Getting Handsy

Introduction: Clustering Hand Posture Data

This report explores unsupervised learning and clustering using data collected from a motion capture system. The system was used to record 14 different users performing 5 distinct hand postures with markers attached to a left-handed glove. A set of markers on the back of the glove was used to establish a local coordinate system for the hand, and 11 additional markers were attached the the thumb and fingers of the glove. 3 markers were attached to the thumb with one above the thumbnail and the other two on the knuckles. Finally, 2 markers were attached to each finger with one above the fingernail and the other in the middle of the finger. A total of 36 features were collected from the camera system. Two other variables in the dataset are the ID of the user and the posture that the user made.

The data were partially preprocessed. First, all markers were transformed to the local coordinate system of the record containing them. Second, each transformed marker with a norm greater than 200 millimeters was eliminated. Finally, any record that contained fewer than three markers was removed. A few issues with the data are worth noting. Based on the manner in which the data were captured, it is likely that, for a given record and user, there exists a near duplicate record originating from the same user. In addition, there are many instances of missing data in the feature set. These instances were denoted with a ? in the dataset. Finally, there is the potential for imbalanced classes, as there is no guarantee that each user and/or posture is represented with equal frequency in the dataset. The dataset contains 78,095 rows and 38 columns. Each row corresponds to a single instant or frame as recorded by the camera system. The data are represented in the following manner:

1. Class (Integer): The hand posture of the given obervation, with 1=Fist (with thumb out), 2=Stop (hand flat), 3=Point1 (point with index finger), 4=Point2 (point with index and middle fingers), 5=Grab (fingers curled as if to grab).
2. User (Integer): The ID of the user that contributed the record.
3. X0, Y0, Z0, X1, Y1, Z1, . . . , X11, Y11, Z11 (Real): The x-coordinate, y-coordinate and z-coordinate of the twelve unlabeled marker positions.

Missing Data Imputation

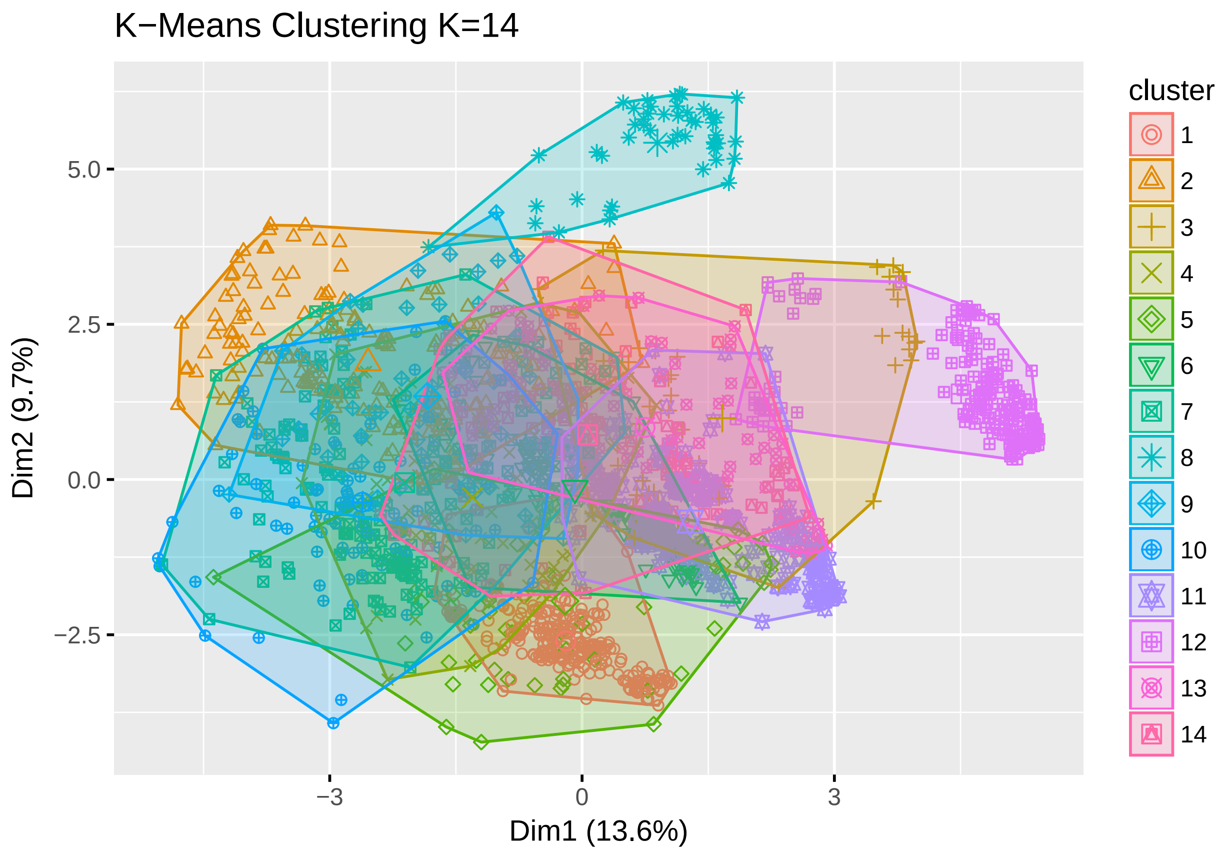
We begin by imputing the missing data values. Given the knowledge of how the data was collected, we can hypothesize that there are two ways in which the data might cluster together: by user and by posture. Perhaps the users have significantly different heights and/or hand sizes, resulting in the data generated by each user to be distinct from each other. Or perhaps the hand postures are sufficiently unique such that the markers on the glove tend to be grouped together by posture, regardless of who the user is. We will examine these hypotheses to see if either one provides a reasonable way to impute the data.

Let's start by imputing missing data by User. There are a couple of cases to consider: 1. There are some number of missing values for a single coordinate (column) for a user. + In this case, we will simply impute that missing value with that coordinate (column) mean. 2. All of the values for a coordinate (column) for a user are missing. + In this case, we will impute those missing column values with the overall mean for all coordinates for that user.

But what if the data is grouped by posture instead? Let's group the data by Class, then impute missing values by that. Again, we consider the same cases as before: 1. There are some number of missing values for a single coordinate (column) for a posture. + In this case, we will simply impute that missing value with that coordinate (column) mean. 2. All of the values for a coordinate (column) for a posture are missing. + In this case, we will impute those missing column values with the overall mean for all coordinates for that posture.

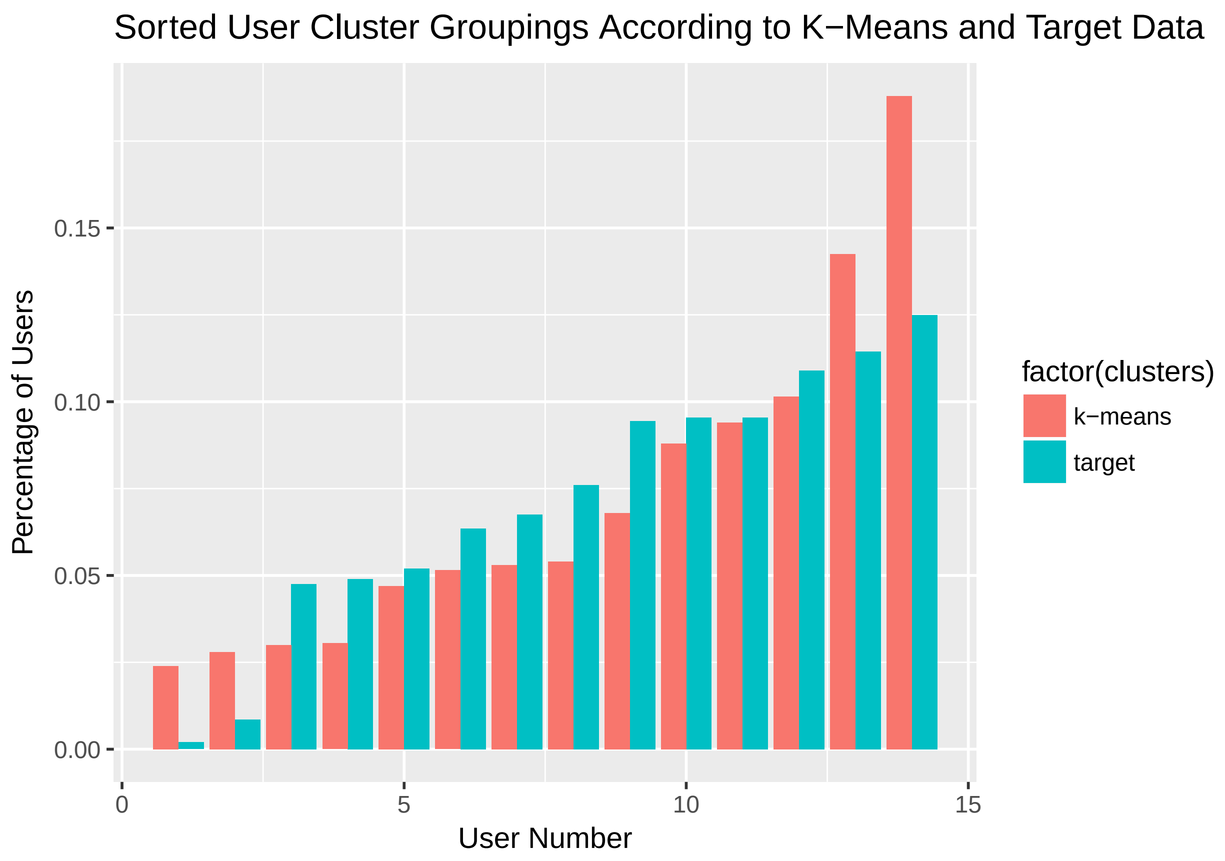
Clustering with k-means

Now that the missing values have been imputed, we can investigate our hypotheses by examining how well the data clusters by user and by posture. We will first cluster using k-means with 14 centroids (because there are 14 users) on the user-grouped data.



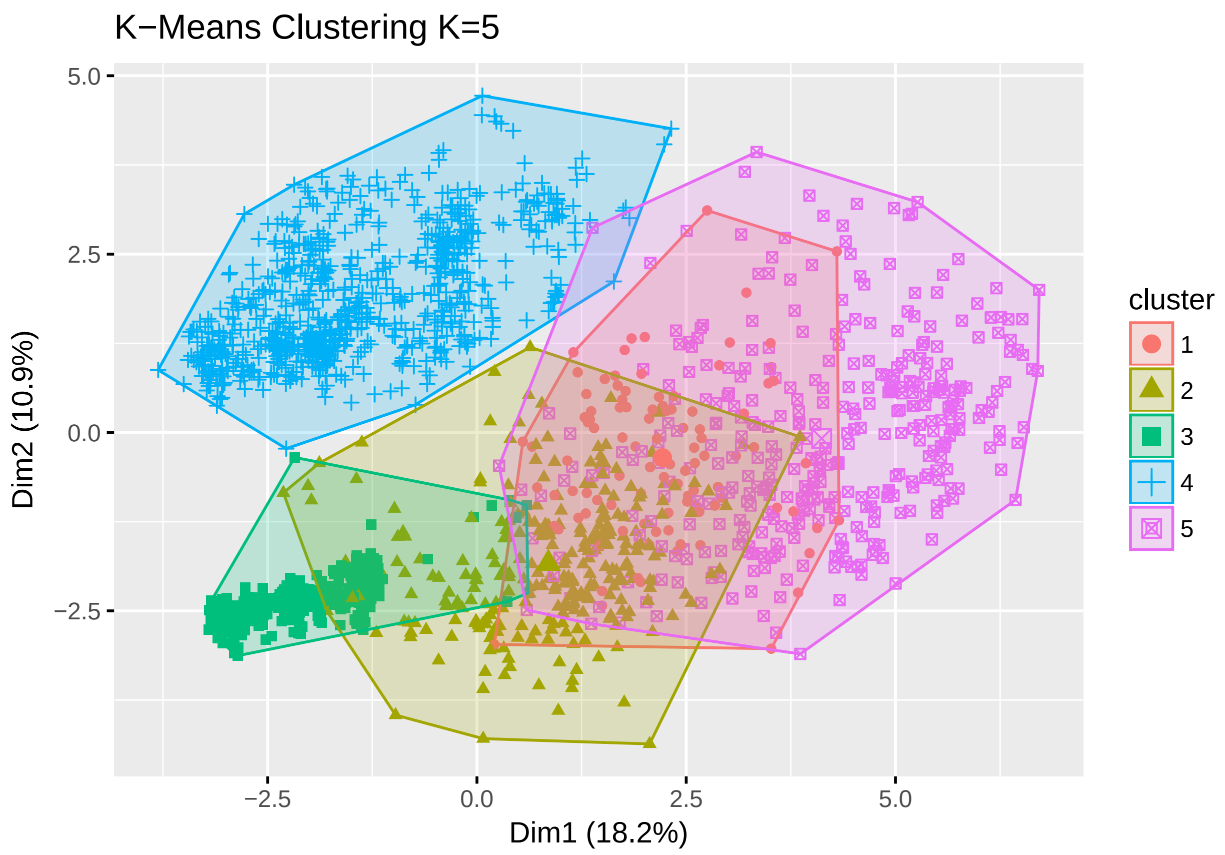
It would appear that the first two components explain less than 25% of the variation in the data, and that the 14 clusters overlap quite a bit, preventing the formation of distinct clusters of users.

We then compare these results to the actual user classifications to determine how well kmeans performed.

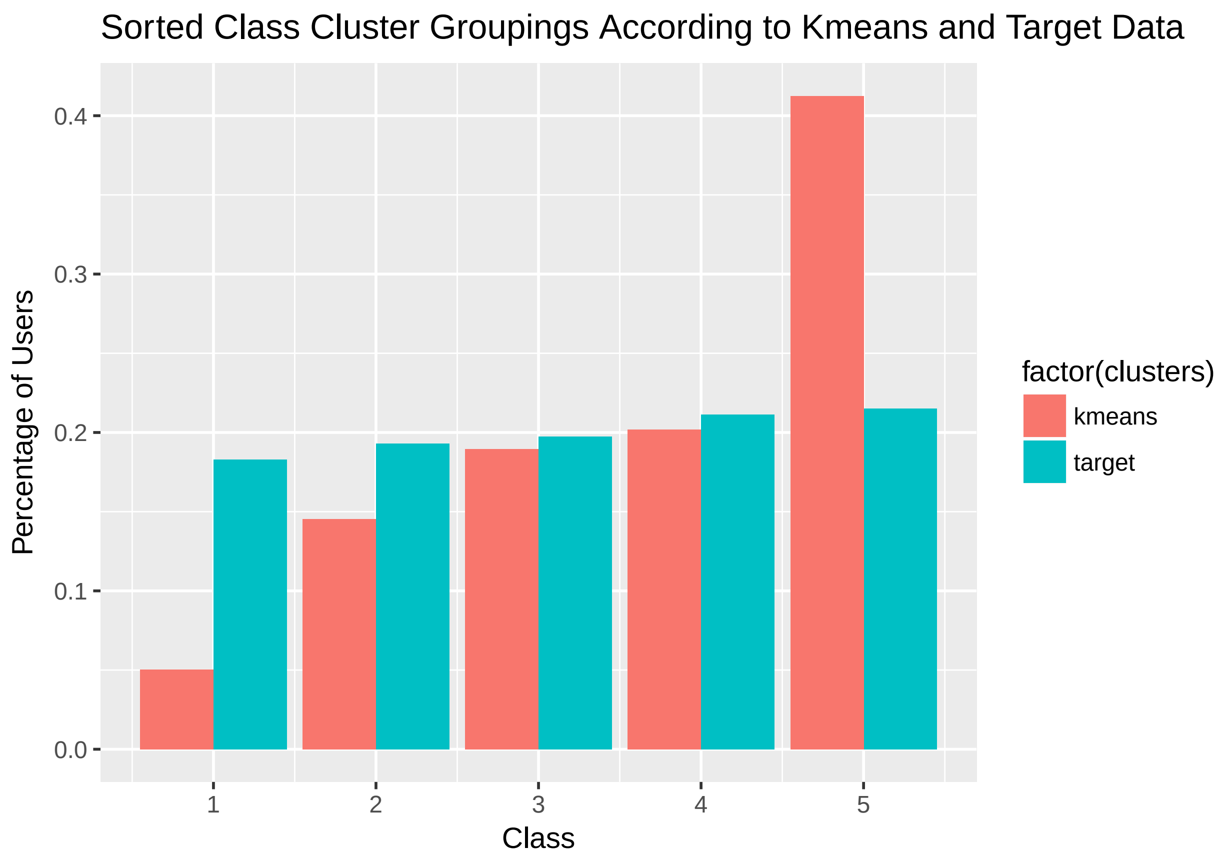


Looking at the bar plot, it does not appear that the k-means method captures the true clustering of data, especially for those clusters that have far fewer users than others, though it comes close for clusters with more users.

Next, we run k-means with five centroids (for the five postures) on the data grouped by posture.



The two principal components here explain just under 30% of the variance in the data. The clusters look slightly more separated than those of the user-grouped data. However, this could also simply be a product of there being fewer clusters made from the posture-grouped data.



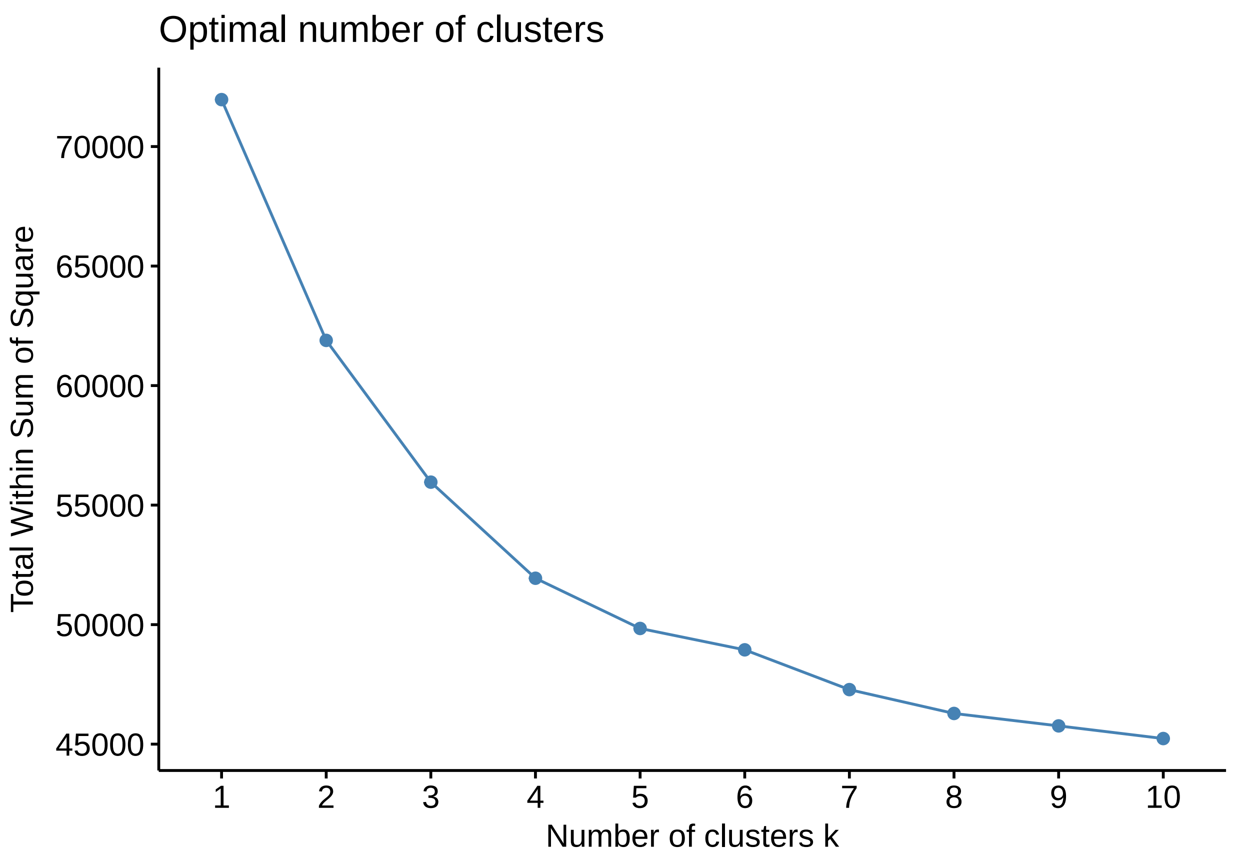
We see that the k-means clusters seem to more closely match the target clusters when grouped by posture (the most glaring exceptions being postures 1 and 5), and they also yield less crowded clustering. This suggests that the data clusters by class rather than by user, which makes sense, since one user's hand gestures would look more or less like those of another user. In other words, fists appear similar regardless of whose they may be, while an individual's fist looks different from their open palm.

Evaluating Clustering

But we must also determine the most appropriate number of clusters for our model. There are several methods we can try, then compare results.

Elbow Method

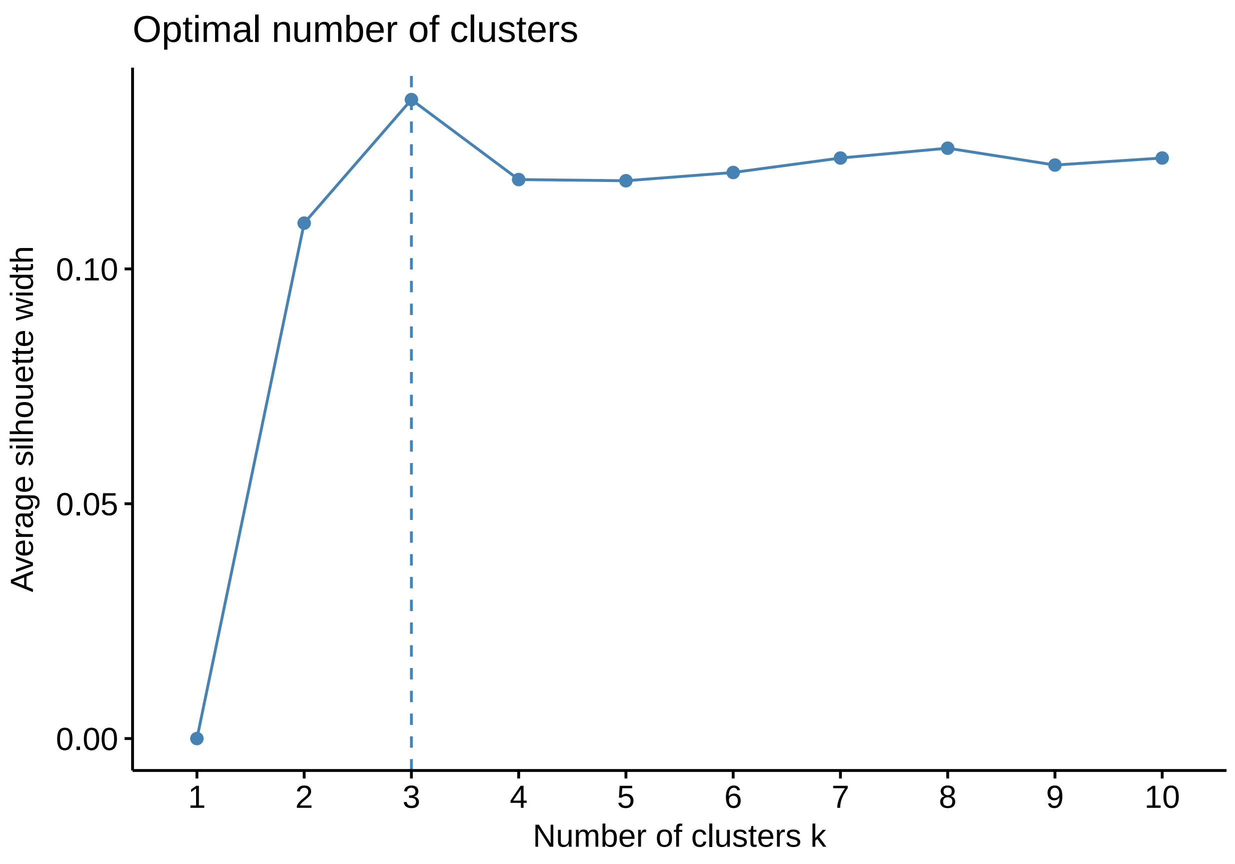
One of the simpler methods is known as the elbow method. In this method, we determine the percentage of variance explained by the k-means models with a variety of k-values (clusters). We will find that the first few clusters will explain a great deal of variance, but at some point the marginal gain will drop. By graphing the percentages of variance explained with each number of clusters, we will see a change in slope, resulting in an elbow-like shape.



We notice the most significant change in slopes at `k = 2` or perhaps `k = 3`. In other words, much less variance is explained by adding a fourth cluster compared to having the first three, so the elbow method would suggest that the optimal value for `k` is `2` or `3`.

Silhouette Method

However, the elbow method can be ambiguous and not terribly reliable, so we can also consult the silhouette method: we compare average "silhouette widths" across different numbers of clusters. The silhouette width ranges from -1 to 1 and is a measure of how similar an object is to its own cluster compared to other clusters, where a high value indicates that a given observation is well matched to its own cluster and poorly matched to neighboring clusters. If most observations have a high silhouette width, then the clustering configuration is appropriate. If many points have low or negative values, then the configutation may have too many or too few clusters.



Here, we see that 3 clusters yields the highest average silhouette width of a bit more than 0.10. Both methods suggest that `3` clusters would be optimal for clustering using k-means.

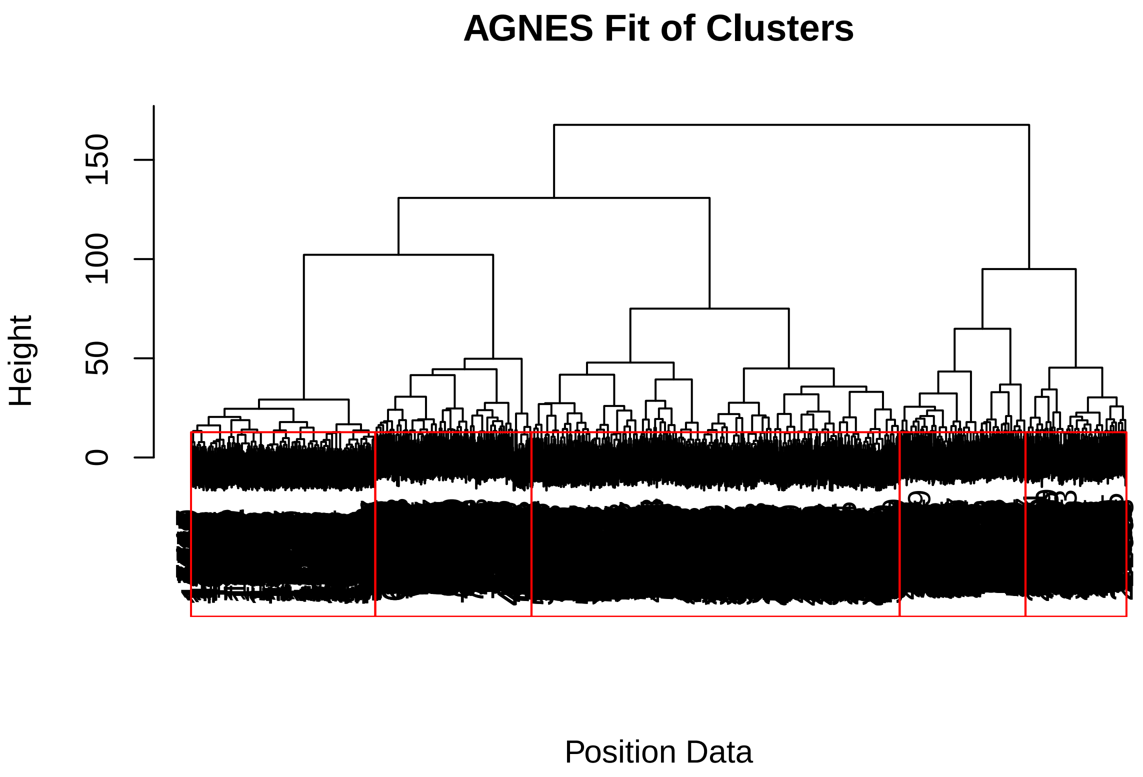
Other Clustering Algorithms

3 clusters might work best for k-means, but what about other clustering methods? We'll experiment with a few on the data grouped by postures and see what patterns we can observe.

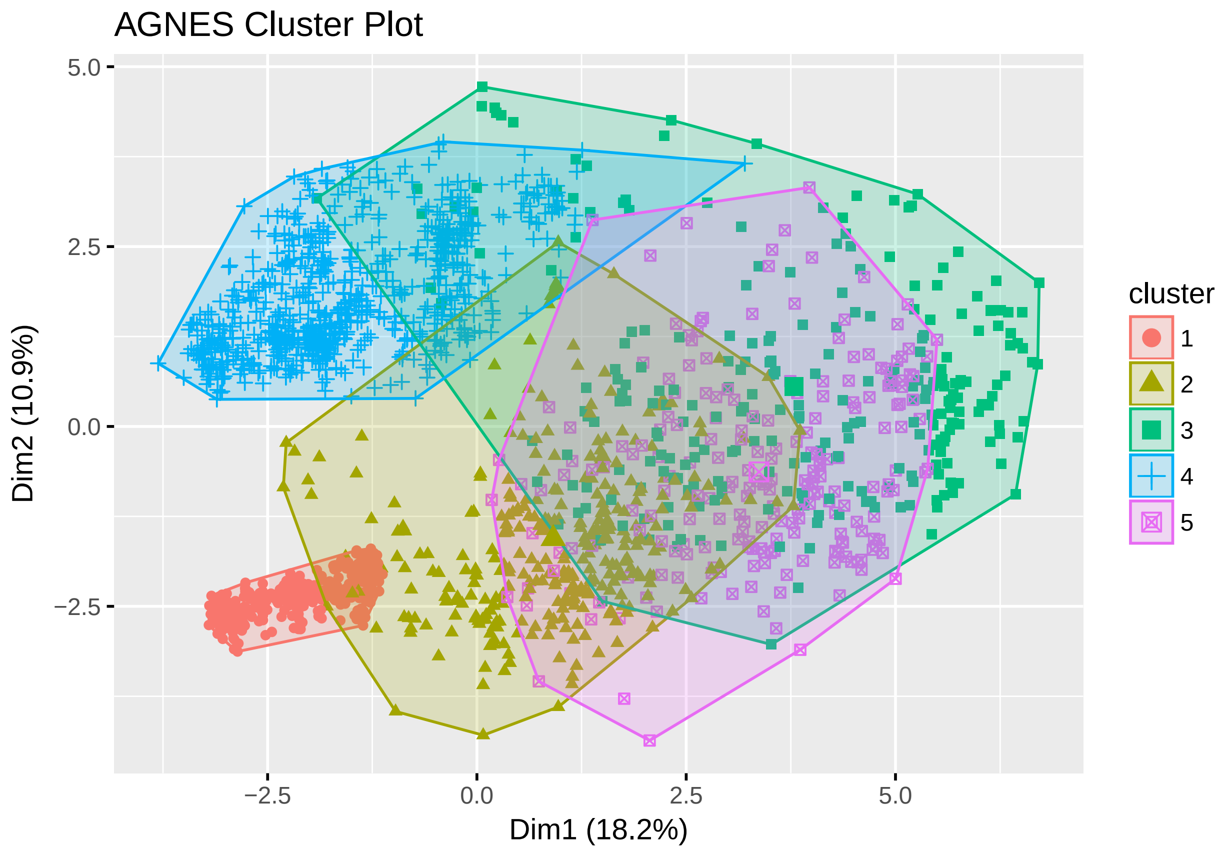
Agglomerative Hierarchical Clustering

Hierarchical clustering is a method of clustering in which similar observations are grouped into clusters. In the case of agglomerative hierarchical clustering, it begins by treating each observation as a separate cluster, then repeatedly executes the following steps: 1. Identify the two clusters that are closest together 2. Merge the two most similar clusters

Let's try this first.



Looking at how the branches of the tree split, it is easy to distinguish five distinct clusters at roughly `Height = 70`.

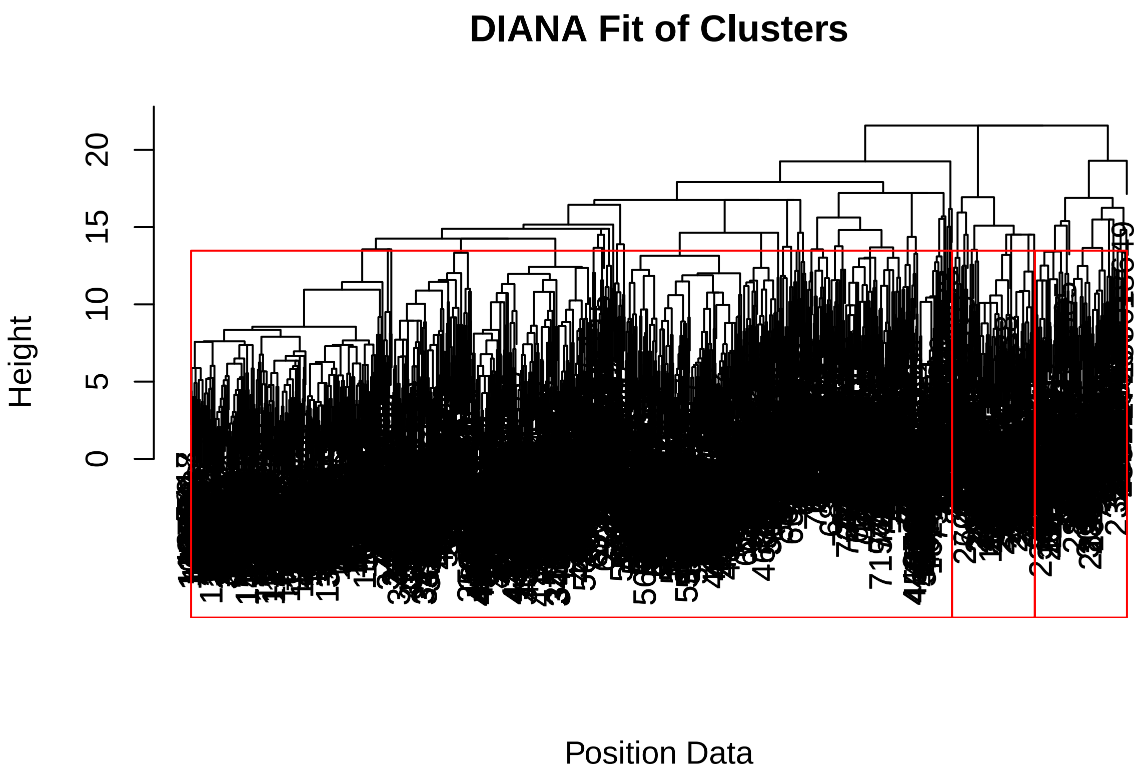


Again, the two principal components here explain just under 30% of the variance in the data. The scatterplot, however, shows quite a bit of overlap between the 5 clusters, the notable exception being cluster 1.

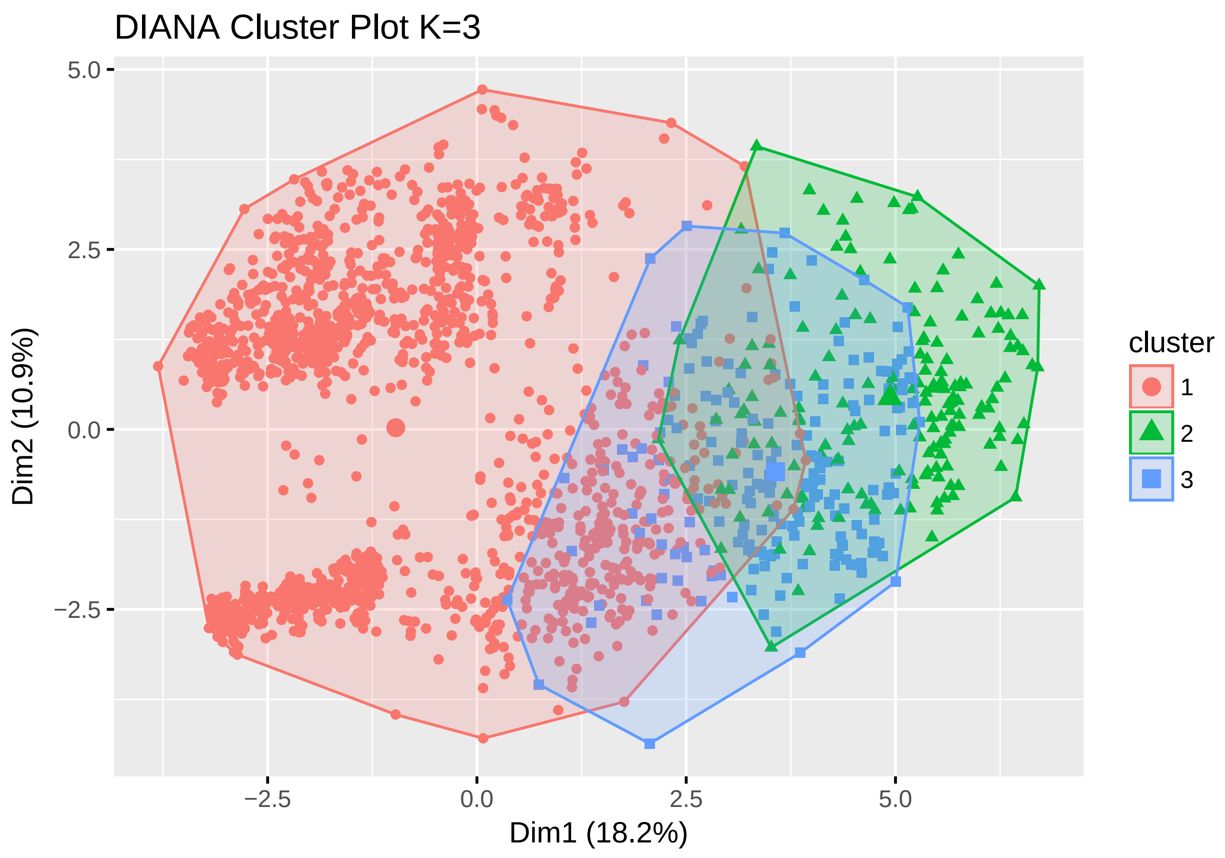
Divisive Hierarchical Clustering

In addition to agglomerative hierarchical clustering, there is also divisive hierarchical clustering, which essentially takes the same steps as agglomerative clustering but in the reverse order: it begins by grouping all observations into a single cluster, and then successively splitting the clusters.

We'll try that next.

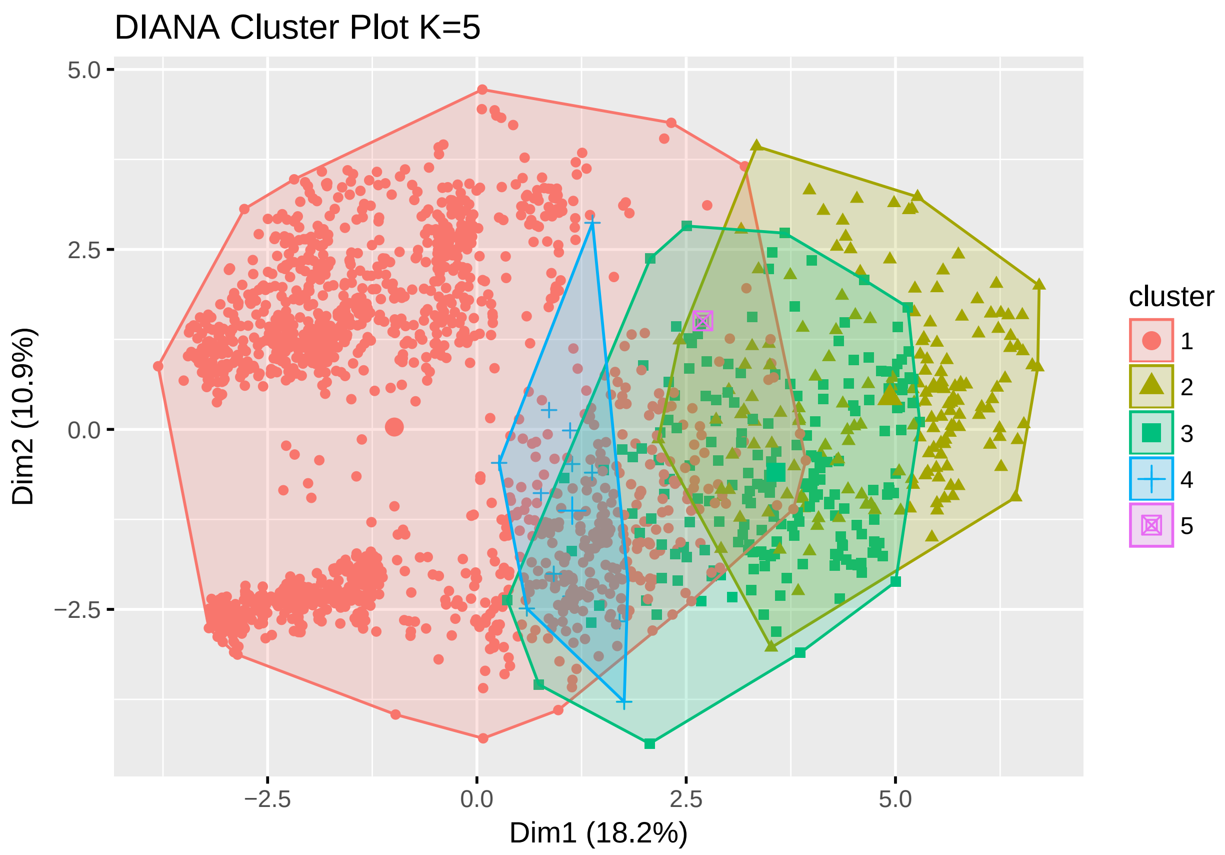


Here, it looks like divisive clustering yields 3 clusters from the tree plot. But does the scatterplot give any other information?



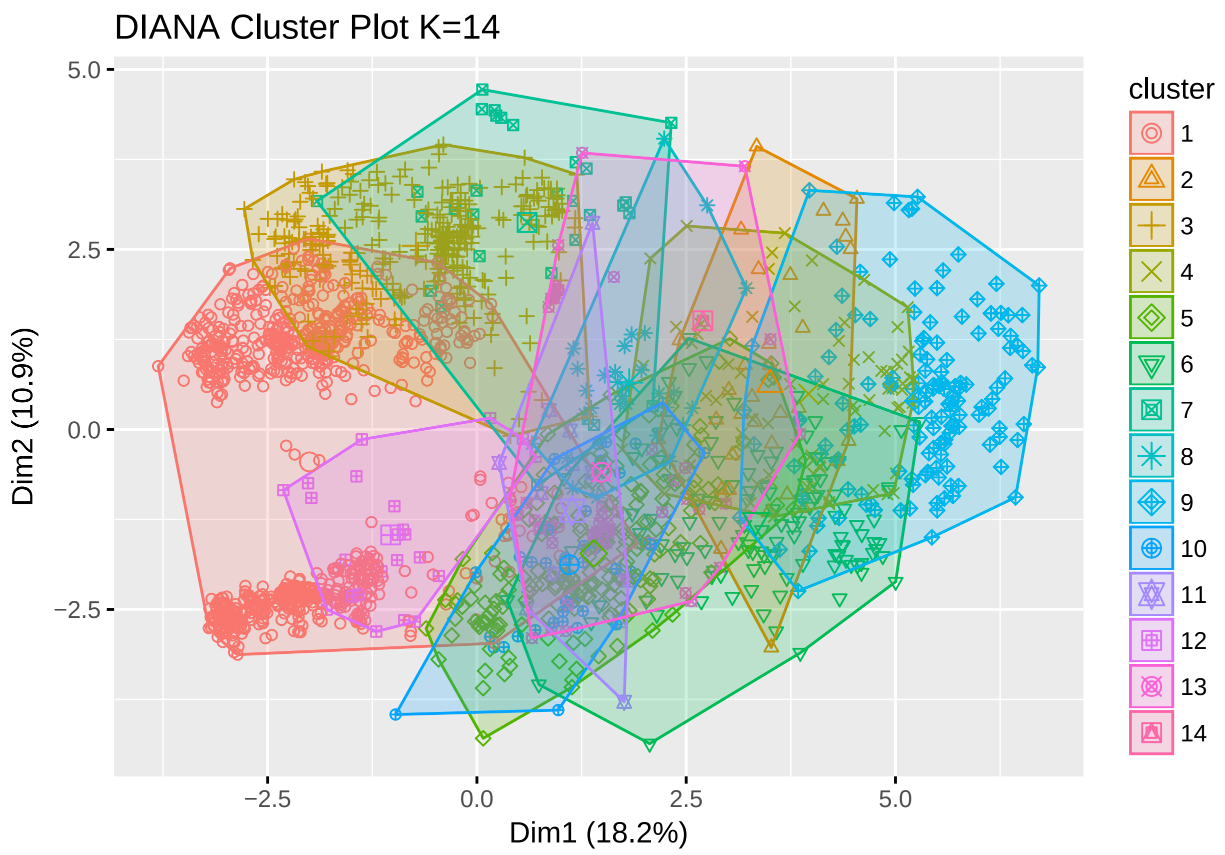
There is still quite a bit of overlap, though the two principal components still explain just under 30% of the data's variance.

Out of curiosity, let's take a look at the scatterplot for divisive clustering with 5 clusters.



Still a great deal of overlap and crowding in the center, and it seems that cluster 5 consists of just one observation, so five clusters does not look great for divisive clustering at all.

What if we try clustering by the 14 users instead?

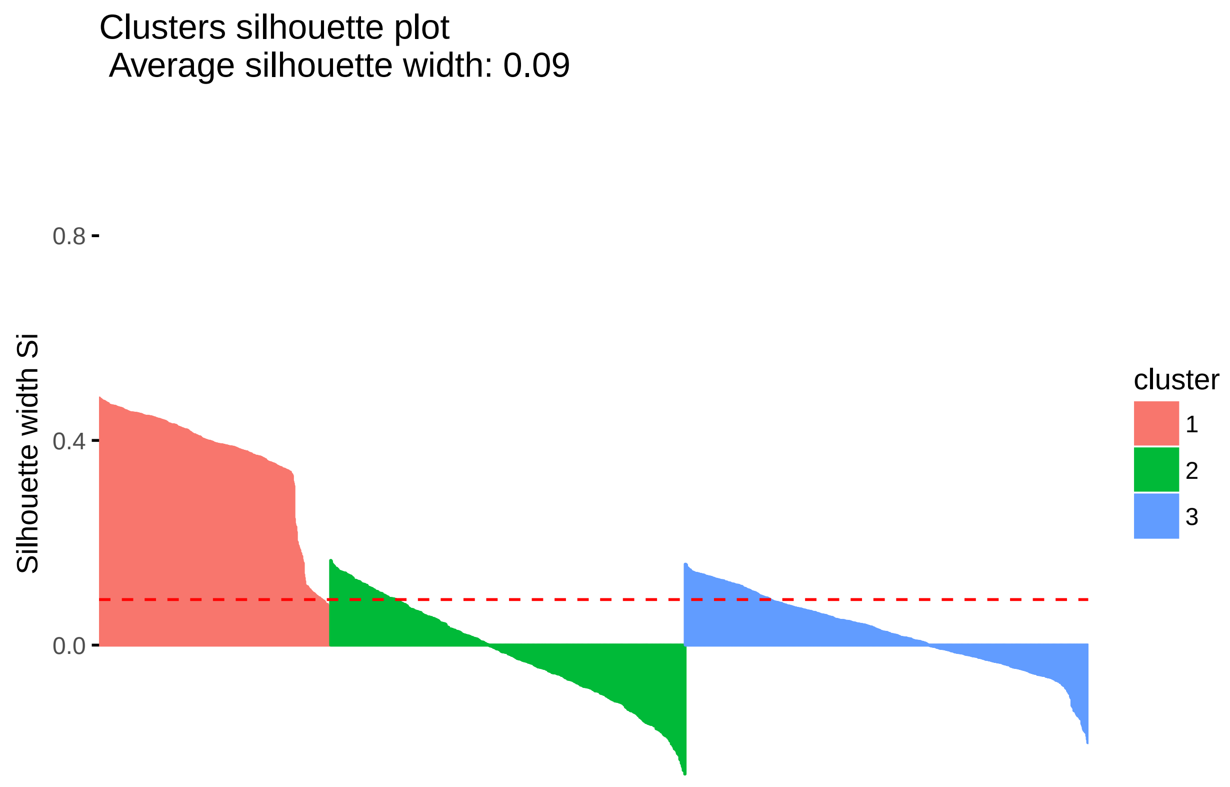


Quite a bit of overlap, as would make sense when we add more clusters. What is more concerning are the clusters that appear to have only one observation. We know there are more than one obersvation for each user, so this seems to not reflect the actual data clustering at all.

Fuzzy Clustering

Other than hierarchical clustering, we can also try soft clustering, or fuzzy clustering, which allows observations to be a part of more than one cluster. Given the amount of overlap we have seen thus far, this could be a useful way to cluster our observations.

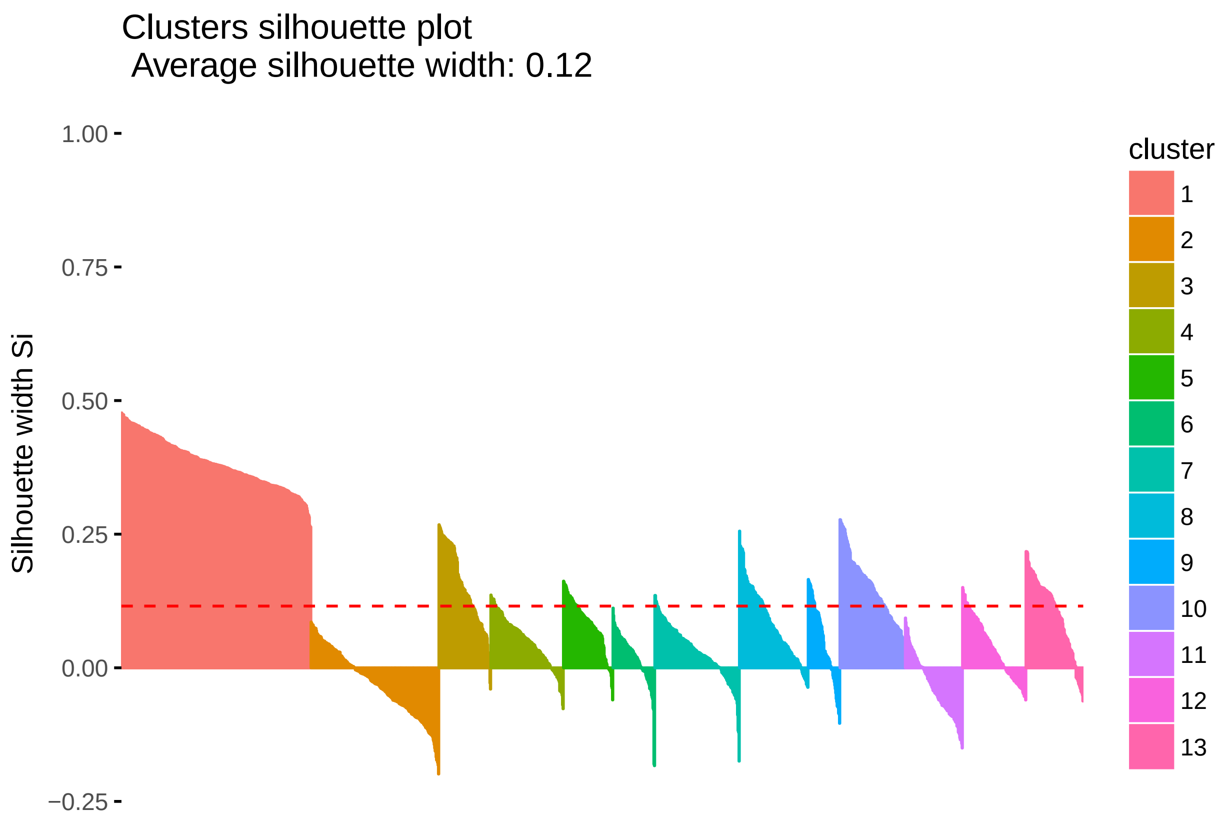
```
##   cluster size ave.sil.width
## 1       1  468          0.36
## 2       2  717         -0.02
## 3       3  815          0.02
```



Despite attempting to have 5 clusters, the `fanny` function seems to prefer 3, as we saw with the `diana` function as well. The average silhouette width of 0.09 is a bit low, but quite a few widths look to be on the higher positive side, so 3 clusters seems perfectly appropriate.

It is also worth noting that cluster 1 is overwhelmingly prefered to the other two - perhaps adding more clusters will help spread the wealth a bit?

```
##    cluster size ave.sil.width
## 1        1  394          0.38
## 2        2  266         -0.03
## 3        3  108          0.15
## 4        4  151          0.05
## 5        5  103          0.08
## 6        6   87          0.02
## 7        7  176          0.03
## 8        8  143          0.08
## 9        9   66          0.05
## 10      10  136          0.15
## 11      11  119         -0.04
## 12      12  132          0.04
## 13      13  119          0.10
```



Cluster 1 remains more favored than the other clusters, but the average silhouette width has significantly increased to 0.12 from 0.09, so perhaps clustering by user works better for fuzzy clustering.