A Tutorial on Clustering

The Cognitive Neuroimaging Laboratory
At
The University of Texas at Dallas

Table of Contents

In this tutorial I will explore the pipeline of solving a clustering problem. The goal of clustering is to find homogenous subgroups within observations (or subjects in our case.) Subjects have immense variability in their brain scans and we are trying to divide them to the best number of subgroups that achieve two goals simultaneously: have minimal variation within groups, and maximal variation between groups. There are many questions that we will try to tackle:

- How sparse data based on the most prominent marks may affect clustering results as opposed to using the full data? (densities)
- How clustering results can be prone to the data preprocessing decisions made? (choosing a good distance metric)
- Given N symmetric matrices representing connectivity correlations between different brain regions, what is the best way to find an optimal or suboptimal number of clusters K? (cluster diagnosis, or internal validation)
- Knowing that most clustering methods are based on random generation of points, how can we avoid the problem of having specific instances that may change dramatically (or minimally) upon a second implementation? (cluster assignments)
- Finally, having determined that K clusters would be the best number of clusters, how one can be sure that it means anything at all? i.e. how can we validate clusters against other types of data we have collected about observations such as behavioral scores? (external validation)

We will begin by discussing preprocessing techniques we used and the rational behind them, followed by a thorough discussion on diagnosis techniques that aim to find the best number of clusters and how good it is. Then I will talk about the cluster assignment based on the best number of clusters decided in the previous step (if any) and finally we see how we will validate our purely data-driven clusters with external data about subjects, i.e. behavioral data.

Before we march on to our discussion I just want to pose few questions we should be considering before and throughout clustering procedure. The main assumption of clustering is that we do have inherent structure on our data that we want to reveal. This assumption is very crucial, as clustering algorithms will always yield answers. They will never let you down, but the question is whether they make any sense to begin with. What if your data really lacks any structure? Or they have some fictitious structures that result from data collection polices and do not reflect any true properties? Clustering should be based on some theory that is backed up by previous findings in the literature. This way we can be confident that we are making a progress either by validating or invalidating previous findings. Otherwise, we will be in a position depicted by the quote saying: "*if you torture your data long enough, they will eventually confess.*"

Step 1: obtaining nodes & preprocessing

Given a cube of correlations between several brain regions of several subjects (formally known as a 3D matrix: node x node x subject), and given a vector of nodes assignments showing which systems they belong to, we can get our hands little dirty. Our goal from this very first step is to transform the data into a 2D matrix (subject x edge, where edge is the correlation value between any two nodes, see Fig 1) making it suitable for further analysis. In addition to extracting the desired edges, we will also zero out all negatives since negative correlations are susceptible to noisy signal and also are hard to interpret. Finally, we also want different densities (such as the top 10%, top 9% and so on up to the top 1%) and the rational behind going after many densities is that data are highly, highly, highly dimensional and we suspect if all correlations (including the low ones) are meaningful. So, we will just pick up the strongest correlations and zero out all the correlations below its threshold, which will lead to an even sparser edges. It's useful to have a z-transformed version of data, but this will be discussed in the next step.

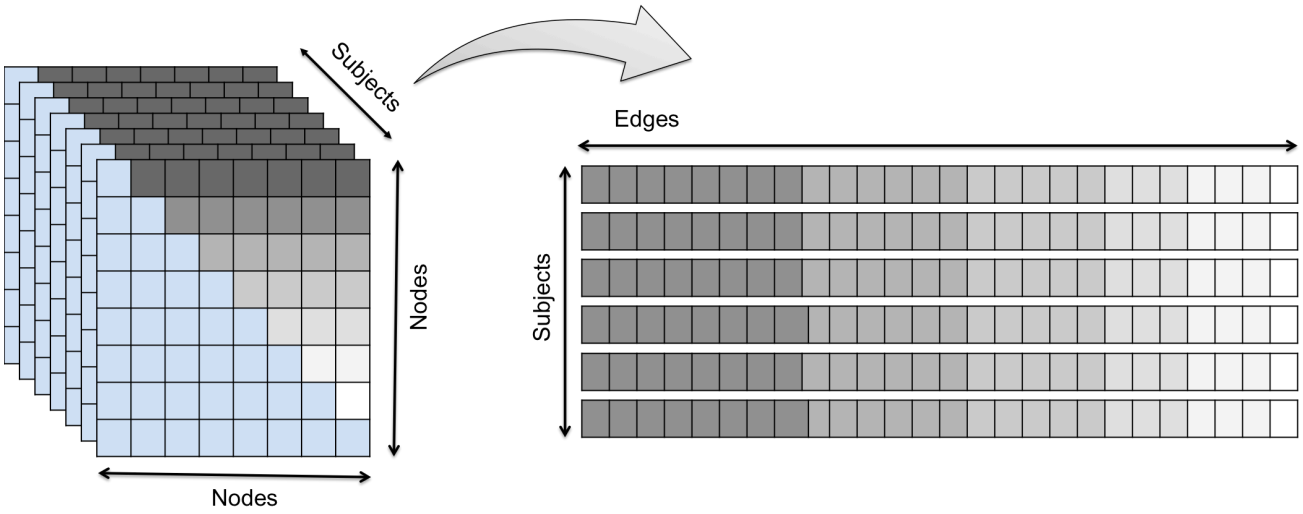


Figure 1: showing how we move from a 3D matrix to 2D matrix.

Pseudocode of step 1

- **Input**: connectivity cube (3D matrix; Node x Node x Subjects), and an optional set of desired systems to be extracted.
- **Output**: a 2D matrix (subjects x edge) for each density.
- For each density:
  - For each subject:
    - Extract the desired system nodes from the connectivity matrix.
    - Take the upper triangle of the correlation matrix and convert it to a single flat vector. (flattening)
    - Zero out all negatives.
    - Zero out all values below the corresponding percentile of current density (by subtracting the density from 100 and zeroing out all values below that number, i.e. obtaining the top 10% of edges would correspond to zeroing out all values below the 90^{th} percentile.)
    - Append vectors of all subjects into a big matrix.

Step 2: Creating the proximity matrix between subjects

Having transformed the connectivity cube into a subject x edge matrix, we can now go on and do one more step to prepare data for clustering, which is creating the proximity matrix (meaning either a distance or a similarity matrix). Why do need proximity matrix at first place? Most clustering algorithms assume a symmetric proximity matrix given and if not given they will create one behind the scenes. Clustering (talking about k-means in specific) works on a metric space. Its core idea is to find optimal locations (called centroids) that will have the minimal distance to all the surrounding observations and at the same time have maximal distance to all other centroids. This enables us defining a formal criterion for quality of clusters, as it will come in the next step.

One issue in defining the distance is scaling (which is usually done *before* creating a distance matrix). Z-scaling stands for transforming a vector of values into a mean of zero and a standard variance (by subtracting the mean from all values and dividing by the standard deviation). One unit difference in raw data is now a one standard deviation difference. We usually need scaling when we have data of different scales, which may skew the distribution and spoil any further analysis (especially in the case of k-means which is extremely sensitive to outliers.) We might need scaling if we are not very interested in raw values. Would having a very sparse dataset mean a need for scaling? And how scaling would affect the results? Answers to such questions can be found in APPENDIX A.

What about the distance matric? Well, the distance we used was either correlation (for large number of edges) or cosine similarity (for aggregated features, like the mean cross-correlations of 10 systems connectivity in addition to behavioral data). Correlation is a crude Pearson Correlation between each pair of subjects where a value of 1 means perfect correlation and a value of -1 translates to perfect anti-correlation, and cosine similarity that measures the cosine angle between two vectors.

The choice of distance here is based on previous works. What distance metrics did other groups use and it yields good and meaningful results? That may also depend on the scope and domain of research. Building on what others have already shown maybe the way to go.

The differences between cosine similarity, Pearson's correlation and Euclidean distance are addressed in APPENDIX A.

Pseudocode of step 2

- **Input**: a 2D matrix of size N x M for each density, where N is the number of observations and M is the number of edges of the desired systems.
- **Output**: a 2D subject-subject distance matrix of size N x N for each density, where N is the number of observations.
- For each density
  - Scale the subject x edge matrix, using *zscore(data)* in Matlab. This will scale each column (or edge) to its corresponding z-distribution.
    - Calculate the cross-subject correlations using the Matlab command *corrcoef(data')*, this will result in a N x N matrix of pairwise correlations between subjects, where N is the number of subjects.
    - Calculate the subject-subject cosine similarity.

Step 3: Cluster Diagnosis

The next move would be doing some cluster analysis. The downside of k-means and many other clustering algorithms we deal with is that they require *a priori* choice of number of clusters, K. You have to provide them with the number of clusters along with your data and they will do the math. On some cases we might have a good idea that we want only two or five clusters and if you find yourself in one of these lucky moments then just skip this section. However, more often than not, we are unsure on what K should we choose or our intent is actually doing exploratory analysis to answer certain questions about data. All those are convincing reasons to see how data behave under different choices of K. The first logical step is to find a range of K values that we are interested in. Finding the maximum K to be explored is discussed in the side box and APPENDIX B.

> ***How to determine the maximum K to be explored?***
> Just for sake of knowledge and maybe later use, I will briefly mention how to find the maximum K. We do hierarchal clustering of K ranging from 2 to half the number of observations. Then we build the dendrogram and sort the distances of each K to look where it no longer significantly decreases. We choose the K just before that.
> * See *APPENDIX B* for more details.

We will use different metrics (I call them diagnoses) that will guide us to find the optimal K. The idea is that we run k-means over a range of K's (say from 2 up to 10) and try to optimize for a group of metrics we have, each tells a different story about clusters. Each of them works on so many runs of k-means at a single K to have an idea about its variability, which will also help us beside the raw value of that metric. The metrics we will explore are: Cluster Quality, Cluster Stability and Subject Stability, and other similarity metrics that will be addressed later.

3.1 Cluster Quality

How clusters of high quality look like? Pretty, or rather dense around their centroids. Remember that k-means tries to find the best locations of centroids that will both (1) minimize the Euclidean distance between all the surrounding points to their centroid and (2) maximize the Euclidean distance between centroids. According to cluster quality, perfect K clusters, say two, would be in two highly populated and very separate islands. Formally speaking, cluster quality should quantify how well the defined clusters represent distinct clouds of data points. We can see that k-means assumes spherical distribution of data points (dense gathering of data points) to work really well. Mathematically speaking, higher cluster quality means smaller average distance between data points and their corresponding centroids, and bigger distance to all other centroids. We relate those two values for each cluster and take the mean quality of all clusters. (http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0009361, see Methods > Cluster Validation > Quality)

Let's go ahead and visualize what quality clusters would look like and how do we interpret quality values and variations. Running quality diagnosis over a set of artificial data points on a rage of K (from 2 to 10) will result in Fig 2A right below. Our bare eyes we tell us that a ground truth value of K would be 3 and only 3. Sure we have 3 as highest quality K.

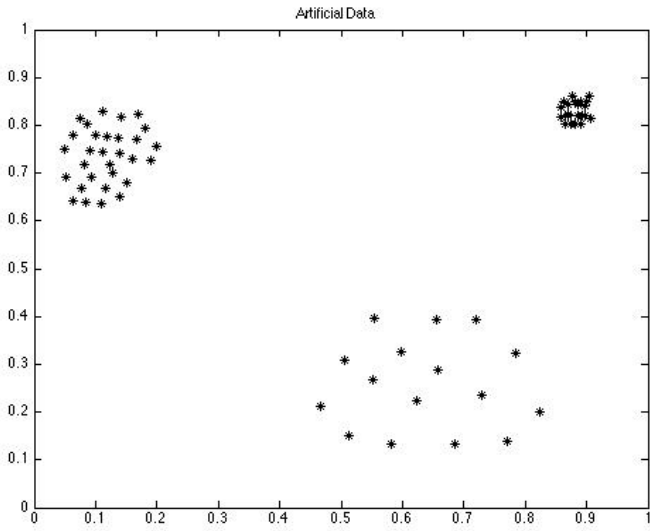


Figure 2A: Data points in 3 clusters with different qualities. The denser the cluster is, the higher its quality is.

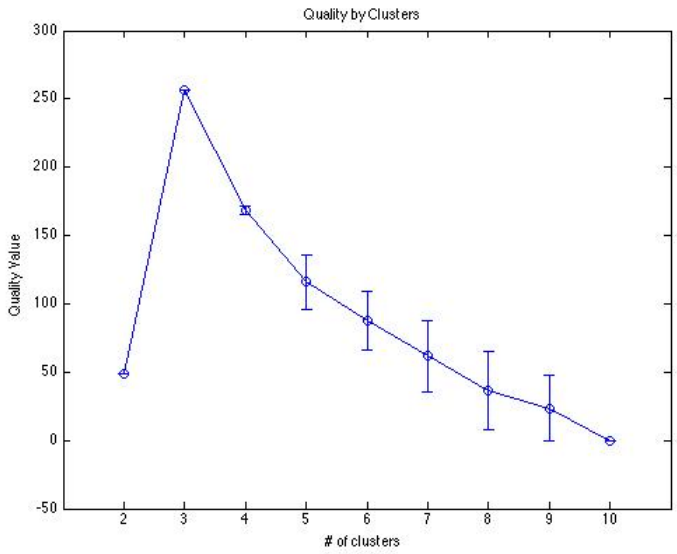


Figure 2B Quality analysis of data points of a range of K's in Fig 1 reveals what we already know, 3 clusters is the optimal K.

But wait, we already see that those 3 clusters have different qualities. The quality values at Fig 2B are the mean quality value of all clusters (over many runs). We have to be sure that we see what we expect. We have 3 clusters; one of them has a very high quality (the dense cluster on top right), and we have another one with very low quality (as points are scattered as seen bottom right) and the third should be somewhere in between. If we plot the individual qualities of each cluster we would end up with something similar to Fig 3.

Let's be sure we read those numbers correctly. So the denser cluster has a value of quality at ~650? What does that even mean?

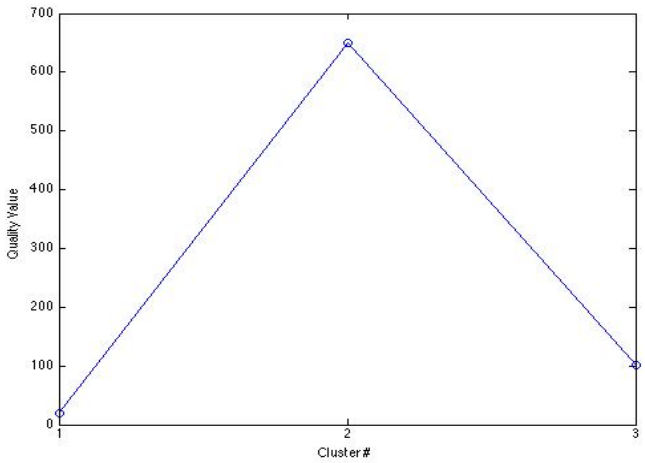


Figure 3 single quality values of the three clusters.

We have three things to elaborate:

- Mean inner cluster distance, which is the mean distance between members of a cluster and their centroid. (this would be small for dense clusters)
- Minimal outer distance, which is the minimal distance between the centroid to other centroids. (this would be big if clusters are apart)
- The quality value which is ratio of the minimal outer distance (#2) to mean inner distance (#1), which happen to be the ratio between 0.38685/0.0005953 = 649.84

We set up the magnitude of ratio to be correlated with higher quality clusters to make it more intuitive as higher value indicates better quality. Now we know that the minimal outer distance is 600X the mean inner distance. Imagine we have one of the clusters closer to this dense cluster. This would mean less quality value as the minimal outer distance would be a bit smaller. Does it make sense? It actually does. Just think when we have a wrong K that is not consistent with the structure of data. What would happen is that centroids will be closer to each other, which would make them very likely to fuse together in one cluster or maybe split apart.

In this case we want lower quality value to reflect this inconsistency. Let's make one more note about quality analysis that is depicted in Fig 4. Do you see the centroids (the red plus signs)? Here when cluster quality analysis goes really bad as the minimal distance between centroids (the two on the left) is really small and not much different from the mean inner distance of objects to their centroids. But wait; don't we see the same data points as above? Sure we do. We should be really careful with the randomization of k-means as it starts randomly at any point in the space and then try to converge. This can be easily avoided by replicating k-means many times and then it will choose the best solutions (minimizing the sum of distances between points and their centroids) over those replications, one that looks like Fig 2A.

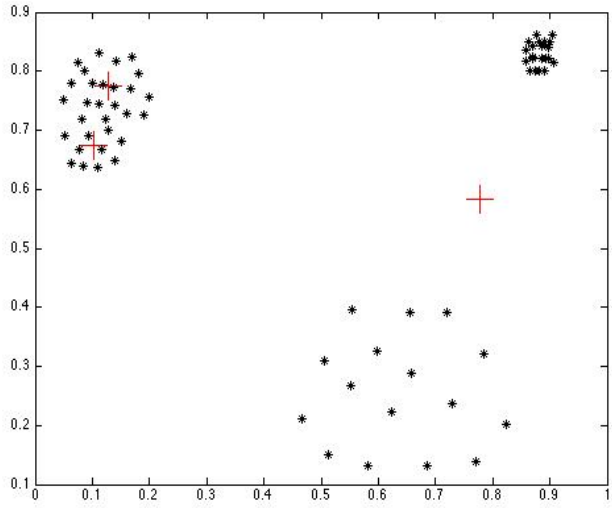


Figure 4: When k-means goes awry.

As we have highly dimensional data, data points are in general scattered in the space would be, unfortunately, very much different from the toy figures above. This means higher K's will always yield higher qualities as well (in most cases.) Don't be tricked and think that 10 clusters is a good K. It does help us, however, to see its raw value and its variation in deciding what good K's are, *simultaneously* with other metrics.

A potential and really remarkable use of this criterion (that haven't been integrated in our work yet) is finding a subset of subjects who make high quality clusters regardless of other clusters, as this criterion is cluster-wise. The quality values that correspond to any choice of K are actually the average quality of K cluster qualities. This note has to be emphasized in the context of the data we work with.

Pseudocode of step 3.1: Cluster Quality

- **Input**: Cluster centroid's locations (C), cluster assignments (IDX) and distances from each point to their centroids (SUMD) for certain K – all returned by Matlab's kmeans() function.
- **Output**: A vector of quality values of each cluster K.
  - Calculate the pairwise squared Euclidean distances between centroids.
  - For each cluster:
    - Calculate the mean inner distance from points to their centroid.
    - Divide the mean inner distance by the minimal outer distance to any centroid.
  - Then you can take the mean of all cluster qualities as cluster quality at certain K.

3.2 Cluster Stability

Having seen quality is not enough in determining a good number of clusters, K, we add another metric that tells a totally different story about data. What highly stable clusters look like? Reliable as -over many runs- k-means always assigns subjects to the same groups. We don't care about points or any Euclidean blah; we are only looking for how confident k-means is in assigning points to their clusters. According to cluster stability, a perfect choice of K would always, and always, assign every subject to the gang he belongs to. If so many subjects switch gangs across runs, this would reflect a poor choice of K that makes us even less confident about whatever clusters we get. Cluster stability is crucial and that's why we have two stability

metrics, and we have a stability value in our final cluster assignments to see which ones we don't have to trust, if there is any (as will be discussed in the next step.)

How do we measure cluster stability? Over many runs, we sample two random sets of subjects (we should select 70% to 90% of subjects at each sample.) Then we do two separate k-means for each subsample. Having done so, we see which subjects show up in both sets by intersecting the two sets. Now we have two different cluster assignments of the same group of subjects and we are interested how similar their assignments are. Avoiding the labeling problem (having different labels of same clusters), we transform each of those two assignments into an association matrix of size N x N that has 1 if subjects i and j belong to the same cluster, and 0 otherwise (See Fig 5 for a worked example). Finally, we take the correlation of those two matrices, which tells us how similar those matrices are. We repeat the process for many runs and we take the mean correlation value over all the runs. (http://psb.stanford.edu/psb-online/proceedings/psb02/benhur.pdf) Repeat the whole thing over different K's and see where the most stable K is.

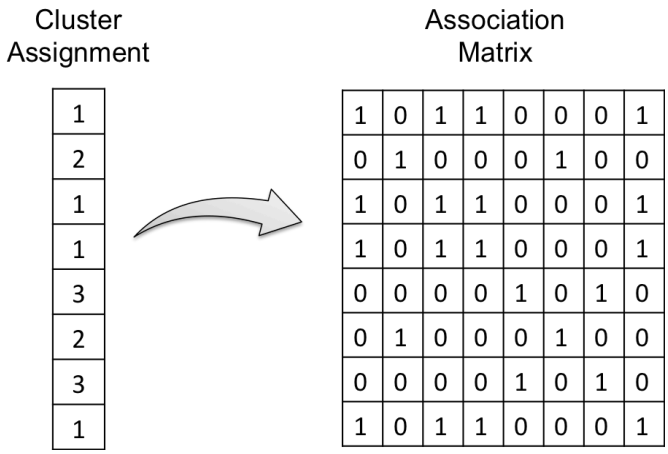


Figure 5 Transforming cluster assignment to association matrix

Let's depict the idea of cluster stability in figures so we can see what is really going on and why cluster quality is not enough for choosing the best K. In figure 6 we can see a scattered points of 3 clusters with very low qualities.

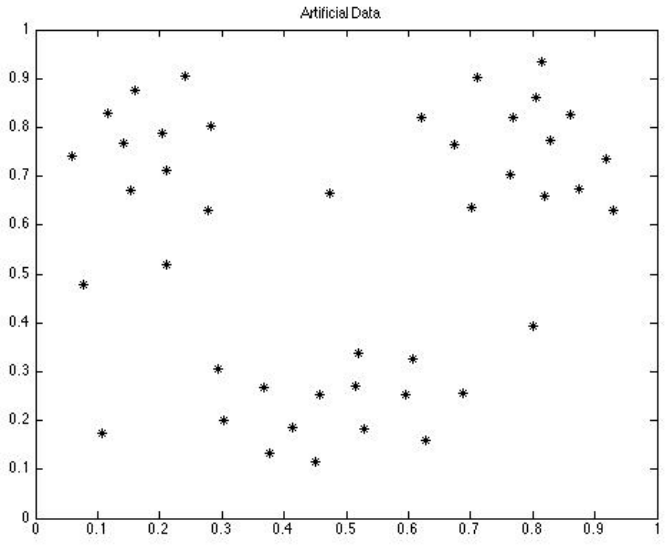


Figure 6 Data points in 3 clusters again, having significantly lower quality compared with Fig 2, but they are still separated clusters.

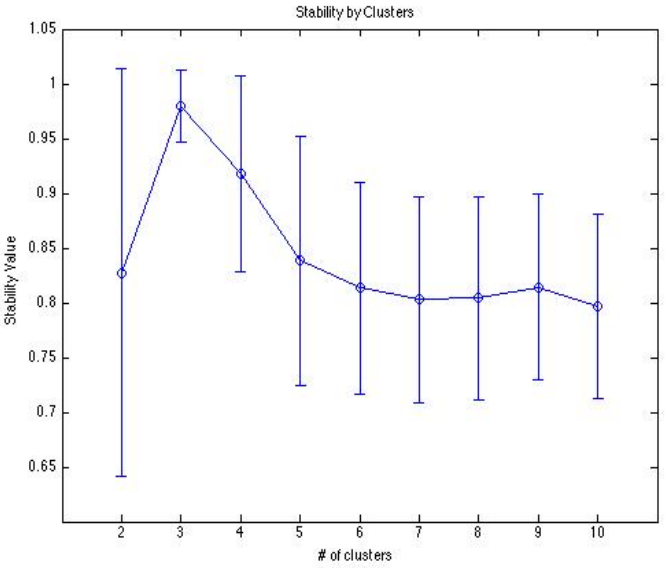


Figure 7 Stability analysis of data points in Fig 6 reveals what we already expect that 3 is an optimal number of clusters.

We see that there are 3 clusters, but they are more scattered less dense so we would expect them having lower quality values compared with the ones we say in Fig 2. But they still have a very good stability value at ~0.96 (meaning clusters are stable 96% of the time.) One more key indicator is the variation; do we see any change in variation across K's in Fig 7? Of course that drop in variation from K=2 to K=3 is indicative of a more robust cluster assignment over iterations, while we see how it increases again. In Figures 8 and 9 below we see a good case

where we may not trust stability if variations are very high and even similar to one another across clusters and even stability values barely exceed 80%.

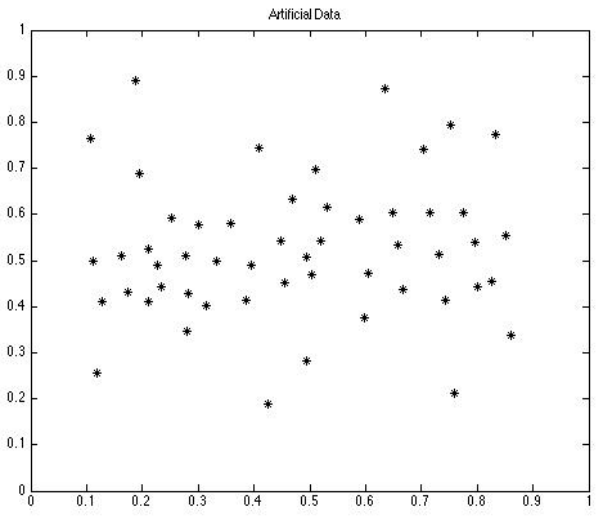


Figure 7 Data points of no specific number of clusters; a moment-of-truth for stability analysis.

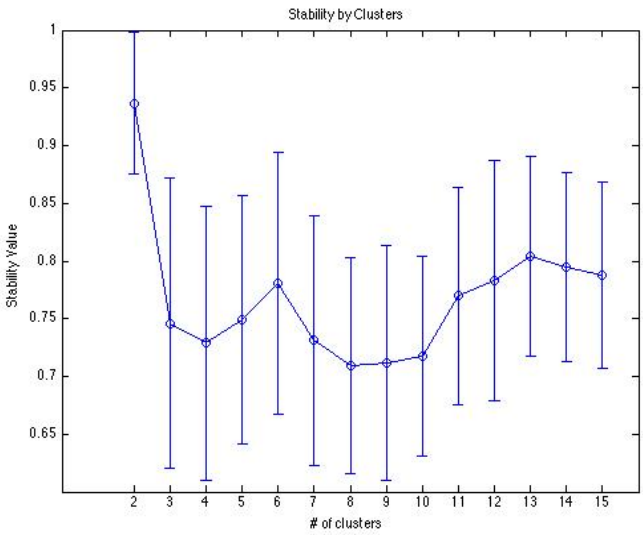


Figure 8 Stability analysis of points in Fig 7 reveals severe lack of stability at all K's -maybe except for K=2. Look at error bars and see how they persist over all K's.

As we have highly dimensional data, again, we will probably see that, in most cases, stability decreases with increasing K, and this should be a predictable behavior. This why, again, we will have to look at stability jointly with other metrics keeping an eye on variations across different K's.

A drawback with this criterion is that it's not cluster-wise. We don't have an idea on whether specific clusters, under some choice of K, are more stable than others, unlike cluster quality. However, looking at Subject Stability may alleviate this drawback.

Pseudocode of step 3.2: Cluster Stability

- **Input**: A subject-subject distance matrix of size N x N for each density, where N is the number of subjects.
- **Output**: A stability value and standard deviation of cluster stability per K over the range of K, where K is the number of clusters.
  - At each region-region density, load the corresponding subject-subject distance matrix.
  - For every number of clusters K within the given range:
    - Sample two random samples from the data (consisting of 70%-90% of data.)
    - Apply k-means clustering on each of them separately.
    - Intersect these two results to find the common subjects.
    - Given two different cluster assignments of the common subjects; convert each cluster assignment to association matrix C of size n x n (where n is the number of intersected subjects) and the values are either 1's (if both i and j belong to the same cluster) or 0's otherwise.
    - Take the correlation of these two matrices according to the formula in Benhur's paper (A link to the paper is within the discussion).
    - Repeat this many times and take the mean correlation of all correlations across runs and the standard deviation as well for every choice of K.

3.3 Subject Stability

Not only we need to know how stable different choices of K are, we also need to know how *individual* subjects are doing. None of the two metrics discussed above have a minuscule measure at the level of subjects, and here we are discussing it. It's true that in cluster stability we know whether subjects switch across runs, but we don't know who are they. Now we can finally track them down. Having a measure of subject stability is very crucial for a reason that I alluded to beforehand. In a highly dimensional space we may not expect all subject to perfectly match up in K clusters all the time, but some will (hopefully?) do. Their features are presumably less noisy than others. It will be interesting to lay down the road for further analysis on those good subjects.

How are we going about measure subject stability at each choice of K?

The idea is simple. Having run k-means at one K over and over, we now have subject x assignments matrix, where each column represents a cluster assignment. We convert each cluster assignment to a binary association matrix again of size N x N where N is the number of subjects and where the value is 1 when subjects i and j belong to the same cluster and 0 otherwise (consult Fig 5 if you are still unsure about this transformation). We now have a cube of size N x N x R where R is the number of runs. Now we will go over all the different runs of cluster assignments of subject i, which show us who is he clustered with, and find how stable his group is. We simply take the mean pairwise correlations of assignments across different runs to determine how stable his group is. Higher values mean having always put with the same group and lower correlation values reflect otherwise. We repeat this for every subject and we may go further and take the mean value of all subjects' stabilities. Did you get anything at all? It looks frustrating but Fig 9 will help you out.

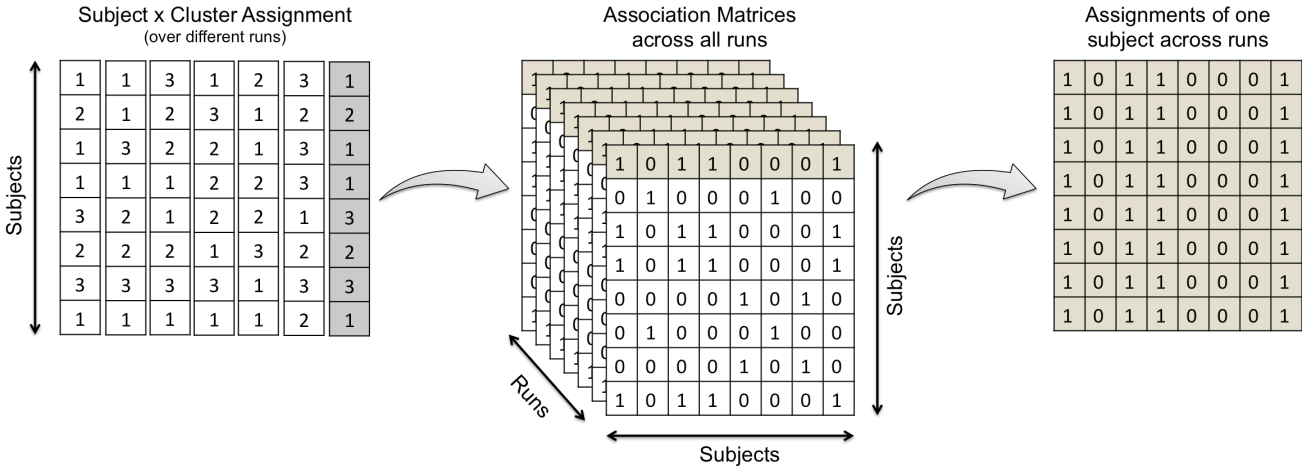


Figure 9 Subject Stability steps. (1) Transform subject x assignment matrix into cube of association matrices across different runs. (2) Then take one subject's assignments and cross correlate them to get a mean correlation that will indicate subject stability.

Pseudocode of step 3.3: Subject Stability

- **Input**: A 2D matrix of subject x edge for each density.
- **Output**: A vector of length N where N is the number of subjects, giving a percentage of each subject that represents how stable the subject is.
  - For each K:
    - Apply k-means many times and store assignments into subject x runs matrix.
    - Transform subject x assignment into a 3D matrix (subject x subject x runs) by taking association matrix of each assignment where the value is 1 if i and j belong to the same group and 0 otherwise.
    - For every subject:
    - Get the subject assignments over all runs.
    - Take the mean cross-correlations across runs.

3.4 Other Similarity/Distance Metrics

We might not be so content or totally convinced with using only those metrics. We also have other helping metrics that will back up our judgments over the best choice of K. I will briefly describe each of those metrics:

- Jaccard Coefficient (JC)

  JC is a commonly used measure of similarity of sets of binary vectors A and B. Its idea is to take the number of items in the intersection of A and B and divide them by the number of items in the union of A and B (which include the intersection plus the unique items in both A and B). So given two cluster assignments (of some K), we would divide the number of which subjects in two assignments belonged to the same cluster, by (1) the number of which subjects in the two assignments belonged to the same cluster (2) plus the number of subjects assigned to different clusters in the first assignment (3) plus the number of subjects assigned to different clusters in the second assignment. A value of 1 would represent two identical sets.
- Jaccard Distance (JD)

  JD is just 1-JC and a value of zero represents two identical sets.
- Rand Index (RI)

  RI is also commonly used in finding similarity between two sets of assignments. Given two sets A and B, RI is informally defined as the number of agreements between A and B divided by the number of disagreements between A and B. A value of 1 represents two identical sets.
- Variation of Information (VI)

  VI is a measure of distance between two cluster assignments that is based on information theory. VI measures the distance between two assignments as the sum of their entropies (minus the mutual information between them.)
- Normalized Mutual Information (NMI)

  MI is a comes from probability theory and it measures the information two clusters assignments share; how much knowing one of them reduces uncertainty about the other one. If two assignments were totally independent then their mutual information would be zero because knowing about one of them doesn't give any clue about the other. However, if they were functionally dependent on each other then their MI value would be 1 as knowing one of them also means

knowing about the other assignment. NMI is independent of labeling, so it doesn't change anything if you changed cluster labels.

The point is, no one should ever feel lost after all of those metrics. One has to also measure their specificity and sensitivity to use them better however this is beyond the scope of this tutorial.

When there is some hidden structure within our data, we only wish that those metrics working together would reveal most if of it – if not all of it – amid the noise.

Choosing the K

Having discussed many different metrics for cluster diagnosis we now can ask, how we are going to choose the best number of K? The rule of thumb is that you know it when you see it. I know that was so clever, but seriously good K's will pop out at most, if not all, metrics. If no single K can be considered good, the decision now is based on a trade-offs between all the metrics. Cluster quality might show a higher qualities of a range of K then we might go and consult cluster stability or Rand Distance that will help us decide which K is the most voted for. The decision, optimally, should not be based on only one metric; but on many metrics that also show some support – a more democratic approach. In APPENDIX C, I illustrate an example of choosing the best number of K based on simulated data points.

Step 4: Cluster Assignment

Now are done with all the diagnosis and we should, at this stage, have a good idea on what is the optimal number of clusters, or at least a range of good Ks. Sometimes you might not have any single good answer, and that's when it comes the idea of using different densities. Having higher densities help us in determining what are the strongest traces in our data that will always yield good diagnosis. Now what is next? Clustering itself.

We don't, however, run a single k-means providing the number of clusters we got from the previous step. We want to run it multiple times and transform the subject x assignments matrix into a 3D matrix of association matrices across runs (See Fig 9 for a refresher.)

Then we would sum over all runs to come up with one co-association matrix that shows how many times subjects i and j have been assigned to the same group. Finally we would run a final consensus clustering on this co-association matrix that would help us in deciding cluster assignments for each subject; it's no more than a hierarchal clustering with the maximum number of clusters set to the optimal K, and there we have our final subject assignments.

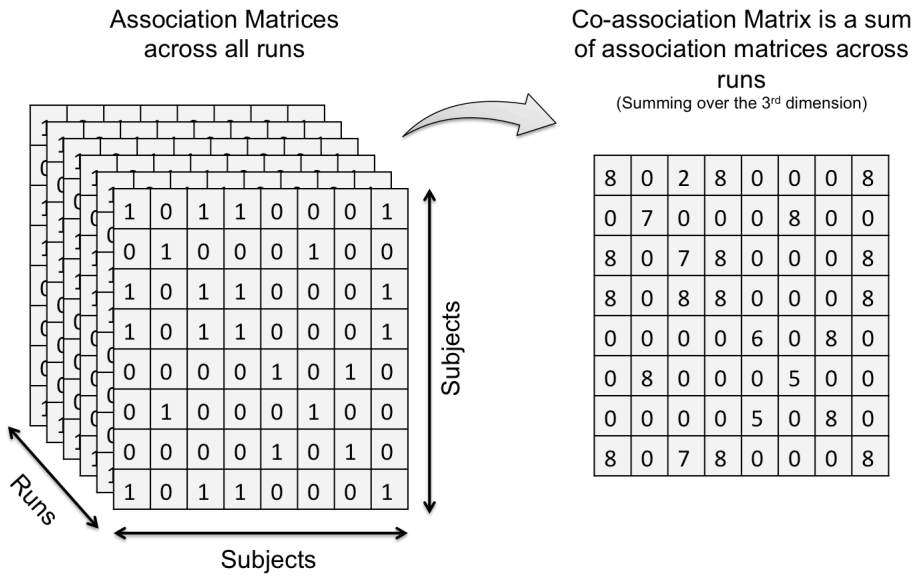


Figure 10: Co-association matrix is a sum of association matrices across many runs, the value then is an index of similarity.

Pseudocode of step 4: Cluster Assignment

- **Input**: A distance matrix of size N x N for each density, where N is the number of subjects,
  - An optimal K.
- **Output**: Cluster assignments of subjects.
- Apply k-means on this matrix many runs given the optimal K, and at each run create an association matrix C of size n x n where n is the number of subjects and where subjects i and j have a value of 1 if they have been assigned to the same cluster and 0 otherwise.
- Given a cube of size n x n x runs, sum over the third dimension to have a co-association matrix of size n x n and values now represent how many times subjects i and j have been assigned to the same cluster.
- Run hierarchal clustering on the co-association matrix with method set to 'Euclidean' and metric set to 'average' and max cluster set to the given value of k.

Step 5: Cluster Validation

One thing I haven't mentioned yet is that all what we have done in our diagnosis is called "internal validation." It means validating the choice of K based on the data themselves and assuming that we know nothing other than the data we have. There is another type of validation that is called "external validation." It means using other data sets that match the observations we used for clustering and see if our clustering solutions match up with those external data in a meaningful way. Statistically speaking each cluster makes a group and it would be the first thing to try is to look at means and standard deviations and how they compare against clusters of null models (or null clusters). Null clusters are the ones you want to reject so one has to consider how to make them; it depends on the research question. Clustering of neurological (or psychiatric) disorders would require null models that are fundamentally different from null models for aging and longevity study.

After deciding on the null clusters you may investigate variations and means of all the variables you have in clustering solution and null model and see if your solution do any better in terms of decreased variations within clusters.

One important thing to do is to visualize all sorts of data you have and in different ways, alone and in combinations with other things. Data visualization opens your eyes to things you'll never see in numbers and you also can catch if there is any interaction or interesting things going on. Examples of visualizations I created includes:

- Age distributions of subjects per each cluster, along with their average behavioral scores (that are used for external validation.)
- Standard deviations of behavioral scores per each cluster.
- Average standard deviation of all standard deviations, and compare that with other null clusters (like clusters of four cohorts that are all purely based on age.)
- Mean connectivity matrices of each cluster.
- Average correlations of the edges within systems and between systems, per each cluster.
- Cluster assignments of all subjects across densities, and also stability value of all subjects across densities.
- And many others.

External validation might not be in your side. Well, nobody has ever said clustering will always and forever be cool and effective. At that time you might want to go back and consider some alternatives like choosing different distance metrics or even choosing other clustering algorithm.

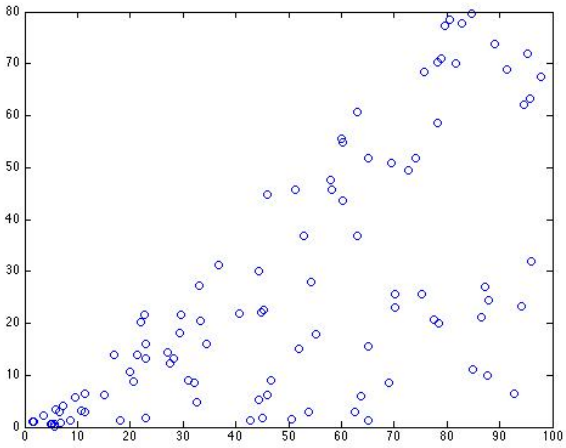
Happy Clustering,

Fahd.

APPENDIX A: When to chose cosine similarity over correlation, or even Euclidean distance?

Let's break that apart.

We will address this question by simulating correlated vectors and then see how transformations affect their similarity values. Assume we have a vector of random variables 'A', and another vector 'B' that's equal to 'A' elements multiplied by a random variable. When we plot 'A' on the x-axis and 'B' on the y-axis they should be correlated as it's shown on figure on the side.



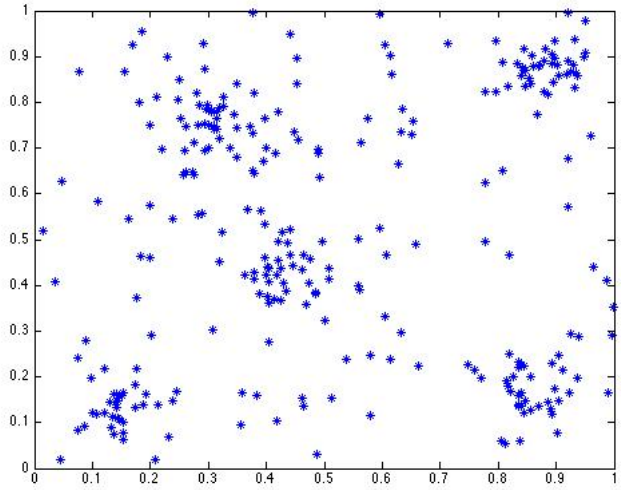
We want to see how each of correlation, cosine similarity, and Euclidean distance change as we make some transformations. We will show their proximity values on raw vectors, then we will add 100 zeros at the end of each vector and see if this would change anything, finally we will scale each variable using z-score and see how values would change.

		Correlation	Cosine similarity	Euclidean distance
1	Raw vectors	0.68073	0.86585	318.77
2	After adding zeros	0.81147	0.86585	318.77
3	After z-scaling of (1)	0.68073	0.68073	7.9508
4	After z-scaling of (2)	0.81147	0.81147	8.6623

As the table shows, correlation is sensitive to zeros. After adding a bunch of zeros its value increased, while both cosine similarity and Euclidean distance remained the same indicating their insensitivity to sparse vectors. This is relevant especially we are zeroing out all negatives at first place, and then we use different densities at which high densities will have lots of zeros. Do we want out distance be influenced by the number of zeros or not? Finally, after we scaled both vectors, we see that correlation and cosine similarity have the exact same value. This means no matter what we chose as a distance as long as we have scaled our data. We should consider scaling very carefully as scaling transforms data from one representation to another.

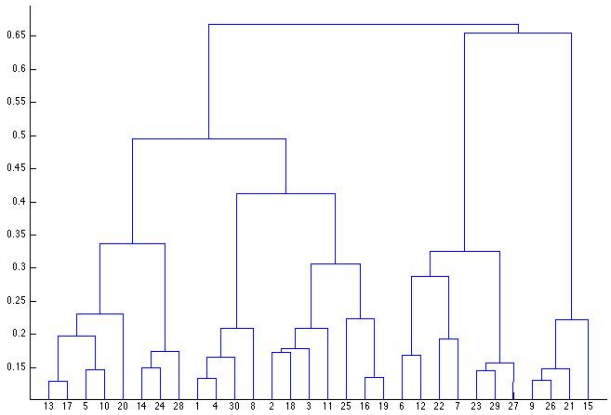
APPENDIX B: How to decide the maximum number of clusters to be explored?

I have briefly discussed how to decide on the maximum number of clusters to be explored. I said we would apply a hierarchal clustering on our dataset and see the distances between centroids as they split into more clusters. The maximum number of clusters is when this distance doesn't show a significant change.

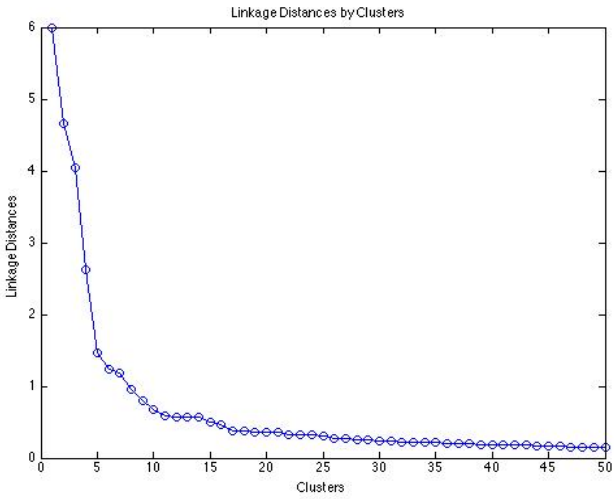


In the figure shown on the right, we see 5 clusters, but what is the range of clusters that we should be exploring?

We use the linkage algorithm. As it's shown in the dendrogram, data have nice splits. We are interested in knowing at which point does those splits don't reflect further distance (or cost) compared with the previous ones. Here we plot the distance of splits between clusters' centroids as they continue to split. At some point, splitting doesn't give further new information about the centroids. This is time where we claim that this would be a perfect maximum number of clusters as geometrical distances between any further numbers of clusters are so minimal.



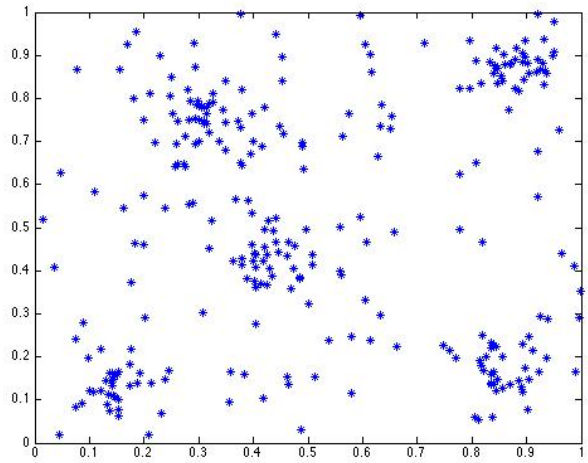
In the last figure, I would estimate this point by the vertical grey line which lies somewhere near 20.

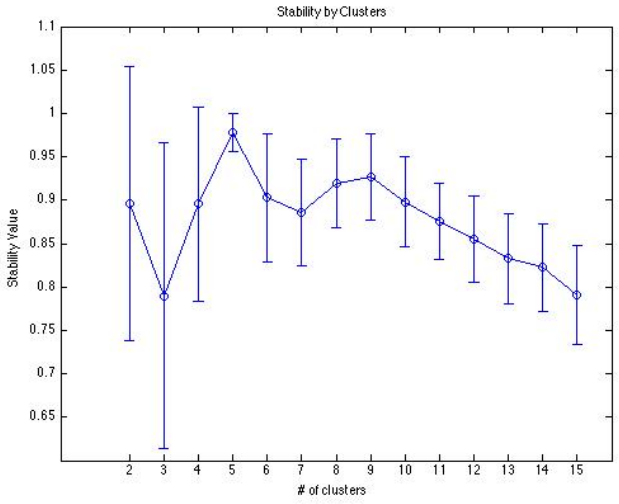


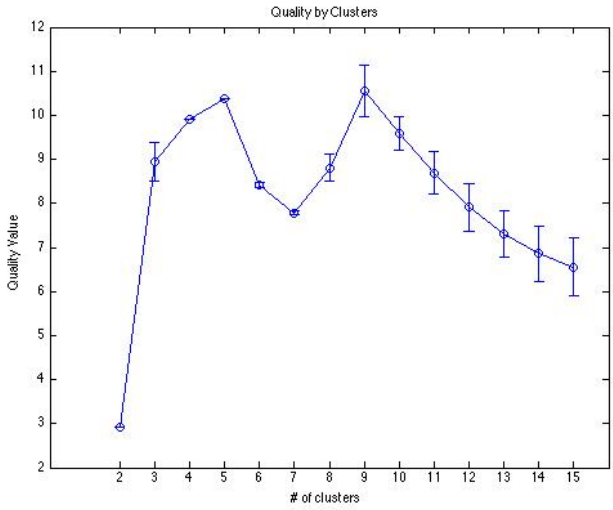
APPENDIX C:

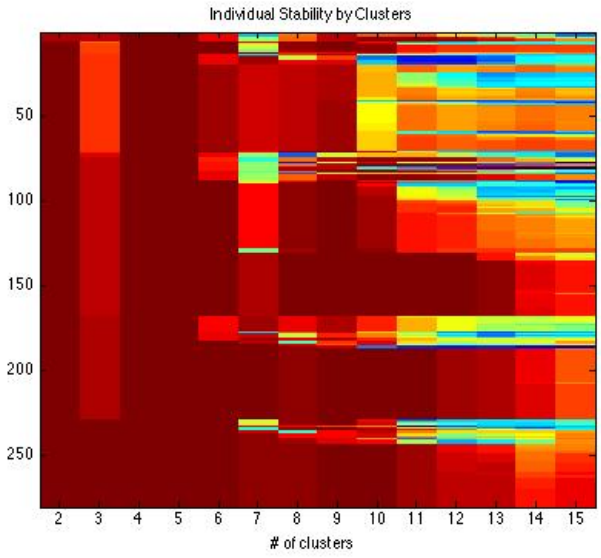
Let's see what our diagnosis tools we have discussed would reveal about the structure of the artificial data points on the right, would they lead us to the 5 clusters?

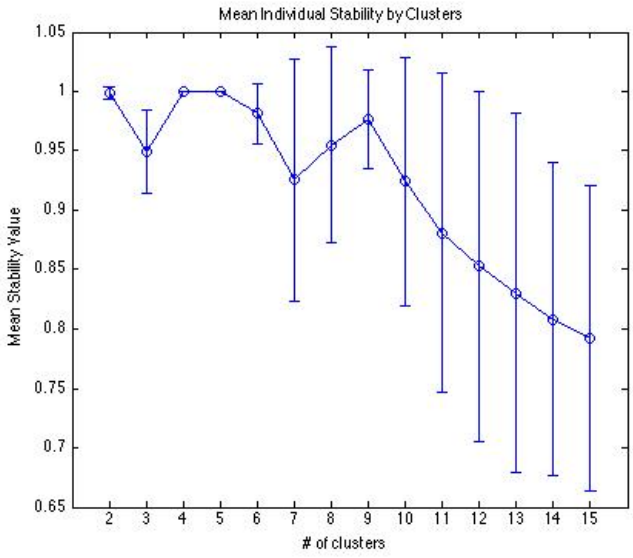
It does, and surprisingly it reveals that 9 is not a bad choice either!

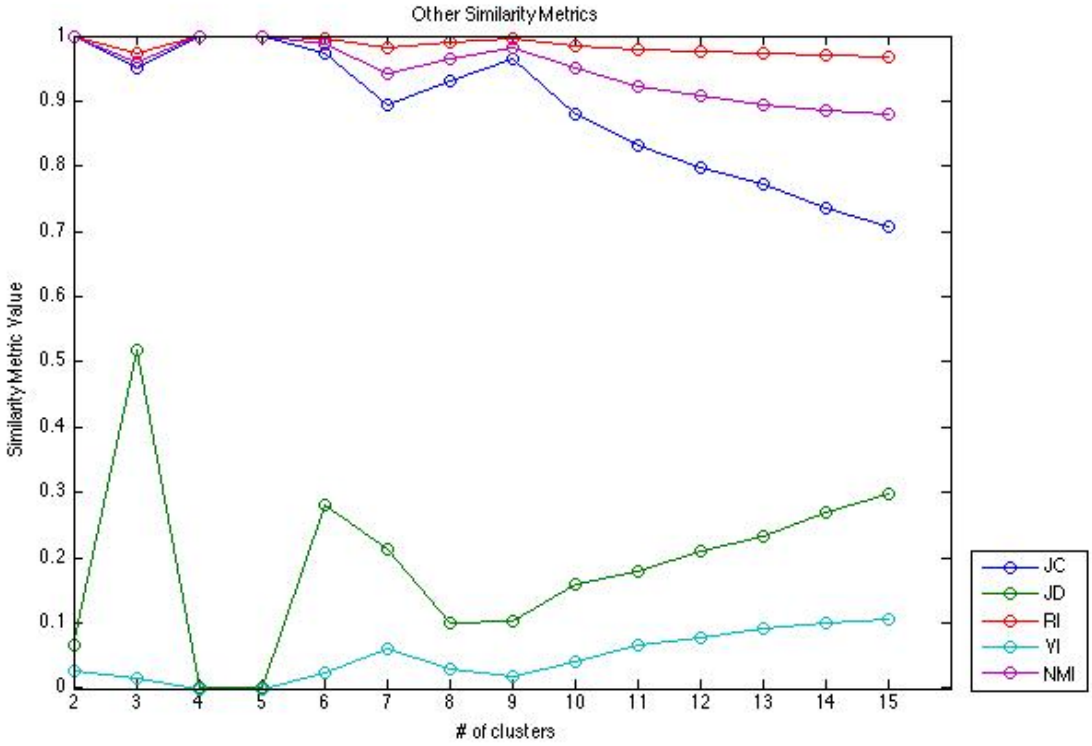











Other Similarity Metrics
Similarity Metric Value
# of clusters
JC
JD
RI
VI
NMI