CLUSTERING ANALYSIS OF VOWEL DATASET

Using K-Means Algorithm

AUGUST 6, 2020

BY VANESSA FOTSO

Introduction

Speech recognition is an important field that has long been a target for many studies. It is defined as the transcription of speech signals to a sequence of word or text (Jennings, 2006). Speech recognition technologies play a vital role in people's lives. Most importantly, they are a great aid for the visually and hearing-impaired community. One important part of speech recognition is the classification of voiced phonemes. A phoneme is what allows us to distinguish a word from another one in a given language. Every single phoneme has its own unique pattern and different phonemes can be differentiated in terms of their formant frequencies (Sarma M. and Sarma K., 2012). Plus, there are many variations in the way different speakers produce the same phoneme. Human naturally distinguish those sounds; however, this represents a challenging task in speech recognition.

The ability to correctly segment speech and identify the speaker provides many advantages in speech processing applications (Daqrouq and Tutunji, 2015). In previous studies, speaker identification systems have been implemented by analyzing the voiced and unvoiced elements, as well as the speech energy distribution. Some techniques employed in those systems for speaker classification include linear predictive modeling, discrete wavelet transform and wavelet packet transform (Daqrouq and Tutunji, 2015). However, there are times when there is little amount of or no labeled data points in a large dataset. In this case, clustering could be a good approach for the segmentation process. Clustering methods are unsupervised learning techniques that partition a dataset into groups of data points based on their similarities. This study aims to carry out vowel recognition using K-means clustering technique. The goal using this technique is to be able to segment individual vowel phoneme in a given speech using a set of specific characteristics, as well as identifying the gender of the speaker.

Method

K - means clustering is one of the most widely used clustering technique in machine learning. The algorithm is used to section datapoints into k groups or clusters based on their similarities by minimizing the sum of squared distance between the datapoints and their assigned cluster, hence reducing the variation within each group. The variation within a cluster is measured by the within sum of squares, which is given by (Ashour et al., 2018):

$$s^2_{j(g)} = \sum_{i=1}^{n_g} (x_{ij(g)} - \bar{x}_{j(g)})^2 \Big/ n_g - 1$$ where

N_g is the number of observations in the g^{th} cluster, and $x_{j(g)}$ is the mean of j^{th} variable in g^{th} cluster.

The algorithm is pretty simple to implement and only requires two parameters to work: the target number of clusters k and the dataset to partition. The algorithm undergoes iterative phases and for each phase, the steps include first randomly assigning each observation to one of the clusters and assign initial random centroid, computing the mean for each cluster and reassigning each observation to the nearest cluster by calculating the distance of each observation from the assigned cluster's mean value (Ashour et al., 2018). The iterative process is repeated until the sum of square errors can no longer be reduced in each cluster or the if specified, the maximum number of iterations has been reached.

Data Structure and Exploration

The vowel dataset (figure 1) analyzed in this paper has been taken from the UCI Repository of Machine Learning Database. It is a collection of ten different vowel sound of British English recorded from different speakers. The sound recognition was collected using a specific training set

(Deterding, 1989). The data contains 990 observations representing sound characteristics from a speaker for sound recognition. There are 14 total variables in the dataset, 10 of which are numerical independent variables (features 0-9) representing the sound wave, one categorical target variable (class) representing the vowel sound and the speaker name, sex, as well as the type of test.

The table in figure 3 shows that the vowel sounds are equally distributed. The summary of the data (figure 2) reveals that the variables have differing scales as they do not have the same range; however, each numerical data seems to have a small range, hence the data appears to be well spread over each range, eliminating the possibility of the presence of outliers in the data. Plus, the data does not contain missing values for any of the variables, which is beneficial since the k-means algorithm cannot handle missing values.

Data Preprocessing

Because k-means algorithm is a distance-based algorithm, it is not suitable for categorical variables. So, the class, train test, sex and speaker number variables were dropped for the analysis. The distribution for each feature variable can be observed in figure 4. The feature variables appear to be well distributed across their range, following normal like distribution. However, as mention above, the dataset is imbalanced, and the distributions will need to be rescale to the same range. This transformation is necessary to reduce the impact of large values on the algorithm (as it weights each dimension equally) and allow the comparison of a sole observation against the mean. Below is the summary of the transformed data, with variables having the same mean value.

```
> summary(newvowel)
    Feature0          Feature1          Feature2          Feature3
 Min.   :-2.30988   Min.   :-2.685135   Min.   :-2.78002   Min.   :-2.5347
 1st Qu.:-0.78713   1st Qu.:-0.706444   1st Qu.:-0.65732   1st Qu.:-0.7652
 Median : 0.06702   Median :-0.004479   Median :-0.09092   Median :-0.1080
 Mean   : 0.00000   Mean   : 0.000000   Mean   : 0.00000   Mean   : 0.0000
 3rd Qu.: 0.69189   3rd Qu.: 0.728543   3rd Qu.: 0.61665   3rd Qu.: 0.7646
 Max.   : 2.60388   Max.   : 2.716168   Max.   : 2.72319   Max.   : 2.4517
    Feature4          Feature5          Feature6          Feature7
 Min.   :-2.74050   Min.   :-2.4281   Min.   :-3.31792   Min.   :-2.84240
 1st Qu.:-0.69717   1st Qu.:-0.7191   1st Qu.:-0.65516   1st Qu.:-0.75406
 Median : 0.01002   Median :-0.1296   Median : 0.05708   Median :-0.01492
 Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.00000
 3rd Qu.: 0.71495   3rd Qu.: 0.6595   3rd Qu.: 0.65133   3rd Qu.: 0.75605
 Max.   : 3.21494   Max.   : 2.8098   Max.   : 3.04673   Max.   : 2.96955
    Feature8            Feature9
 Min.   :-2.2976367   Min.   :-2.6634
 1st Qu.:-0.7033519   1st Qu.:-0.7892
 Median : 0.0008344   Median :-0.1410
 Mean   : 0.0000000   Mean   : 0.0000
 3rd Qu.: 0.6958128   3rd Qu.: 0.7324
 Max.   : 2.8272260   Max.   : 2.4294
```

Figure 6: Data Summary After Transformation

Clustering Analysis in R and Interpretation

Before starting the analysis, it is important to set a seed for R's random number generator (using the set.seed() function), as the k means algorithm begins with a randomly selected k number of centroids. This will allow the results of the analysis to be reproducible. One difficulty with k – means algorithm is deciding on the number of clusters k. A large k value may improve the stability, but it might also overfit the model, leading to the model struggling to generalize the overall pattern. One general rule to select a k value is applying the following formula:

$k = \sqrt{2/n}$ where n is the total number of observations.

Experiment with k = 4

The algorithm was first run with the transformed data frame and k parameter set to 4. A plot animation was used to visualize the iterative processes, and the steps are displayed in figure

7. In step 1, the algorithm randomly chooses 4 centroids, then calculate the Euclidean distance between the observations and draw the clusters. This generates the two black, one red and one green clusters observed in step two of figure 7. Then the algorithm computes the mean of the clusters and repeat the steps until no more data point can be reassigned.

The result of the algorithm is displayed in figure 8. The algorithm converged after five iterations resulting in four clusters sizes of 193, 181, 211 and 405. Figure 8 shows the cluster means (centers) for the four groups across the 10 vowel features, as well as the cluster assignment for each observation in the dataset (clustering vector). Since the dataset has been standardized, a positive value a given cluster center indicates that its z-score is above the overall mean. For example, cluster 1 has the highest feature0 average among all clusters. Another important component of the results is the within sum of squares. This measures the variability of the observation within each cluster. In general, the total sum of squares within the clusters must be as small as possible. The within sum of squares for each of the 4 clusters was 1105.666, 1138.639, 1338.122, and 2490.789; and the total of within sum of squares was evaluated to 6073.216. This value measures the compactness of the clustering. Next, a cross tabulation was generated to compare the clusters against the sex variable from the original dataset. The result is shown below:

```
          1   2   3   4
Female   17 164  25 256
Male    176  17 186 149
```

Female speakers are dominant in cluster 2 and 4, while Males are dominant in cluster 1 and 3. Overall, the algorithm was able to identify the speaker's gender with about 45% accuracy.

Experiment with k = 11

A second experiment was performed with k = 11 to check how accurately the algorithm will recognize the eleven vowel phonemes in the dataset. The result of the algorithm is display in figure 9. The algorithm converged after six iterations the generated 11 clusters of sizes 99, 31, 106, 101, 129, 109, 49, 101, 126, 67, and 72. The total sum of squares within the clusters here is 4242.297 which is smaller than the one obtained with k = 4, thus the variability of the observations within the clusters has decreased by increasing the k value. The generated clusters were then evaluated against the actual vowel class. The result is displayed below:

	1	2	3	4	5	6	7	8	9	10	11
had	0	0	3	26	0	36	0	0	20	0	5
hAd	0	0	0	0	11	42	0	0	37	0	0
hed	6	0	0	10	15	30	7	4	18	0	0
hEd	35	0	0	0	12	1	0	0	35	7	0
hid	12	6	0	0	37	0	0	0	0	35	0
hId	24	0	0	0	31	0	0	0	15	20	0
hod	0	8	32	9	0	0	0	21	0	0	20
hOd	0	14	27	16	0	0	0	27	0	0	6
hud	16	0	0	3	9	0	24	7	0	5	26
hUd	6	3	14	10	12	0	18	15	0	0	12
hYd	0	0	30	27	2	0	0	27	1	0	3

cluster 6 is the dominant cluster for phoneme had, hAd and hed and cluster 5 is dominant for hid and hId. Plus, the algorithm correctly clustered approximately 35% of the vowel phonemes in the dataset.

Visualizing Both Experiment

A cluster plot was generated to compare both analyses. The dataset is multi-dimensional, having more than two variables. The plot of the dataset clusters in this state will be very noisy. A solution to this problem will be to apply a dimensionality reduction algorithm, like principal component analysis (PCA), that will pot the observations according to the principal components coordinates that explain the majority of the variance. The plots can be visualized below:

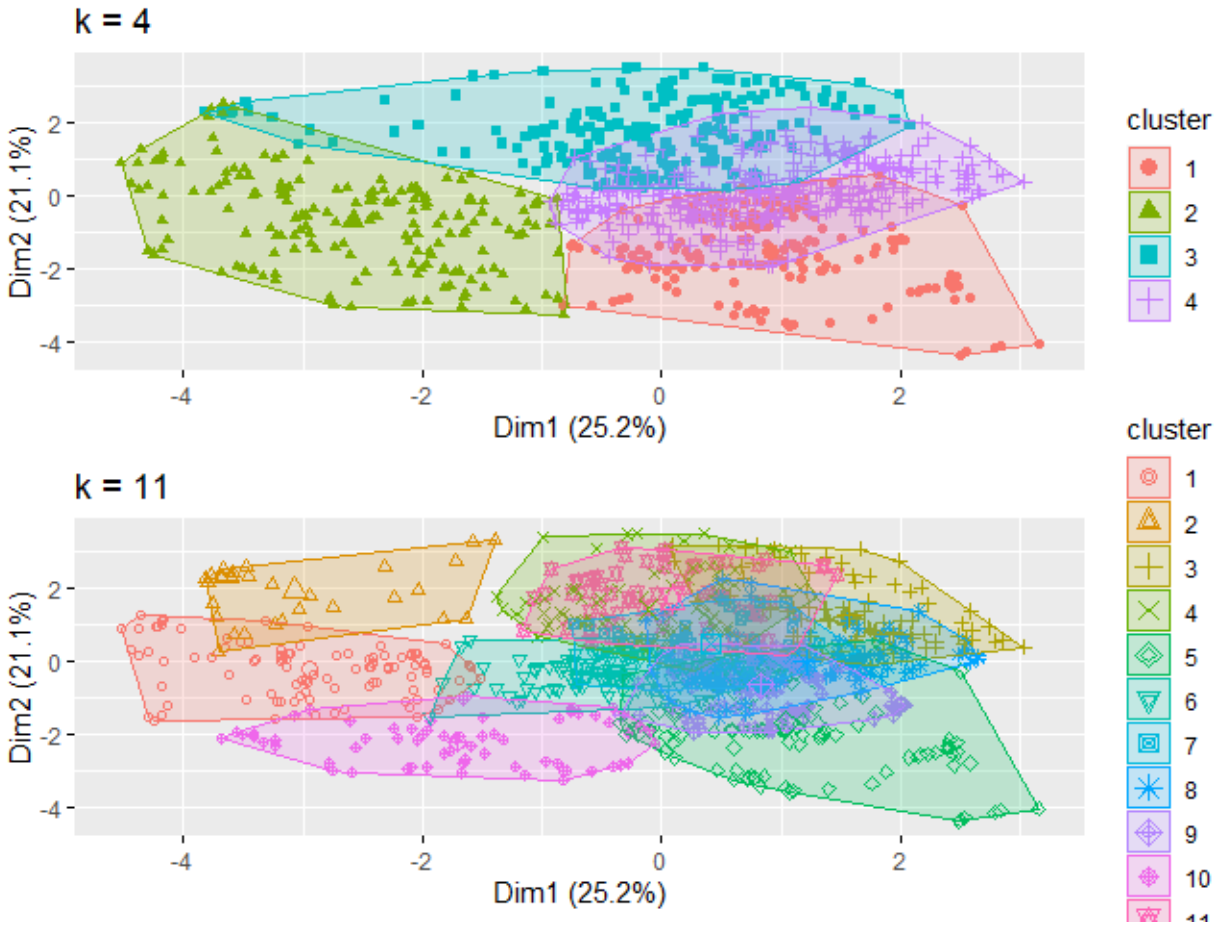


Figure 10: Cluster Plots for k= 4 and k = 11

The visualization reveals where the true delineations occur or not. The clusters in both plots are not fully separated and overlapped with each other, with 46. 3% of the point variability explained by each cluster. This may explain the low accuracy obtained in both experiments. Although the delineations can be observed with this visual assessment, the plot does not reveal the optimal number of clusters.

Determining the Optimal K Clusters

The optimal k value was determined using the elbow method. This technique uses the within-cluster homogeneity to measure the variability. Thus, the interest here is in the percentage

of variance explained by each cluster. The method works as follow: first perform clustering analysis with different k values; calculate the total within sum of squares (wss); plot the wss values against the k values. The optimal k is indicated by the point where the curve bend.

The Elbow method was computed using the fviz_nbclust function and the resulting plot is shown below:

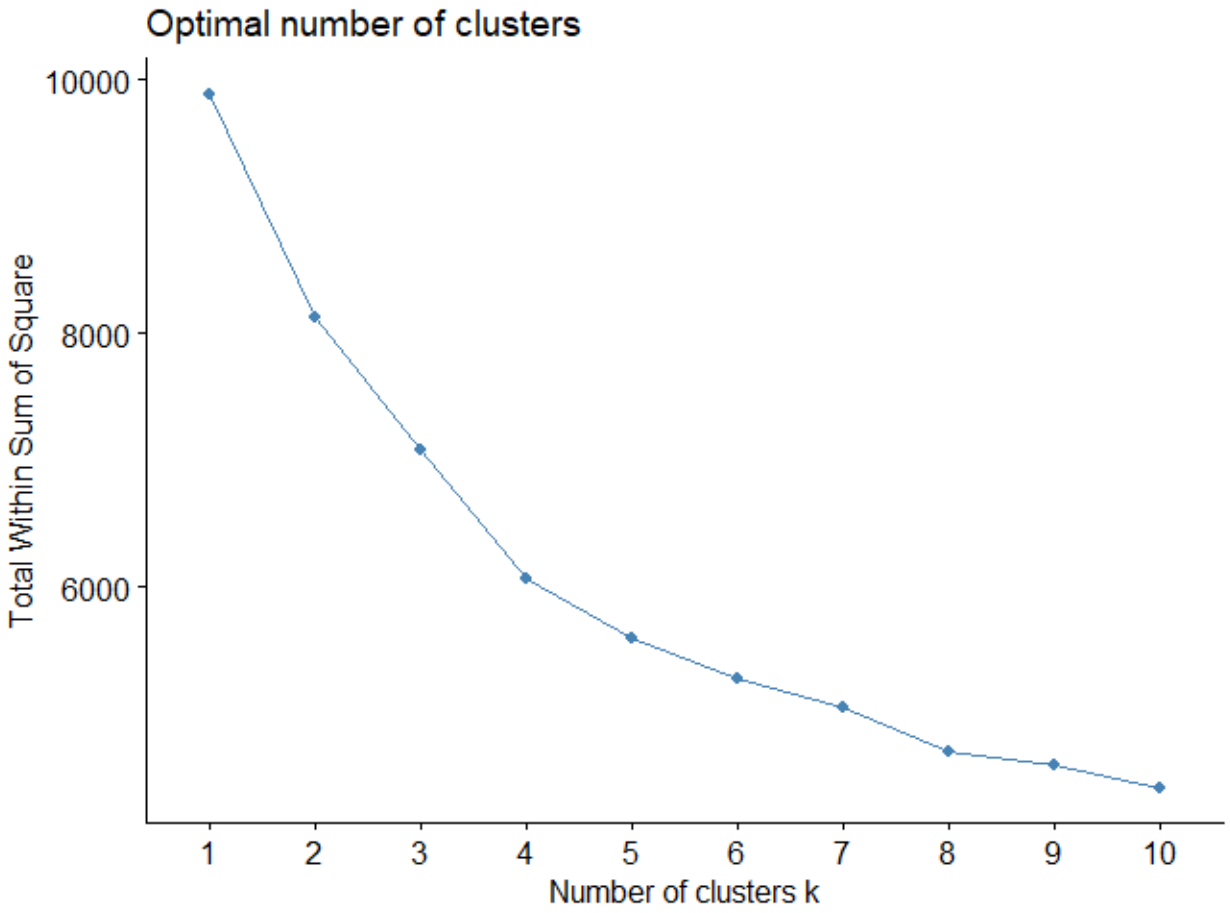


Figure 11: Total Within Cluster Sum of Squares Vs. Number of Clusters k

The graph suggest k = 5 to be the optimal number of clusters, as it is the k value where curve appears to bend. The cluster plot for the optimal k value can be visualized below

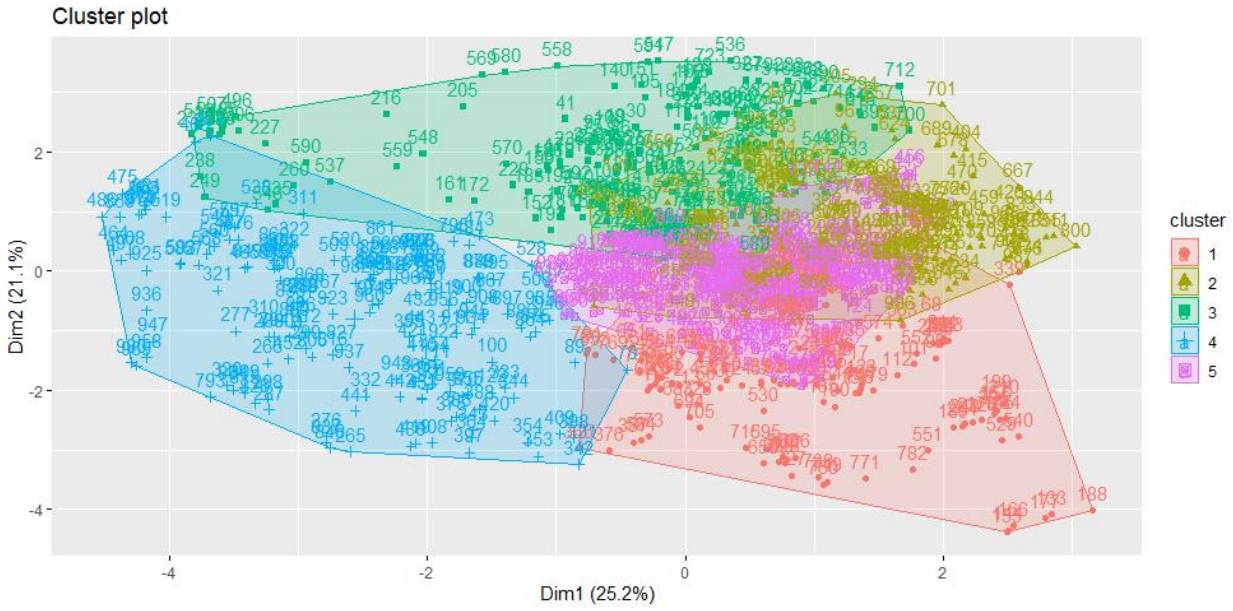


Figure 12: Cluster Plot for Optimal K = 5

The distance between the points in each cluster has significantly decreased, with the points closer to each other compare to the other two cluster plots (with k = 4 and 11). However, the clusters are still overlapping. The cross tabulation was also rebuilt to compare the optimal number of clusters with the actual vowel sounds and actual gender distribution in the data. The results are below:

	1	2	3	4	5
had	4	56	24	6	0
hAd	12	78	0	0	0
hed	19	35	10	14	12
hEd	27	21	0	42	0
hid	43	0	4	43	0
hId	51	0	0	39	0
hod	0	0	45	0	45
hOd	0	10	37	3	40
hud	12	0	22	18	38
hUd	12	0	25	6	47
hYd	4	52	27	0	7

Cross Tabulation the Vowel Class

	1	2	3	4	5
Female	17	147	22	157	119
Male	167	105	172	14	70

Cross Tabulation the Sex Variable

The algorithm was able to recognize 53% of the vowel sounds correctly, while the percentage for correct speaker's gender recognition was only 32%, lower than the one obtained with k = 4.

Conclusion

The results of the analysis show that the generated models were overall not able to accurately recognize all the eleven vowel phonemes or the speakers' gender. Thus, their application in real life will not be useful. With the determined optimal number of clusters (5), the algorithm correctly identified only 53% of the vowel sounds and 32% of the speaker's gender. Determining the optimal number of clusters slightly improve the model's performance for vowel recognition; however, it has significantly decreased the performance for speaker recognition.

It appears that the performance of the k-means algorithm is not solely dependent on the number of clusters. As seen in the analysis, the optimum k value can be used but still results in a bad performance, even with well prepared data. The problem could be with initialization of the algorithm. The algorithm starts by selecting a random number of data points as the initial set of centroids. It seems like the algorithm failed to optimize the centroid locations globally with the given dataset as it starts with a poor initialization. Thus, trying to improve the goodness of centroid initialization such that the initial centroids are as close as possible to the optimal cluster centers could be a great technique to optimize the model in this study.

References

Ashour A. et al., “Neutrosophic Set in Medical Image Analysis”, 2018. Retrieved from: https://sciencedirect.com/topics/computer-science/k-means-clustering

Daqrouq K. and Tutunji T., “Speaker Identification Using Vowels Features Through a Combined Method of Formants, Wavelets, and Neural Network Classifiers”, 2015. Retrieved from https://doi.org/10.1016/j.asoc.2014.11.016

Deterding D., “Speaker Normalization of Automatic Speech Recognition”, University of Cambridge, 1989.

Jennings N. et al., “Cognitive Systems – information Processing Meets Brain Science”. 2006

Sarma M. and Sarma K., “Segmentation of Assamese Phonemes Using SOM”. Conference Paper January 2012.

Appendix

```
> head(vowel)
  TrainTest SpeakerNumber  Sex Feature0 Feature1 Feature2 Feature3 Feature4 Feature5
1     Train         Andrew Male   -3.639    0.418   -0.670    1.779   -0.168    1.627
2     Train         Andrew Male   -3.327    0.496   -0.694    1.365   -0.265    1.933
3     Train         Andrew Male   -2.120    0.894   -1.576    0.147   -0.707    1.559
4     Train         Andrew Male   -2.287    1.809   -1.498    1.012   -1.053    1.060
5     Train         Andrew Male   -2.598    1.938   -0.846    1.062   -1.633    0.764
6     Train         Andrew Male   -2.852    1.914   -0.755    0.825   -1.588    0.855
  Feature6 Feature7 Feature8 Feature9 Class
1   -0.388    0.529   -0.874   -0.814   hid
2   -0.363    0.510   -0.621   -0.488   hId
3   -0.579    0.676   -0.809   -0.049   hEd
4   -0.567    0.235   -0.091   -0.795   hAd
5    0.394   -0.150    0.277   -0.396   hYd
6    0.217   -0.246    0.238   -0.365   had
> |
```

Figure 1: Vowel Dataset Preview

```
> summary(vowel)
  TrainTest         SpeakerNumber          Sex              Feature0
 Length:990         Length:990         Length:990         Min.   :-5.211
 Class :character   Class :character   Class :character   1st Qu.:-3.888
 Mode  :character   Mode  :character   Mode  :character   Median :-3.146
                                                          Mean   :-3.204
                                                          3rd Qu.:-2.603
                                                          Max.   :-0.941
    Feature1         Feature2          Feature3          Feature4          Feature5
 Min.   :-1.274   Min.   :-2.48700   Min.   :-1.4090   Min.   :-2.1270   Min.   :-0.8360
 1st Qu.: 1.052   1st Qu.:-0.97575   1st Qu.:-0.0655   1st Qu.:-0.7690   1st Qu.: 0.1960
 Median : 1.877   Median :-0.57250   Median : 0.4335   Median :-0.2990   Median : 0.5520
 Mean   : 1.882   Mean   :-0.50777   Mean   : 0.5155   Mean   :-0.3057   Mean   : 0.6302
 3rd Qu.: 2.738   3rd Qu.:-0.06875   3rd Qu.: 1.0960   3rd Qu.: 0.1695   3rd Qu.: 1.0285
 Max.   : 5.074   Max.   : 1.43100   Max.   : 2.3770   Max.   : 1.8310   Max.   : 2.3270
    Feature6           Feature7          Feature8          Feature9
 Min.   :-1.537000   Min.   :-1.29300   Min.   :-1.61300   Min.   :-1.68000
 1st Qu.:-0.307000   1st Qu.:-0.09575   1st Qu.:-0.70400   1st Qu.:-0.54800
 Median : 0.022000   Median : 0.32800   Median :-0.30250   Median :-0.15650
 Mean   :-0.004365   Mean   : 0.33655   Mean   :-0.30298   Mean   :-0.07134
 3rd Qu.: 0.296500   3rd Qu.: 0.77000   3rd Qu.: 0.09375   3rd Qu.: 0.37100
 Max.   : 1.403000   Max.   : 2.03900   Max.   : 1.30900   Max.   : 1.39600
    Class
 Length:990
 Class :character
 Mode  :character
```

Figure 2: Data Summary

```
> table(vowel$Class)

had hAd hed hEd hid hId hod hOd hud hUd hYd
 90  90  90  90  90  90  90  90  90  90  90
> |
```

Figure 3: Vowel class distribution

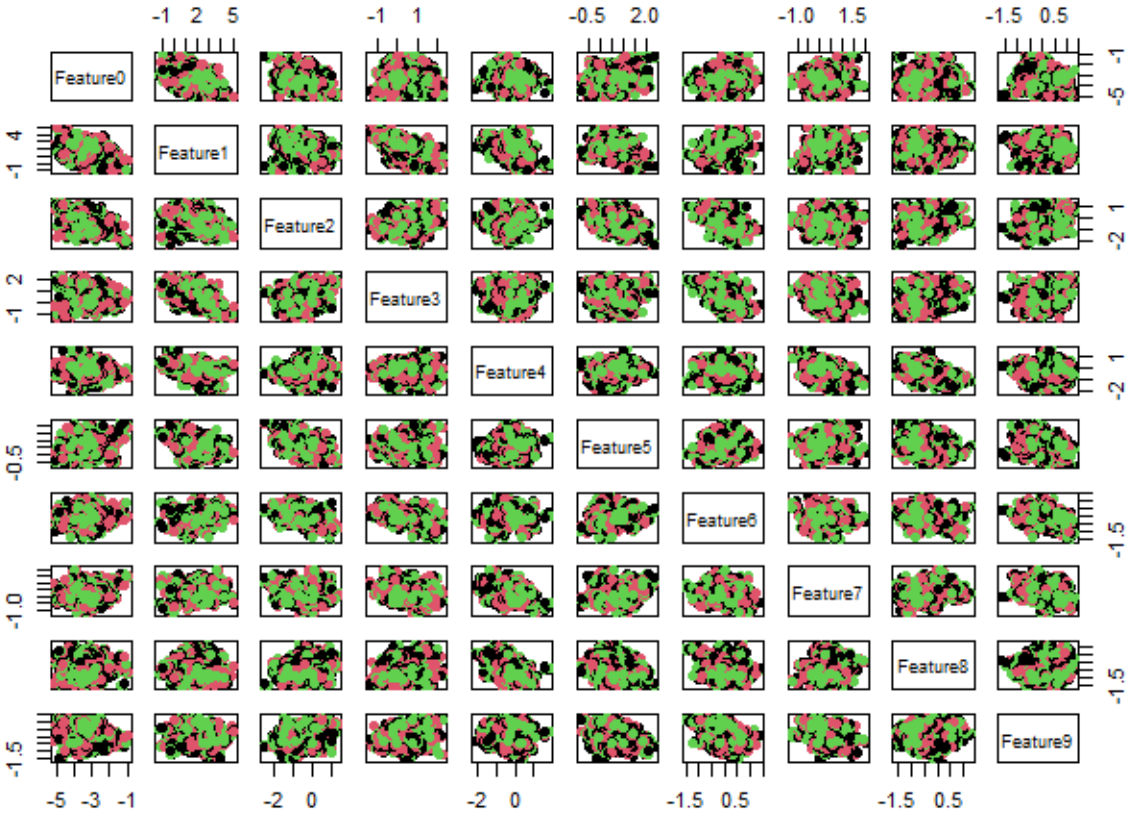
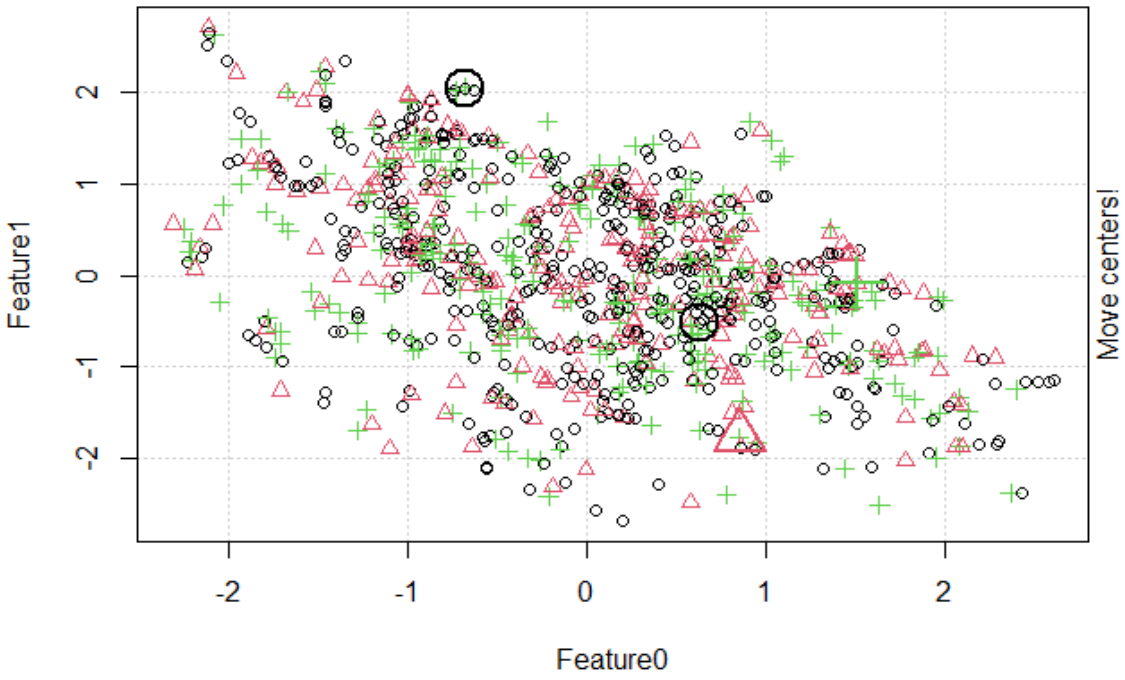


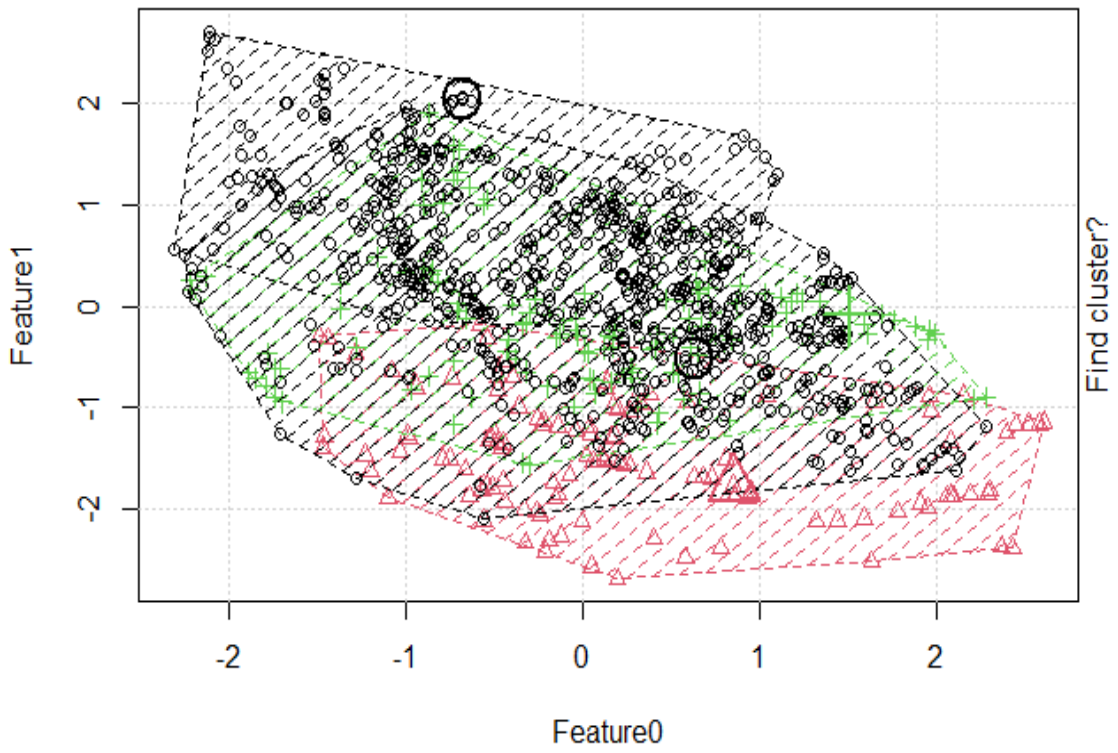
Figure 4: Vowel Features Plot

```
> apply(vowel, 2, function(vowel) sum(is.na(vowel)))
    TrainTest SpeakerNumber           Sex      Feature0      Feature1      Feature2      Feature3
            0             0             0             0             0             0             0
     Feature4      Feature5      Feature6      Feature7      Feature8      Feature9         Class
            0             0             0             0             0             0             0
```

Figure 5: Checking Missing Values in the Dataset



Step 1



Step 2

Figure 7: K-Means Algorithm Steps per Iteration

```
K-means clustering with 4 clusters of sizes 193, 181, 211, 405

Cluster means:
     Feature0   Feature1   Feature2   Feature3    Feature4   Feature5
1  0.74790120 -0.9680898 -0.8570254  0.4714346  0.09206420  1.2183727
2 -0.44934501 -0.6463731  0.7463631  1.0191980  1.00938552 -0.4663031
3 -0.24278290  0.3840590  0.7695085  0.1248977 -1.12925220 -0.4283782
4 -0.02910195  0.5501195 -0.3260546 -0.7452225  0.09334579 -0.1490303
     Feature6   Feature7   Feature8   Feature9
1 -0.01131354  0.6989178 -0.1042671 -0.7198085
2 -0.99109234 -1.2115625 -0.4311794  1.1353903
3 -0.25326696  0.6778014  1.1886202  0.1560457
4  0.58027297 -0.1447270 -0.3768687 -0.2456994

Clustering vector:
  [1] 1 1 1 1 3 3 3 3 1 1 1 1 1 1 1 3 3 3 3 1 1 1 1 1 1 1 1 4 3 3 1 1 1 1 1 1
 [38] 1 4 3 3 1 1 1 1 1 1 1 1 4 3 3 1 1 1 1 1 1 1 1 4 3 3 1 1 1 4 1 1 4 3 3 4
 [75] 3 3 4 4 1 1 4 3 3 3 3 3 3 4 2 1 1 4 3 3 3 4 3 3 4 2 1 1 4 4 3 3 3 3 3 4
```

```
[112] 1 1 4 4 3 3 3 3 3 4 2 1 1 4 4 3 3 3 4 4 4 1 1 1 1 3 3 3 3 3 3 1 1 1 1 1
3
[149] 3 3 3 3 3 1 1 1 1 1 3 3 3 3 3 3 1 1 1 1 1 3 3 3 3 3 3 1 1 1 1 1 3 1 3 3
3
[186] 3 1 1 1 1 1 3 1 3 3 3 3 1 1 1 4 4 3 3 3 3 3 3 3 1 1 1 4 3 3 3 3 3 3 3 1
1
[223] 1 4 3 3 3 3 3 3 3 3 1 1 1 4 3 3 2 3 3 3 3 1 1 1 4 3 3 2 3 3 3 3 1 1 1 1 3
3
[260] 3 3 3 3 3 2 2 2 4 4 4 3 3 4 4 4 2 2 2 4 4 4 3 3 4 4 4 2 2 2 4 4 4 3 3 4
4
[297] 4 2 2 2 4 4 4 3 3 4 4 4 2 2 2 4 4 4 4 3 4 4 4 2 2 2 4 4 4 4 3 4 4 4 2 2
2
[334] 4 4 4 4 3 1 4 1 2 2 2 4 4 4 4 3 1 4 1 2 2 2 4 4 4 4 3 1 4 1 2 1 2 4 4 4
4
[371] 3 1 4 1 2 1 2 4 4 1 4 3 1 4 1 2 1 2 4 4 1 4 3 1 4 1 2 2 2 4 4 4 3 4 4 4
4
[408] 2 2 2 4 4 4 3 4 4 4 4 2 2 2 4 4 4 3 4 4 4 4 2 2 2 4 4 4 3 4 4 4 4 2 2 2
4
[445] 4 4 4 4 4 4 4 2 2 2 4 4 2 4 4 4 4 4 4 2 2 2 4 4 4 4 4 4 4 2 2 2 2 4 4 4
4
[482] 4 4 2 2 2 2 4 4 4 4 4 4 4 2 3 2 2 4 4 4 4 4 4 4 2 2 2 2 4 4 4 4 4 4 4 2
2
[519] 2 2 4 4 4 4 4 4 4 4 2 1 1 1 4 3 3 2 3 3 2 3 1 1 1 4 3 3 2 3 3 2 3 1 1 1 4
3
[556] 3 2 3 3 2 3 1 1 1 1 3 3 2 3 3 2 3 1 1 1 1 3 3 2 3 3 2 4 1 1 1 1 3 3 3 3
3
[593] 2 4 1 1 1 4 4 4 4 3 4 3 4 1 1 1 4 4 4 4 3 4 3 4 1 1 1 4 4 4 4 3 4 3 4 1
1
[630] 1 4 4 4 4 3 4 3 4 1 1 1 4 4 4 4 3 4 3 4 1 4 1 4 4 4 4 3 4 3 4 1 1 4 4 4
4
[667] 4 3 4 3 4 1 1 4 4 4 4 3 3 4 4 4 1 1 4 4 4 4 3 3 4 4 4 1 1 4 4 4 4 3 3 4
3
[704] 4 1 1 4 4 4 4 3 3 3 3 4 1 1 4 4 4 4 3 3 3 3 4 1 1 4 4 3 3 4 4 4 1 4 1 1
4
[741] 4 3 3 4 4 4 1 4 1 1 4 4 3 3 4 4 4 1 4 1 1 4 4 3 4 4 4 4 1 4 1 1 4 4 3 4
4
[778] 4 4 1 4 1 1 4 4 3 4 4 4 4 1 4 2 2 2 4 4 4 4 4 4 4 4 2 2 2 4 4 4 4 4 4 4
4
[815] 2 2 2 4 4 4 4 4 4 3 4 2 2 2 4 4 4 4 4 4 3 4 2 2 2 4 4 4 4 4 4 3 4 2 2 2
4
[852] 4 4 4 4 4 4 4 2 2 2 4 4 2 4 4 2 2 2 2 2 2 4 4 2 4 4 2 2 2 2 2 2 4 4 2 4
4
[889] 2 2 2 2 2 2 3 4 2 4 4 2 2 2 2 2 2 4 4 2 4 4 2 2 2 2 2 2 4 4 2 4 4 2 2 2
2
[926] 2 2 2 4 4 4 4 4 2 2 2 2 2 2 4 4 4 4 4 2 2 2 2 2 4 4 4 4 4 4 2 4 2 2 2 4
4
[963] 4 4 4 4 2 4 2 2 2 4 4 4 4 4 4 2 4 2 2 2 2 4 4 4 4 4 2 4

Within cluster sum of squares by cluster:
[1] 1105.666 1138.639 1338.122 2490.789
 (between_SS / total_SS =  38.6 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss
"
[6] "betweenss"    "size"         "iter"         "ifault"
```

Figure 8: K-Means Algorithm Results for k = 4

```
K-means clustering with 11 clusters of sizes 99, 31, 106, 101, 129, 109, 49,
101, 126, 67, 72

Cluster means:
     Feature0     Feature1    Feature2    Feature3    Feature4    Feature5
1  -0.5424369 -0.389358394  0.86127585  1.14881991  0.84689675 -1.01154108
2  -0.4587486 -0.247401142  1.89610829  1.34991270 -0.90787335 -0.97759478
3  -0.2237317  1.015774928  0.04832061 -1.08289434 -0.71571105 -0.23524187
4   0.4015917  0.060722179  0.82945926 -0.08195518 -0.95515455 -0.49664212
5   0.6799015 -0.949893971 -1.04784757  0.67387726 -0.01390553  1.53422894
6   0.3717679  0.003362592 -0.22134769 -0.34031903  0.17176136 -0.58232316
7  -1.2520726  0.594957049  0.74376866 -0.64745147  0.78912385 -0.30105013
8  -0.4872090  1.335522921 -1.00914360 -1.04336380  0.09098593  0.45630209
9   1.0059995 -0.510109662 -0.49190995 -0.47008759  0.32071760 -0.11226248
10 -0.3941867 -1.376171279  0.54876891  1.06856901  1.57816437  0.88280240
11 -0.9297012  0.653016895  0.23678020  0.58875271 -1.30965245 -0.07271604
     Feature6   Feature7   Feature8    Feature9
1  -1.2424876 -1.2425397 -0.3677391  1.32073209
2  -2.1327701  0.1434740  1.4771278  1.50145248
3   1.1544067  0.3776276  0.3322201 -0.61503451
4  -0.5185805  0.9459220  1.0441117  0.09542683
5  -0.1145012  0.9222478 -0.2056418 -0.76199553
6   0.1907595 -0.7163591 -0.6102349  0.85591997
7   0.2728088  0.6133792 -0.7510648 -1.04582082
8   0.5905703 -0.3425493 -0.8286594  0.09677104
9   0.2193872  0.1102411  0.1956831 -0.71943066
10 -0.2358249 -1.1590624 -0.9376400  0.26445578
11  0.3923858  0.1447080  1.4118135 -0.03248247

Clustering vector:
  [1]  5  5  5  5 11 11 11 11  5  5  5  5  5  5  9 11 11 11 11  5  5  5  5  5
5
 [26]  5 11 11 11 11  5  5  5  5  5  5  5  5 11 11 11  5  5  5  5  5  5  5  9
3
 [51] 11 11  5  5  5  5  5  5  5  5 11 11 11  5  5  5 10  9  9  9  4  4  3  3
11
 [76] 11  9 10  9  9  9  4  4  4  3 11 11  6 10  9  9  9  4  4  4  3 11 11  6
10
[101]  9  9  9  3  4  4  3 11 11  9 10  5  9  9  3  4  4  3 11 11  9 10  5  9
9
[126]  3  4  3  3  3  8  9  5  5  5  5  4  4  4  4 11 11  5  5  5  5  5  4  4
4
[151] 11 11 11  5  5  5  5  5  4  4  2 11 11 11  5  5  5  5  5  4  4  2 11 11
11
[176]  5  5  5  5  5  4  4  4 11 11 11  5  5  5  5  5  4  4  4 11 11 11  5  5
9
[201]  9  9  4  4  2  2  4  4  4  5  9  9  9  4  4  2  2  4  4  4  5  9  9  9
4
[226]  4  2  2  4  4  4  5  9  9  9  4  4  2  2  4 11  4  5  9  9  9  4  4  2
2
```

[251] 4 11 4 5 9 9 9 4 4 2 2 4 11 4 10 1 1 6 3 6 4 3 7 7
9
[276] 10 1 1 6 3 6 4 3 7 7 9 10 1 1 6 3 6 4 3 7 7 6 10 10
1
[301] 6 3 6 4 3 7 7 6 10 1 1 6 3 6 3 3 7 7 6 10 1 1 6 3
6
[326] 3 3 7 7 6 10 10 10 6 3 9 3 11 5 7 5 10 10 10 6 3 9 3 11
5
[351] 7 5 10 10 10 6 3 9 3 11 5 7 5 10 10 10 6 3 9 3 11 5 7 9
10
[376] 10 10 6 3 9 3 11 5 7 9 10 10 10 6 3 9 3 11 5 7 9 10 10 1
6
[401] 3 6 4 3 7 7 6 10 10 1 6 3 6 4 3 7 7 6 10 10 1 6 3 6
4
[426] 3 7 7 6 10 10 1 6 3 6 4 8 7 7 6 10 10 1 6 3 6 3 8 7
7
[451] 6 10 10 6 6 3 6 3 3 7 7 6 2 1 1 6 8 6 8 8 8 8 6 2
1
[476] 1 6 3 6 8 8 8 8 6 2 1 1 6 3 6 8 8 8 8 6 2 1 1 6
3
[501] 6 8 8 8 8 6 2 1 1 6 8 6 8 8 8 8 6 2 1 1 6 8 6 8
8
[526] 8 8 6 5 5 9 9 4 4 2 4 2 1 4 5 5 9 9 4 4 2 4 2 1
4
[551] 5 5 10 9 4 4 2 4 2 1 4 5 5 9 9 4 4 2 2 4 1 4 10 5
9
[576] 9 4 4 2 2 4 1 9 10 5 9 9 4 4 2 4 4 1 9 5 5 9 9 8
9
[601] 3 11 3 11 7 5 5 9 9 8 9 3 11 8 11 7 5 5 9 9 8 9 8 3
8
[626] 11 8 5 5 9 9 8 9 8 3 8 11 8 5 10 9 9 8 9 8 11 8 11 8
5
[651] 10 9 9 8 9 8 3 8 11 8 10 5 9 9 3 9 3 4 3 11 7 10 5 9
9
[676] 3 9 3 4 3 11 7 10 5 9 9 3 9 3 4 3 11 7 5 5 9 9 3 9
3
[701] 3 3 11 7 5 5 9 9 3 3 3 3 4 11 7 5 5 9 9 3 3 3 4 11
11
[726] 9 5 5 9 9 4 4 3 8 8 5 9 5 9 9 9 4 4 3 8 8 5 9 5
9
[751] 9 9 4 9 3 3 3 5 9 5 9 9 9 4 9 3 8 3 10 9 5 9 9 9
4
[776] 9 3 8 3 10 9 5 9 9 9 4 9 3 3 3 10 9 10 1 1 6 8 6 8
3
[801] 8 7 6 10 1 1 6 8 6 8 3 3 7 6 10 1 1 6 8 6 8 3 3 7
6
[826] 10 1 1 6 8 6 8 3 3 7 6 10 1 1 6 8 6 8 3 3 7 6 10 1
1
[851] 6 8 6 8 3 8 7 6 1 1 1 6 8 6 8 8 1 1 1 1 1 1 6 8
6
[876] 8 8 1 1 1 1 1 1 6 8 6 8 8 1 1 1 1 1 1 6 8 6 8 8
1
[901] 1 1 1 1 1 6 8 6 8 8 1 1 1 1 1 1 6 8 6 8 8 1 1 1
1
[926] 1 1 6 8 6 8 8 7 10 6 1 10 1 6 8 6 8 3 7 10 6 1 10 1
6

```
[951]  8  6  3  3  7  1  6  1 10  1  6  8  6  8  3  7  1  6  1 10  1  6  8  6
8
[976]  8  7  1  6  1 10  1  6  8  6  8  8  7  1  6

Within cluster sum of squares by cluster:
 [1] 396.7937 120.8050 443.2046 446.1042 683.5489 373.0717 191.0185 490.8773
 [9] 424.6406 360.3789 311.8533
 (between_SS / total_SS =  57.1 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss
"
[6] "betweenss"    "size"         "iter"         "ifault"
```

Figure 9: K-Means Algorithm Results for k = 11