```

- Initialization binary similarity matrix speed test --- complexity $O(n^2)$, pretty fast with multi-binary comparison implementation (can be ignored compared with cost in following clustering step)



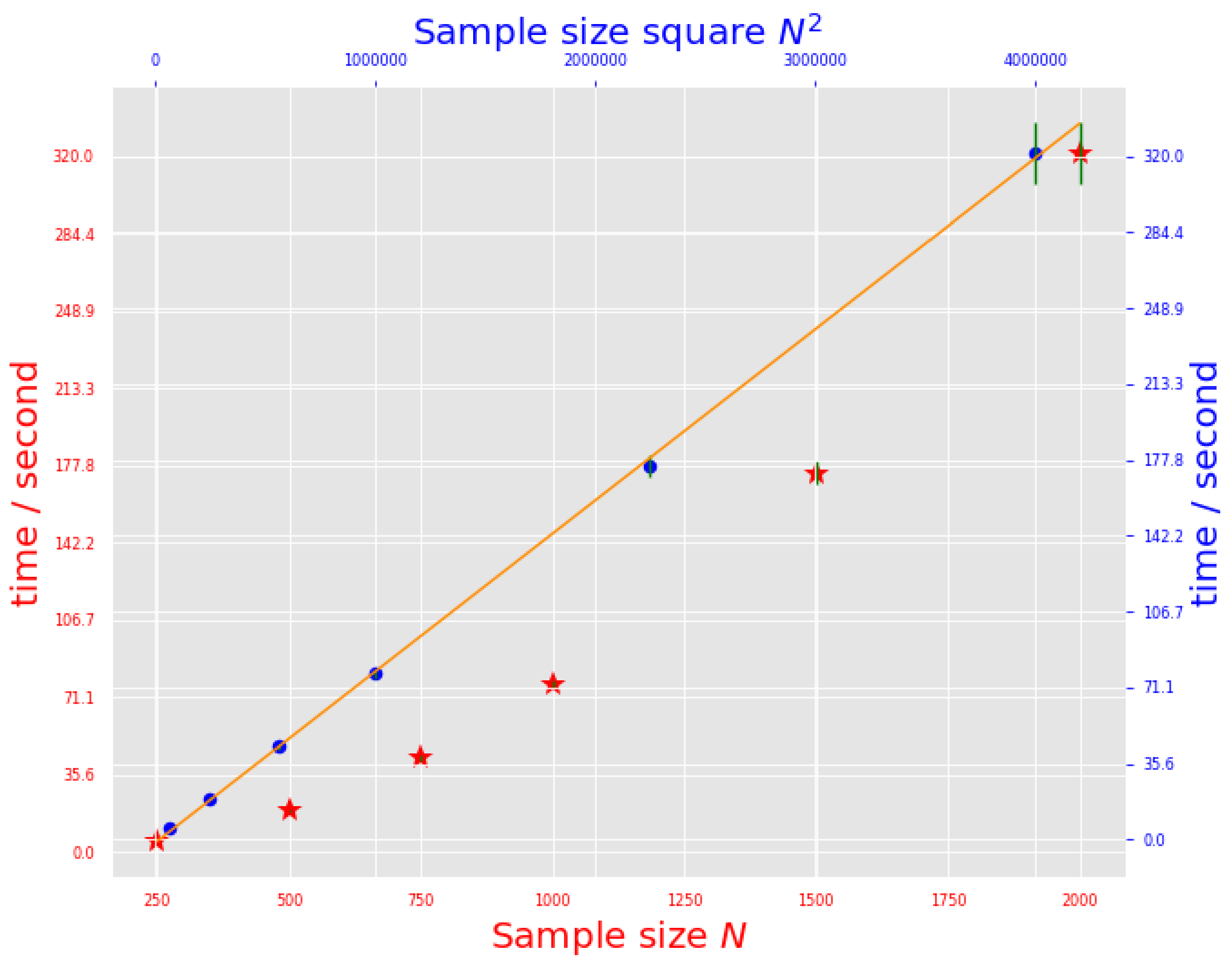
- start agglomerative hierachical clustering by updating the above like binary similarity matrix. At each clustering step, calculating the similarity of group any two clusters.

  The size of cluster will become larger along clustering process, just like the words from Ramon's note, if we have two sets, A and B, each with Na and Nb elements, respectively, this will scale as $O(NaNb) \approx O(N^2)$, which leads current algorithm scaled as $O(N^3)$ in total. However, if we link the sets by considering which of them will be more similar after we combine them, and we use the multiple comparisons described above, this will scale as $O(Na + Nb) \approx O(N)$, which leads our implementation scaled as $O(N^2)$ in total.

  I found this paper seems to be the current fastest implementation. https://www.jstatsoft.org/article/view/v053i09 (https://www.jstatsoft.org/article/view/v053i09)

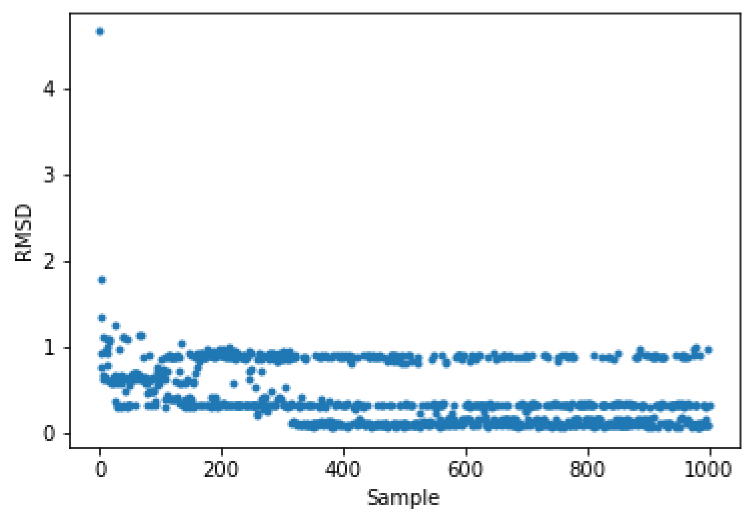
- Clustering step speed test --- complexity $O(N^2)$

  Note: the time is the average time of 10 runs with green error bar as 2*std



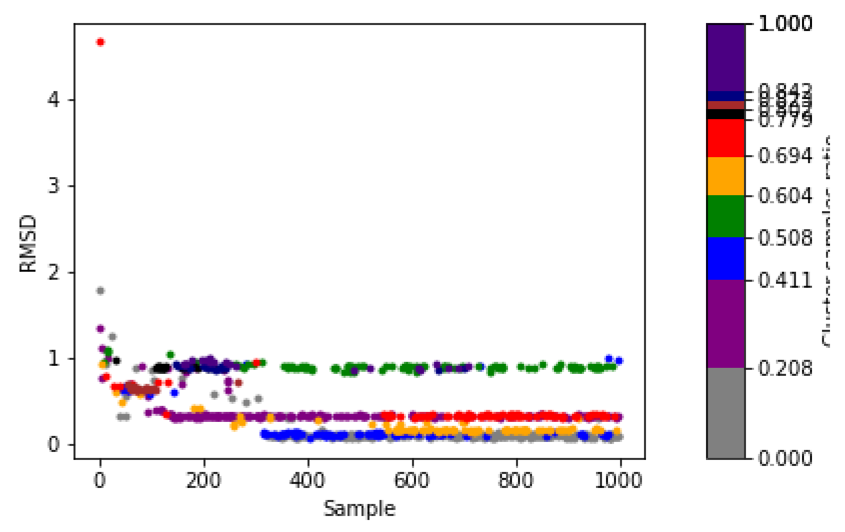
Clustering quality test

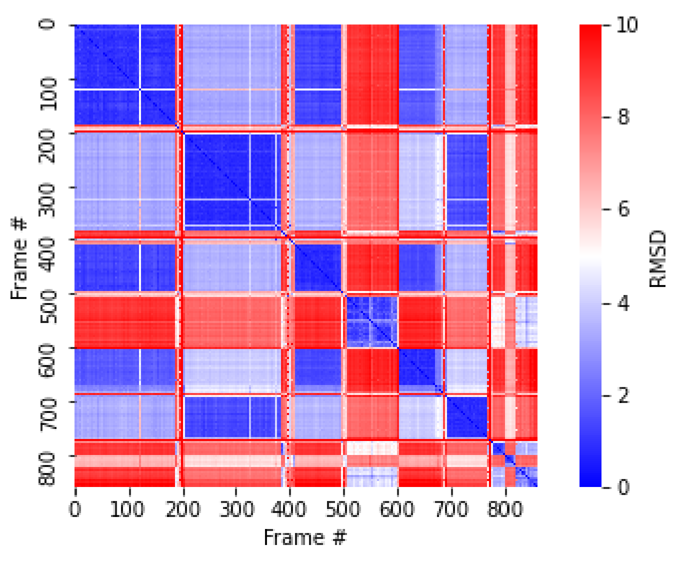
- case 1: 1000 protein samples with rmsd compared to native structure.



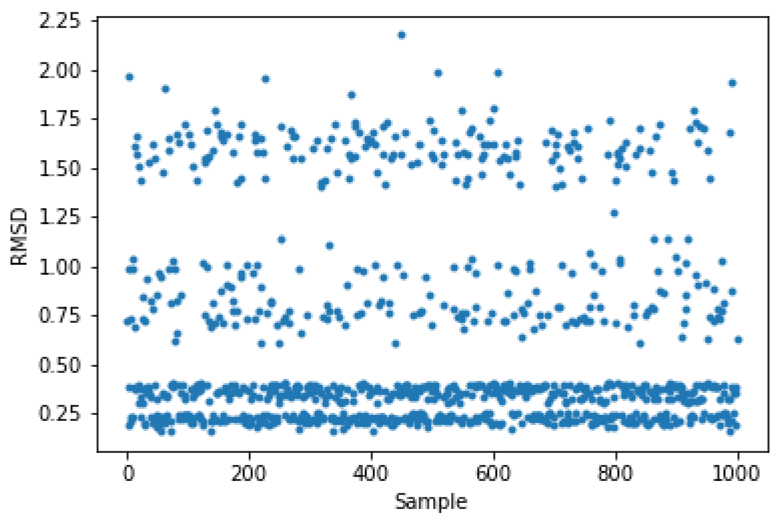
e.g.: clustering results of largest ~10 clusters from the 975th step.

1d rmsd; side bar shows number of samples / 1000 in each cluster **2d rmsd; blue square block shows ~10 clusters intra-rmsd, good**



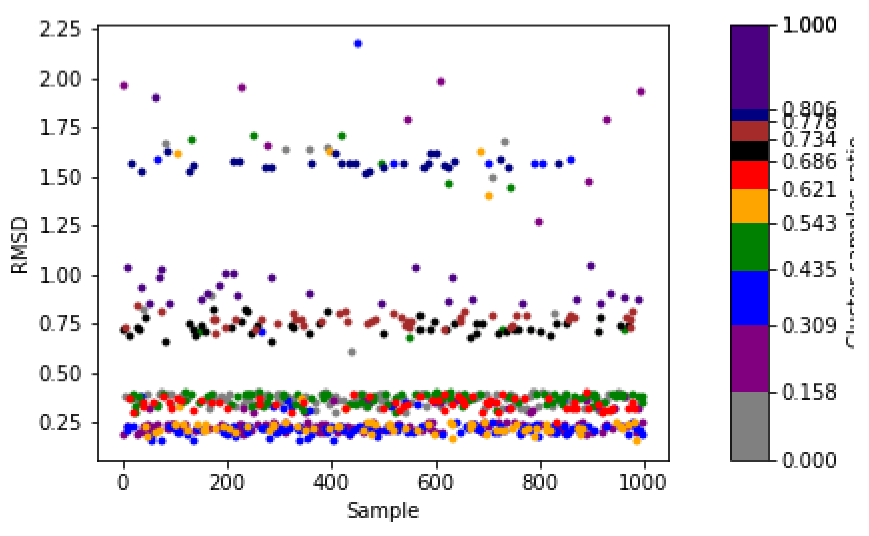


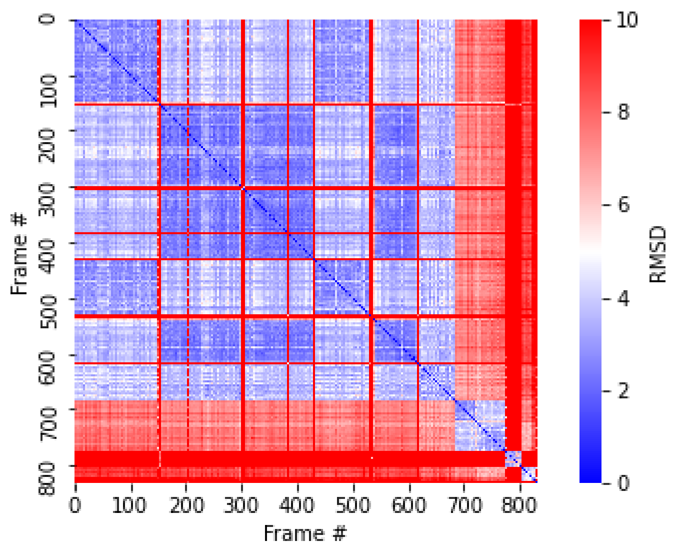
- case 2: 1000 protein-DNA binding samples with rmsd compared to native structure.



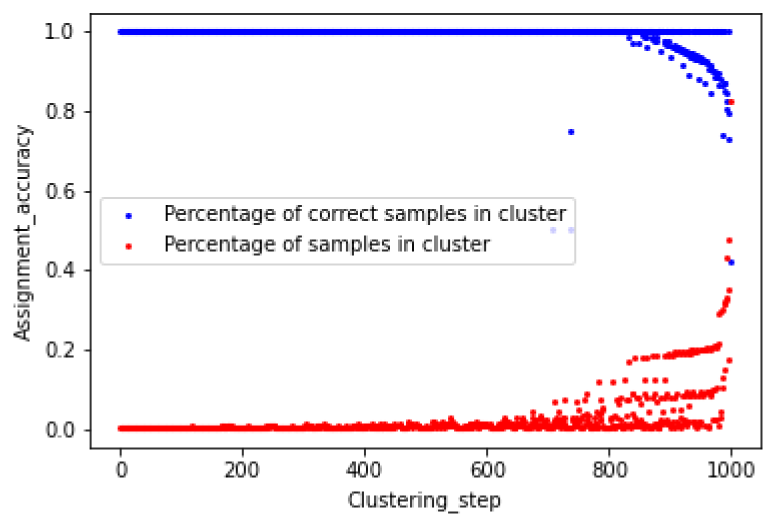
e.g.: clustering results of largest ~10 clusters from the 975th step. (Note: the fingerprint is constructed on the binding interface and the following rmsd is calculated over all "C" atoms in the system, so each blue block is not "that blue" compared with case 1, but still identifiable.)

1d rmsd; side bar shows number of samples / 1000 in each cluster **2d rmsd; blue square block shows ~10 clusters intra-rmsd, good**



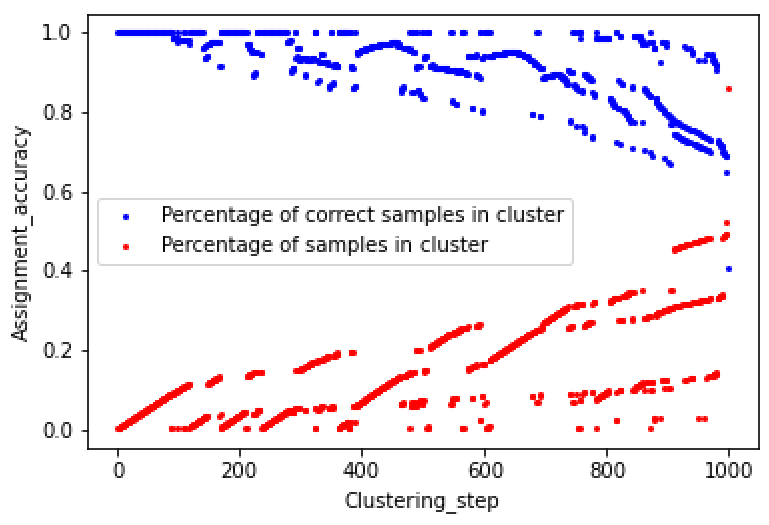
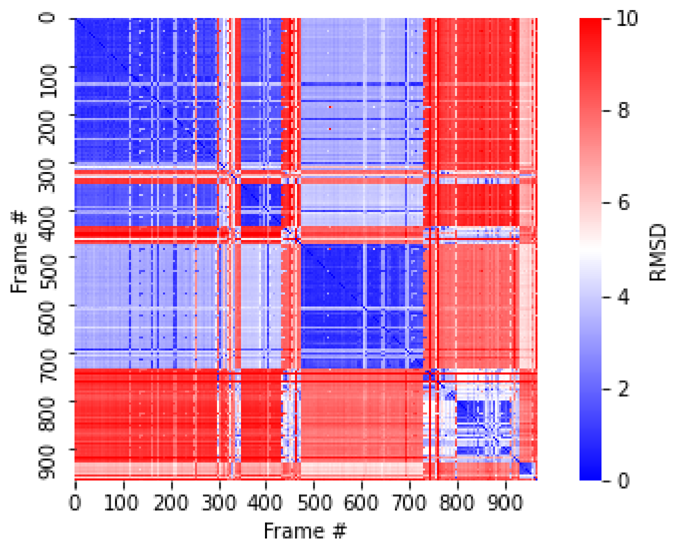


Clustering accuracy for all clustering step



- This simply implies that as more samples goes into one cluster, the cluster will become less clean. But overall, it's almost maintain ~ 90%.
- **Compared with Faith index**

  This implies that Faith index tends to group more samples with less similarity into one cluster in ealier clustering steps.

accuracy vs clustering step	**2d rmsd; blue square block shows ~5 clusters intra-rmsd, not good**
 Assignment_accuracy Clustering_step Percentage of correct samples in cluster Percentage of samples in cluster	 Frame # Frame # RMSD