Birds of a Feather: Finding Similar People and Predicting Matches in a Speed-Dating Experiment

Spencer D. Hall

August 3, 2016

Contents

List of Figures

List of Tables

1 Definition

People like to fit their world into categories; and the dating market is no exception. Internet quizzes invite people to test their Myers-Briggs similarity to their significant others and online dating sites use sophisticated algorithms to try to match people on a number of different traits. The sheer availability of potential dates via apps like Tinder^{TM}can leave people drowning in information on potential partners. A reflective person, upon looking at the myriads of factors that may go into choosing a significant other, may ask himself or herself whether (as the emphasis on knowing as much as possible about a potential match implicitly suggests) there may be unstated groupings of people based on many of these factors, that is, whether people who share the same three favorite hobbies are on average likely to have the same personality traits and similar fields. Some collections of people are relatively easy to identify based on a few key and easily-ascertained traits, like political or religious affiliation, socio-economic background, race, etc. But do more subtle groupings exist based on larger numbers of factors that are harder to consciously associate together? That is the question I am going to answer in this report. By "groupings," I mean clusters of people in the population who are consistently associated with one another on a number of different traits.

I will employ three metrics in this report. For the principal component analysis and clustering algorithm, I will select the appropriate number of principal components and clusters using the explained variance ratio and sum-of-squared errors within clusters, respectively. I will give more detail on these methods and how they are evaluated in **Methodology**, but for now, all the reader needs to know is that explained variance ratio and sum-of-squared errors are ways of making sure that principal component analysis and clustering are done in a way that maximizes the amount and quality of the information obtained in the analysis. The last metric I use, by the far the most important, will be a test of how stable the clusters obtained in this analysis are - that is, how likely are we to observe clusters like the one seen in this dataset in a new sample drawn from the same population? This metric, the Jaccardian coefficient, quantifies the probability that the cluster observed will show up in future samples, and is described in more detail in **Benchmark** below.

2 Analysis

2.1 Data Exploration

The dataset for this problem was provided by Kaggle (see references), and comes from several rounds of a speed-dating experiment. 552 people participated in all, but the rows of the dataset correspond to individual *pairings* of people, not to individuals alone, meaning that the dataset has 8,378 rows. The experimental design for data collection was ambitious; between numerous questions about the participants, information on the ratings of all participants by his or her partners, and extensive survey questions at several different points after the event, the total number of variables present in the original dataset was 195. Because I was interested only in the matches and personality traits indicated at the event itself, getting this data required extensive preprocessing. Two major problems with the dataset presented themselves. First, there were a large number of variables which represented survey questions taken after the speed dating event and which weren't relevant to my purposes of investigating only personal traits that were described at the event itself. Second, many of the variables contained huge (over 80% in some cases) percentages of missing values. The preprocessing of the data is described in **Methodology**, and left 57 variables remaining. With that many variables left over after the preprocessing, the dataset was still far too large to show a complete head of; however, the sample below shows a selection of 10 of the variables from the dataset.

	dec_o	attr_o	sinc_o	intel_o	fun_o	like_o	prob_o	met_o	field_cd	date
0	0.555556	6.777778	7.555556	8.222222	7.333333	6.944444	5.666667	1.888889	1.000000	7.0
0	0.444444	5.555556	8.666667	8.000000	5.333333	6.222222	5.444444	1.888889	8.000000	5.0
0	0.200000	5.066667	7.266667	7.600000	5.626707	5.866667	5.013883	2.000000	5.000000	7.0
0	0.166667	4.000000	6.611111	6.333333	5.777778	5.166667	4.166667	1.944444	7.662488	7.0
0	0.666667	7.444444	6.888889	6.555556	6.888889	6.444444	3.555556	2.000000	11.000000	3.0

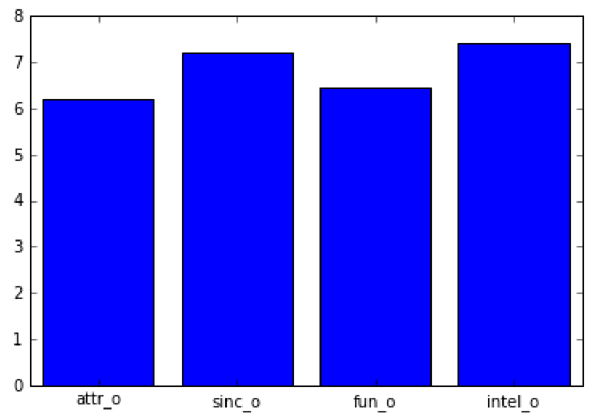

Figure 1: Partner Rating Variable Means - Bar Plot

The variables ending in "_o" are variables relating to the partner's rating of the participant. field_cd and date are the variables corresponding to the participant's field and frequency in dating.

Given that I am trying to find and validate clusters in the data, no single variable or group of variables is more important than the others *at this stage*. However, I have included summary statistics for three variables in my analysis: "match," (which corresponds to whether both the participant being surveyed and his or her partner were interested in going out), "attr_o" and "fun_o," the partner ratings of the participant on how attractive and fun she or he was. Basic summary statistics for each of these are included below.

	match	attr_o	fun_o
Mean	0.1721	6.2253	6.4511
Median	0.1429	6.2857	6.5429
Std. Dev.	0.1540	1.1762	0.9933

Table 1: Summary Statistics: match, attr_o, fun_o

By the same principle, picking variables for the exploratory visualization is fairly arbitrary, as no single variable has yet been identified as more or less important than any of the others. For the exploratory visualization below, I chose to represent the mean values of four personality variables (partner ratings of the participant on attractiveness, fun, sincerity, and intelligence) as a bar plot.

2.2 Algorithms and Techniques

There are two machine learning methods I have used in this study to identify clusters in the data: principal component analysis and k-means clustering. The details of each of these methods are discussed below.

Principal component analysis (Hastie et al.) is commonly-used technique for dimensionality reduction. Observations (rows) in a dataset exist in k-dimensional space, where k is the number of variables (columns) in the dataset. The number of observations needed to estimate a model grows exponentially with the number of dimensions in the data, which means that as the number of variables for a dataset gets larger, it becomes harder to obtain accurate results from analysis of the data. Principal component analysis helps to solve this problem by starting with the recognition that not all dimensions of the data are equally important, and in fact can and often do vary widely in terms of how much variation they show. The more variation a dimension displays, the more useful information it provides about the data. Principal component analysis seeks out the dimensions with the most variation, and then projects the original high-dimensional data onto these lower dimensions. In doing so, some of the original information is invariably lost, but often the first few principal components contain a large (70% or more) percentage of the data. Given that this lower-dimensional representation still preserves the majority of the original variation in such a way that models can be estimated more quickly and with fewer observations, the trade-off is usually a good one. By this point, the reason I am using principal component analysis in this study is probably obvious: remember that my purpose in this report is to find subtle collections of people based on large numbers of variables. The whole reason why an analysis is needed to do this is because a larger number of variables makes it much harder to tease out patterns of clustering, as *every single person* has 57 different features that have be considered. Finding patterns in high-dimensional data like this is easier if the variation is reduced to as low-dimensional a space as possible.

K-means clustering (Sklearn User Guide) is a method which is designed to do the very thing I am trying to do in this report - to find clusters of points in the data which are more similar to each other than any of them are to members of other clusters. It works by randomly inserting k "centroids" - points which are meant to be the centers of the clusters, where k is the desired number of clusters - into the data space, and assigning all of the points to one of the k clusters defined by these centroids. Assignments are made based on the Euclidean distance between a point and the centroid; that is, whichever centroid is closest to point A in the data space is the one to whose cluster point A will be assigned. The algorithm then runs iteratively, creating a new set of centroids for the k clusters which are the average of all of the points in those clusters, and then reassigning every point to a cluster based on which of the new centroids it is closest to. This process continues until the centroids no longer move noticeably. Once it is finished, the data is divided into k clusters, with the centroids of each cluster representing the average of all the points in that cluster. In the context of this dataset, the centers produced by k-means clustering would not be actual data points observed, but would represent an "average" person from that particular cluster.

The two challenges of k-means clustering which are relevant to this problem are choosing the correct number of clusters, and making sure that the clusters obtained actually represent meaningful divisions in the data rather than arbitrary noise. Let me explain each of these briefly. The k-means clustering algorithm depends on having the number of clusters you expect to see be already determined - once the algorithm begins, it will not change the number of cluster centers. If there are three real divisions in the data - groups of points that are actually quite different from each other with respect to most of their variables - and the algorithm is set to find five clusters, it will end up dividing one or more of these groups into multiple clusters and provide a bad representation of the data. If the data exists in only two or three dimensions, you can often determine a reasonable number of clusters just by looking at scatterplots of the data. But if the data is in much higher dimensions, visualizing it at all is impossible, and the choice of cluster centers must be made another way. The method I have used is something called an elbow plot. (Even when clusters can be inspected by eye, this method is often helpful in deciding ambiguous cases.) This technique finds the optimal cluster number based on within-cluster sum-of-squared errors (SSE). SSE is a measure of how "tightly" the data points in the cluster are concentrated around the centroid. If the centroid provides a poor fit to the data and does not define a cluster well, the SSE will be very large. On the other extreme, if the centroid overfits the data and all the points in its cluster are very close to it, the SSE will be quite low. The ideal number of clusters will strike a balance between having too much variance and overfitting the data. In making an elbow plot, you perform k-means clustering on the dataset with several different choices of cluster sizes, and then plot the within-cluster sum-of-squared errors by cluster numbers to see how SSE declines by number of clusters. As SSE usually declines sharply with the first few numbers of clusters and then begins to level out as the number increases, the ideal choice for k usually comes at the "elbow" of the plot, hence the name. If visualizing the plot is difficult, feel free to skip ahead to the section in **Methodology** where I show how the ideal number of clusters was chosen using one of these plots.

The second challenge with k-means clustering is making sure that the clusters obtained actually represent real divisions in the population from which the data was obtained, and are not simply artifacts of noise in the data. In case "real divisions" seems somewhat obscure, let me give an example. If we had a dataset on political views in the United States which consisted of surveying large numbers of Americans about their views on a number of different policy issues, we already know in advance that unless something went seriously wrong with the sampling process, there would be two or three major clusters in the dataset roughly corresponding to liberal, conservative, and independent viewpoints. These clusters would represent a divide that actually exists among American voters. On the other hand, if we randomly generate data and perform k-means clustering, the algorithm will still collect each of the points into the pre-specified number of clusters, even though the points have real association with each other. Clustering in and of itself is useless if it cannot be used to make inferences about the population from which the clusters came. Checking within-cluster SSE is an important metric for finding if the cluster fits the data well, but ideally we want something more than this to check the quality of the clustering. This brings me to a discussion of the final technique I will be using, the Jaccardian statistic.

3 Benchmark

The Jaccardian similarity statistic is a measure of how similar two sets are to one another, and is defined as the ratio of the number of elements shared in common by the sets to the total number of objects in both sets together. It can be used to validate cluster stability by seeing whether any clusters made from a distorted form of the original dataset match closely with the clusters from the original data. (Hennig) By "distorted form," I mean the original dataset with some sort of reshuffling of observations or introduction of noise, so that models formed from the new data will not be exactly the same as from the unchanged version. This distortion can take the form of putting noise into the dataset (e.g., replacing some real observations with pseudo-observations with randomly chosen values for each variable), or by obtaining a bootstrap pseudo-sample. A bootstrap sample from the original data is created by filling a data frame of the same size as the original with randomly chosen observations from the real data, where these observations are chosen with replacement - e.g., real observation 3 might end up in the bootstrap sample 4 times, while real observations 88 and 149 don't show up at all. If the clusters in the original dataset are well separated from each other - if the distance between the edges of each of the clusters is large compared to the intra-cluster distance - then the clusters are probably robust enough to survive distortion of the data.

The Jaccard coefficient allows us to quantify how well clusters are preserved in the distorted dataset by counting the number of observations an original cluster shares with its counterpart in the new dataset, and then dividing this number by the total number of the points in each cluster that appear in the original and distorted datasets. The Hennig paper referenced above suggests a cut-off point of 0.5 as the Jaccardian critical value for evaluating whether a cluster is preserved - if at least 50% of the observations in the original cluster are preserved in the new cluster, the cluster is considered to be stable. Of course, the higher the Jaccardian coefficient, the more confidence we have in the stability of the cluster.

Using bootstrapping, we can calculate the Jaccard statistic for the original and new clusters many times (100 is standard for bootstrap simulations) to obtain a simulated distribution of the Jaccard coefficient for each cluster. To ensure fair comparison between the clusters, this method only compares the clustered points which are present in both the original and bootstrapped dataset. The algorithm then takes each cluster in the original dataset, compares it with every cluster in the bootstrapped dataset, and returns the maximum Jaccard coefficient - in other words, it finds which bootstrapped cluster is the original cluster's counterpart. By bootstrapping, we can repeat this process 100 times, and obtain a simulated distribution of the Jaccard coefficient for each cluster. Once we have this distribution, testing for whether groupings exist in the population reduces to a simple statistical hypothesis test.

4 Methodology

4.1 Data Preprocessing

The key accompanying the dataset gave a brief description of each variable. I used this key to pick out which variables represented traits of the participants collected at the event itself, and removed the rest. I then used an algorithm to determine which columns had the most missing observations, a problem that was present in many of the dataset variables. I pruned out the variables with many missing observations until about 95% of the observations (rows) had over 90% of their variables (columns) observed. To eliminate the remaining problem of missing data, I used the mean of each column to fill in the missing values. The mean was chosen because it is, by definition, the expected value of its variable. In absence of any other predictors, we would expect the variable to most likely take a value close to the mean.

Once the relevant variables had been selected and the missing data eliminated, I was still left with a significant problem. The dataset had 8,378 rows, but was only about 552 individuals; this was possible by providing information on the partner combinations for each of the individuals, so that ten or more rows might be dedicated to a single person, with each row corresponding to a different partner. Given that my objective in this analysis was to cluster individual people, I needed all of the observations to correspond to just one person. Many of the variables were only about the individual himself or herself and had nothing to do with the partner; these were of course the same for every row that corresponded to a given individual, because they were only collected once about that person. But all of the variables corresponding to partner ratings of an individual were different from row to row, and I could not afford to lose the information from these ratings.

My solution was to create an algorithm that went through the dataset and grouped all observations corresponding to a single individual together. A single row was created from these observations as the average of all of their column values. These averaged rows were then merged together into a single data frame with 552 rows, each corresponding to one of the original 552 participants. This approach had the double advantage of retaining all of the fixed variables for individuals (by multiplying them by the number of rows for that individual and dividing them by the same number, these values were left unchanged) and of preserving the information about the individuals' partners by their averages for partner-related values. I concluded by taking the logarithm of each column in this new dataset to standardize the distributions of each variable.

4.2 Implementation and Refinement

Let's recall what steps I discussed taking earlier in the report:

1. Principal component analysis to get the data into lower dimensional form to make clustering faster and more efficient
2. K-means clustering to sort each of the data points by their similarity to one another
3. Calculating the Jaccardian similarity distributions for each cluster and comparing their means with the critical value of 0.5 to see if the clusters are stable.

After preprocessing, the dataset still possessed a daunting 57 variables. However, performing these three steps with 57-dimensional data is absolutely possible - bioinformatics and finance applications regularly use principal component analysis with data that has thousands or even millions of dimensions. Once I log-transformed each one of the variables to move each of them onto the same scale before principal component analysis, I considered what the optimal number of principal components would be and made my choice based on a plot of the explained variance ratio for each number of principal components. Similar to my discussion of the elbow plot used in k-means clustering, the purpose of this plot was to show which number of components would strike a good balance between explaining a large amount of the variation in the data without using too many dimensions.

Almost 85% of the variation in the data is explained by the first three principal components alone. While it would be possible to explain even more of the variation by adding a few more components, I chose not to do so in the interests of clustering in as simple of a dataset as possible. Having the data be in only three dimensions allowed me to actually plot the data and look for clusters by eye, which helped to reinforce my choice of the proper number of clusters. Once the data was transformed, I plotted it in the new three dimensions (Figure 3).

Looking at this plot, there do seem to be several clusters of points in the dataset. It's not easy to tell by eye alone exactly how many clusters should be used to describe the data, which is why we turn to using the

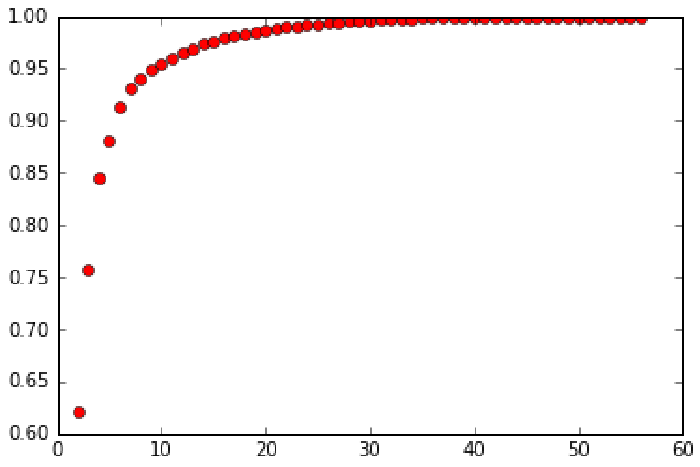

Figure 2: PCA Explained Variance Plot

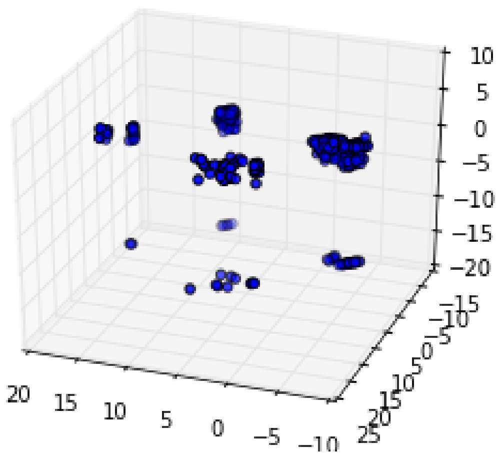

Figure 3: PCA Transform of Dataset in 3 Dimensions

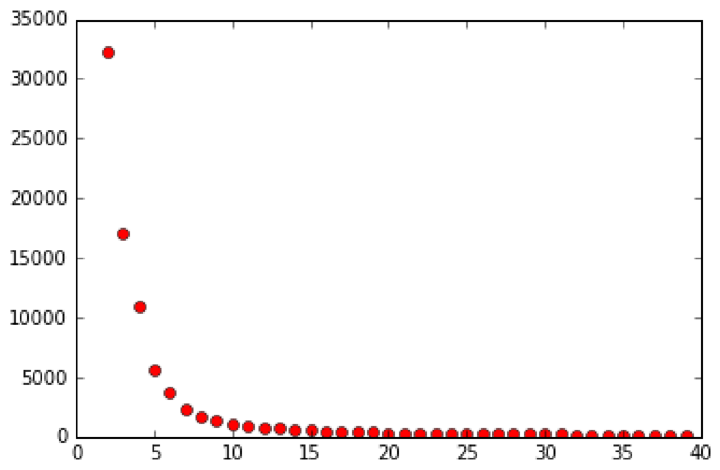

Figure 4: K-Means Clustering Elbow Plot

elbow plot to obtain a better idea of how many centroids to use in the k-means clustering.

Trying k-means clustering on this new three-dimensional data with several different values for k yields the elbow plot in Figure 4.

When evaluating the number of clusters to use from an elbow plot, the ideal case is one in which the number chosen sticks out at an angle from the other points like an elbow (hence the name), with the points to the left of it making a sharp decline, and the points to the right quickly evening out to an almost horizontal, slightly downward-sloping line. In a case like that, the plot is a good indication that the elbow is the best balance between minimizing within-cluster SSE and overfitting the data. In many cases, however, the choice is not as obvious, as is the case here. The points lie on a pretty smooth curve, and it is not easy to identify just one as the best candidate. In cases like this, some trial and error is necessary. Something between 4 and 6 seems reasonable; for my first attempt at fitting a clustering model, I tried 4 clusters. Let us examine the plot of the data colored by cluster in Figure 5.

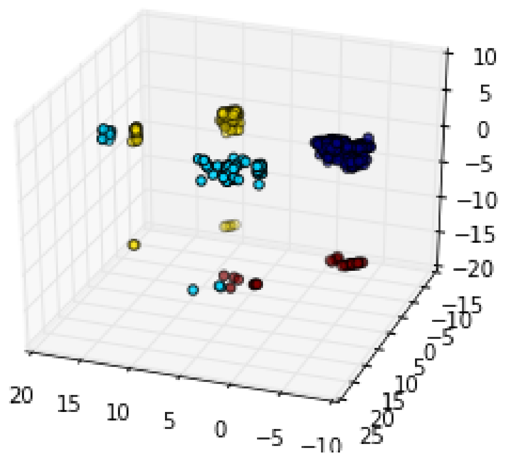

Figure 5: K-Means Clustering - 4 Clusters

4 clusters do not seem to capture the divisions in the dataset well - as you can see, groups of points which

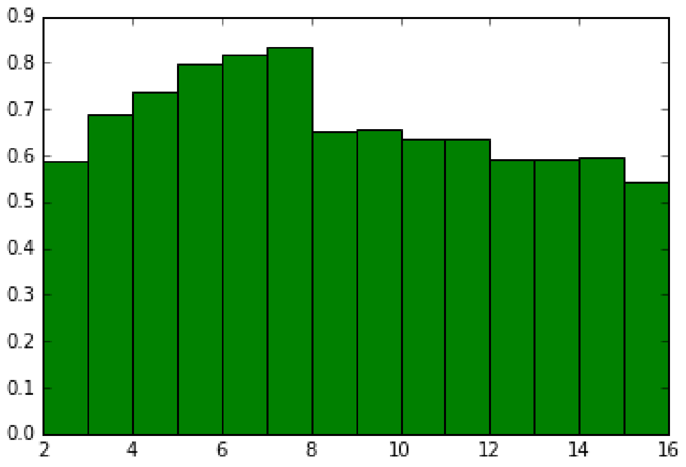

Figure 6: Mean Silhouette Score Bar Chart

are close to one another have different colors while other groups which are farther away have the same color - this is the case with both the yellow and light blue colored points. The odd fit indicates that 4 clusters is not the best fit to the data. Given the ambiguity of the elbow plot, it is a good idea to supplement our use of it with another method for determining the ideal number of clusters: the mean silhouette score. If b is the distance between a given cluster and the cluster nearest to it, and a is the average intra-cluster distance (i.e., the average distance from a point in the cluster to the centroid), then the silhouette score for a given cluster is:

$$\frac{b-a}{\max(a,b)} \tag{1}$$

The silhouette score, then, represents how well divided the cluster is from other clusters. If b is larger than a, the ratio will be positive, indicating that the nearest cluster is farther away from the cluster under consideration than any of its own points are from its centroid. If a is close to b, it means that the ratio will be close to 0, and the clusters are likely overlapping. In the worst case scenario, if a is actually greater than b, the ratio will be negative, indicating that some of the cluster's points are actually farther away from the centroid than the next-nearest cluster is. In short, the closer the silhouette score is to 1, the better divided the cluster is from the others. The mean silhouette score is just the average of the silhouette scores for all the clusters, and represents how good of a job the clustering algorithm did in assigning the points to distinct groups. We can use the mean silhouette score to get a better idea of how many clusters are ideal for this dataset by checking to see what number of clusters has the best score without overfitting. While determining when a clustering algorithm has overfit a dataset can be difficult to do, having a graphic to examine can be a good reality check. In the plot of the data above, the maximum reasonable number of clusters there seem to be is 8, although the actual number could certainly be less. Let's look at the mean silhouette scores for numbers of clusters between 2 and 15 in Figure 6.

As you can see, the number of clusters with the best mean silhouette score is 7. (The numbers on the x-axis are on the left side of the bars they accompany.) The shape of the distribution depicted here represents the balance between underfitting and overfitting: when the model does not adequately account for the variance in the data, the clusters are not going to be well-defined and will consequently have too much of a spread. On the other hand, when the model overfits the data, it will produce too many clusters in a small region, which means that the distance between clusters will be relatively large compared to the intra-cluster distance.

Now that we know that the optimal number of clusters is 7, let us look at the updated clustering model I fit with the new parameter in Figure 7.

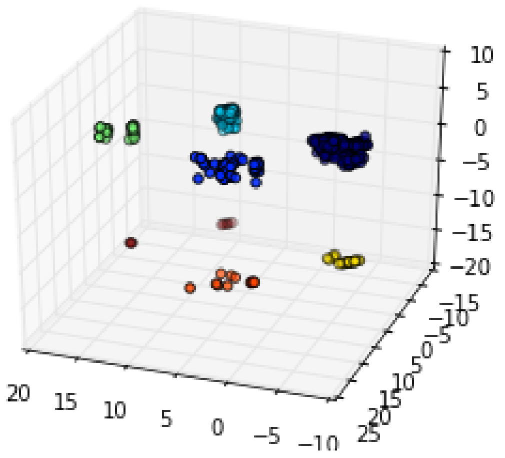

Figure 7: K-Means Clustering - 7 Clusters

This seems like a much more reasonable assignment of clusters based on the shape of the data, and an examination of the mean silhouette score indicates that this number of clusters provides the best fit to the data. Now that the clustering has been refined, the model-fitting stage of the analysis is finished. Next, I will discuss the results of testing the quality of the cluster-fit using Jaccardian similarity.

5 Results

With the clusters created and refined, let's return to the original question I posed at the beginning of the report: can college students on the dating market be reliably sorted into categories based on large numbers of variables that wouldn't usually be explicitly put together? We've established that the clustering developed so far fits the given dataset well. But remember that our question isn't ultimately about *this particular dataset*; it's about the population of college students as a whole. What we want to know now, and what will answer our original question, is whether the clusters we see in this dataset are likely to reoccur in other samples drawn from the same population.

As I stated earlier, I am using bootstrapping to obtain a simulated distribution of the Jaccard coefficients for each of the 7 clusters. The mean of these distributions will then be compared to the critical value of 0.5; the clusters whose coefficients exceed that value will be judged to be stable. Stable clusters will be considered representative of real groupings in the population rather than mere artifacts of this particular dataset.

I ran the clustering bootstrap 100 times to obtain 100 estimated values for the Jaccard statistic for each bootstrap cluster / original cluster pair. As you can see by looking at the table of the 95% confidence intervals for the means of each distribution, every cluster had a very high average Jaccardian value, in most cases approaching or equal to 100%. Even the cluster with the smallest lower bound for its confidence interval at the bottom of the table still had a Jaccardian value at that bound over 20% higher than what was needed to conclude that the cluster was stable over multiple samples. The confidence interval for the mean was obtained by adding and subtracting a margin of error from the mean equal to the standard deviation of each distribution multiplied by the distribution's 95th percentile, as is standard in the construction of confidence intervals. (In all but two cases the addition of the margin of error to the mean produced a value greater than 1.0. Such a value has, of course, no practical interpretation, as you can't have more than 100% similarity, so those values were rounded down to 1.0.)

Given the number of variables considered (57), the number of people (552), and the number of bootstrap

Lower Bound	Mean	Upper Bound
0.87928	0.97404	1.0
1.0	1.0	1.0
0.9887	0.99763	1.0
0.9901	0.9985	1.0
1.0	1.0	1.0
0.9403	0.9919	1.0
0.7134	0.9388	1.0

Table 2: Confidence Interval Table for Jaccardian Distributions

replications (100), the high mean values for the Jaccard coefficients for each cluster give us very good reason to infer that the population of young singles from which the sample came does exhibit consistent clustering over a number of seemingly-unrelated traits. The intervals are well above the cut-off point for the dissolution of clusters, and we can thus conclude that the clusters are statistically significant indicators of real groupings in the population, and not merely the effects of noise in the data.

6 Conclusion

6.1 Free-form Visualization and Reflection

As you read these reflections on the project, you are invited to consider the graphs included on this page and the following. Each of these depicts the clustering of one of three of the bootstrap samples; as can be seen by comparing these three with each other and all with the clustering of the original dataset, the clusters were just as stable as their high Jaccardian values indicated, changing very little from one replication to another. Comparing these pictures allows you to see slight changes in the shape and composition of each cluster from one pseudo-sample to the next, but the size of these changes is dwarfed by the consistency in the cluster divisions observed from one sample to the next. Of particular interest for those looking for how the samples changed by bootstrap is the cluster in the bottom center foreground, colored brown in the first image. The shape and size of this cluster is quite similar in the second and third images, but differs in these two from the first bootstrap sample. The relative homogeneity in shape of the clusters is a good indicator of their stability, but what makes their permanence especially clear is their *distances* from one another - the between-cluster distances are much greater than the intra-cluster distances in every case. This would be a very improbable and difficult to replicate result if the clustering was based primarily on noise in the data rather than real features in the population.

By far the most difficult element of the project was simply getting the data into the format where I could actually perform principal component analysis and clustering. The dataset came in a badly-organized form, where rows represented interactions instead of people (hence why the original dataset had over 8,000 rows when there were only 552 participants - there were many rows per participant), some of the variables were singular per person and thus constant in the rows for each person while others were specific to the interaction and changed by row, and large numbers of variables had so much missing data that they showed up in less than 15% of the observations. Sorting through this data to get rid of the variables with little information or which weren't relevant and then getting the dataset into a form where the rows represented people, not observations, took easily 30% of the total time I spent on the

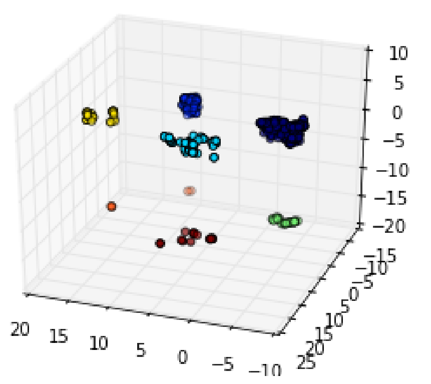

Figure 8: Bootstrap Sample - Example 1

project.

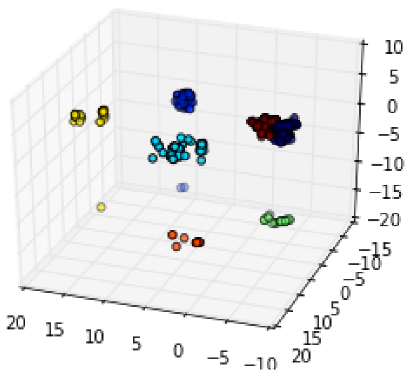

Figure 9: Bootstrap Sample - Example 2

As difficult as this was, once the data was actually put into a usable format, the actual analysis flowed fairly smoothly. The next-most challenging aspect was writing the bootstrapping loop to get the Jaccard coefficients. The **R** programming language has a package (clusterboost) which has the cluster validation via Jaccard similarity already implemented; had this project been done in **R**, that part of the coding would have probably taken a tenth of the time. The upside, however, is that I learned far more about this method of cluster validation by actually coding all of it myself than I would have if I had simply used a ready-made version of the function. Making sure that the points were compared in the correct domain was especially difficult - I had to make sure that the points in the bootstrap clusters were being referenced by the indices in the *original data* that these bootstrapped points had come from.

A bug in an earlier version of the code that I wrote without noticing compared the original and bootstrapped points across clusters with the original points being referenced (of course) by their indices in the dataset, and the bootstrapped points being referenced by their indices in the bootstrap dataset - so for instance, if point 157 from the original dataset was in cluster 1, and point 157 was point 3 in the bootstrap dataset, the buggy code interpreted these points (157 and 3) as being two different observations, even though bootstrap 3 was actually original 157. This led the bootstrap loop to underestimate the Jaccardian values, giving coefficients lower than 5% in some cases. I had almost concluded that there really were no clusters in the data that weren't noise-dependent until I hit upon the idea of plotting the bootstrap samples with the clusters colors (i.e., the graphics included in this concluding section of the report). As soon as I saw a few of those plots, I realized that the clusters were staying incredibly stable across multiple pseudo-samples, and that something had to be wrong with the code that was producing the low Jaccardian values. Once I finally corrected the bug, the Jaccardian values shot up to what I had already guessed they would be based on the clustering plots.

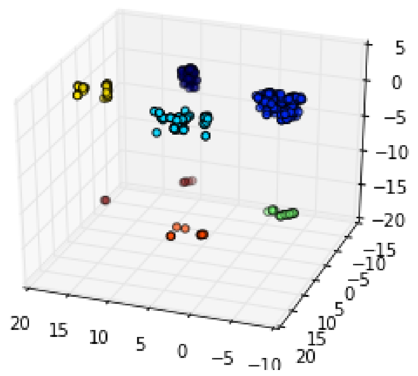

Figure 10: Bootstrap Sample - Example 3

7 Improvement

The best way I can see to improve the solution I've obtained here would be to take the results I've gotten here and try to start modeling the dating preferences and successes of each of the groups, as well as describing what characterizes these clusters across the range of their variables. I didn't print out the cluster centers in the original dataset form here, because they consisted of 57 variables - even showing an array of that size, much less trying to explain the relationships between all of the variables in a coherent narrative, would be very difficult to do and would require much more space than is available in this report. This isn't really surprising, however - if the dataset had few enough variables that the cluster centers could be easily described, using dimensionality reduction and automatic clustering probably wouldn't have been necessary. The whole point of getting 57 dimensions down to 3 and then clustering is that the cluster relationships in the original high-dimensional data are too subtle for us to perceive or easily describe. Giving an adequate description of what makes each cluster unique would require doing more research on how various types of variables (e.g., preferred activities, personality traits, career, etc.) relate to each other in the young singles population.

Additionally, it would be interesting to see whether people from a given cluster of people are more likely to get dates with people from specific clusters (whether their own or others), or whether some clusters are more likely overall to get dates, regardless of the cluster to which the date belongs. This dataset did not have the information needed to answer these kinds of questions - there wasn't enough information given on each participant's partners to be able to assign them to the clusters or identify them as other participants in the dataset - but a future study would do well to include enough information to show which people in the dataset matched with which. A study that builds upon what I've discovered here and incorporates such matching information could shed light on *why* some people seem to be more likely to get dates, and answer this question on a level more informative and satisfying than a mere appeal to looks or general notions such as "vibes" or "charisma."

8 References

The dataset for this project is hosted by Kaggle and was uploaded by Anna Montoya. The link to it may be found here: https://www.kaggle.com/annavictoria/speed-dating-experiment

Hastie, Trevor et al. *Introduction to Statistical Learning in R*, 2013. Accessed online at http://www-bcf.usc.edu/ gareth/ISL/ISLR%20Fourth%20Printing.pdf

Sklearn User Guide, accessed at http://scikit-learn.org/stable/modules/clustering.html#k-means

Hennig, Christian. "Cluster-wise assessment of cluster stability." *Elsevier Science*
Accessed at http://www.homepages.ucl.ac.uk/ ucakche/papers/clusta.pdf