Free Classification: Clustering Project

Clustering

Contents

Data

We have data in wide format. Each row is a talker type and each col is a participant.

```
df <- read_csv("data/class_wide_1.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   speaker = col_character(),
##   `54` = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

Clustering

Introduction

In this task, individuals heard spoken speech tokens and freely classified them into groups. Using hierarchical clustering we aimed to see what clusters or groups appear as a result of the free classification task.

45 speech samples were selected from The Speech Accent Archive. The talkers included three American English regional dialects, three international English dialects, and nine nonnative accents. The nonnative accents were split into three accents from East Asia, three accents from South Asia, and three accents from Southeast Asia. The American English dialects included the New England dialect, the Southern dialect, and the Midland dialect. The international English dialects included British English, Australian English, and Africaans. The native languages of the nonnative-accented talkers were Mandarin, Korean, and Japanese from East Asia, Bengali, Gujarati, and Urdu from South Asia, and Indonesian, Tagalog, and Thai from Southeast Asia.

```
library(here)
library(tidyverse)  # data manipulation
```

```
library(cluster)    # clustering algorithms
library(factoextra) # clustering visualization
library(dendextend) # for comparing two dendrograms
```

Data Preparation

1. Rows are observations (individuals) and columns are variables
2. Any missing value in the data must be removed or estimated.
3. The data must be standardized (i.e., scaled) to make variables comparable (I am not doing this here).

Read in the data

```
clust_data <- read_csv(here("data", "class_wide_1.csv")) # read in data
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   speaker = col_character(),
##   `54` = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
clust_data <- select(clust_data, -X1, -`54`) # remove extra col sub 54 has weird formatting

clust_data <- as.data.frame(clust_data) # turn into df

rownames(clust_data) <- clust_data$speaker # make row names speaker

clust_data <- select(clust_data,-speaker) # remove extra col sub 54 has weird formatting

head(clust_data)# show first couple rows
```

```
##             8 7 1 10 11 12 14 15 16 17 18 19 2 20 23 25 26 27 28 29  3 30 31 32 33
## bengali_9   1 5 5  1 11  2  1  8  4  2  2  1 9  7  4  5  1  1  1 11  5  1  7  7  9
## bengali_13  6 5 5  7 14  4  2  7  4  2  6  3 9  1  4  5  4  2  3 11 12  1  8 11  9
## bengali_16  1 5 5  7  7  4  3  6  2  8  3  3 3  1  3  4  6  1  3 10  2  1  7  8  9
## gujarati_5  4 5 5  1 14  4  1  7  4  9  9  1 9  4  3  5  4  2  4  8  9  1  7  9  9
## gujarati_13 1 5 5  1 15  4  2  8  4  2  6  1 9  4  5  5  6  2  4  8  5  1  7  9  9
## gujarati_14 5 5 5  1  7  4  1  8  4  9  9  3 9  5  7  5  4  4  6  1  5  1  6  9  9
##             34 35 36 38 4 40 41 42 43 44 45 46 47 48 49  5 50 51 52 53 55 56 58 59
## bengali_9    8 10  8  1 3  1 10  1  1 12  1  5  1  5  8  1  3  7  1  8  9  1  1  5
## bengali_13   8 10 11  1 4 12  1  1  8 11  1  5  4  1  8  5  4  7  3  8  9  1  8 11
## bengali_16   8 10 11  6 3  8  8  1  8 11  1  1  2  5  7  5  4  7  3  8  9  8  1  6
## gujarati_5   6 10  8  1 2  8  7  1  8  8  3 11  2  2  8 10  3  7 11  8  9  7  3 11
## gujarati_13  8 10  2  1 2 12  1  1  8 11  1 11  6  5  8  2  4  7  1  8  9  7 10  5
## gujarati_14  6 10  9  1 4 13  2  1  8 11  6  3  4  5  8  3  3  7 15  8  8  8  6  5
##             6 78 87 90 91 96 105 110 111 115 121 123 125 132 133 135 148 151 152
## bengali_9   9  1 11  1 11  1   6  11   1   2   1   9   1   3   1   1   1   5   1
## bengali_13  7  1 11  1 11  1   6  11   7   8   6   9   1   6   2   3   6   9   1
## bengali_16  7  5 11  2  6  1   6  10   6   8   6  10  10   4   2   3   1   7   4
## gujarati_5  7  1 11  2 10  1   1  11   6   8   6   9  10   4   3   2   3   5   3
## gujarati_13 7  1 11  2 11  1   6  11   2   8   6   9  10   4   2   1   1   5   1
```

```
## gujarati_14 6  2 11  8 11  1   6  11   6   9   6   9   8   4   2   1   2   5   3
##             153 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169
## bengali_9     8   4   8   6   5   1   5   1   7   4   1   5   1   7   1   2
## bengali_13    8   4  10   6   5   2   1  14   7   6   4   5   1   4   2   4
## bengali_16    8  11   8   6   3   2   3  14   7   7   1   5   1   7   4   4
## gujarati_5    2   5   9   6   3   1   3  14   7   5   6   6   1   4   2   5
## gujarati_13   2   5   9   6   3   1   4  14   7   5   6   5   1   7   3   6
## gujarati_14   8   5   9   6   3   8   4   5   3   7   6   7   6   7   5   1
```

Agglomerative Hierarchical Clustering

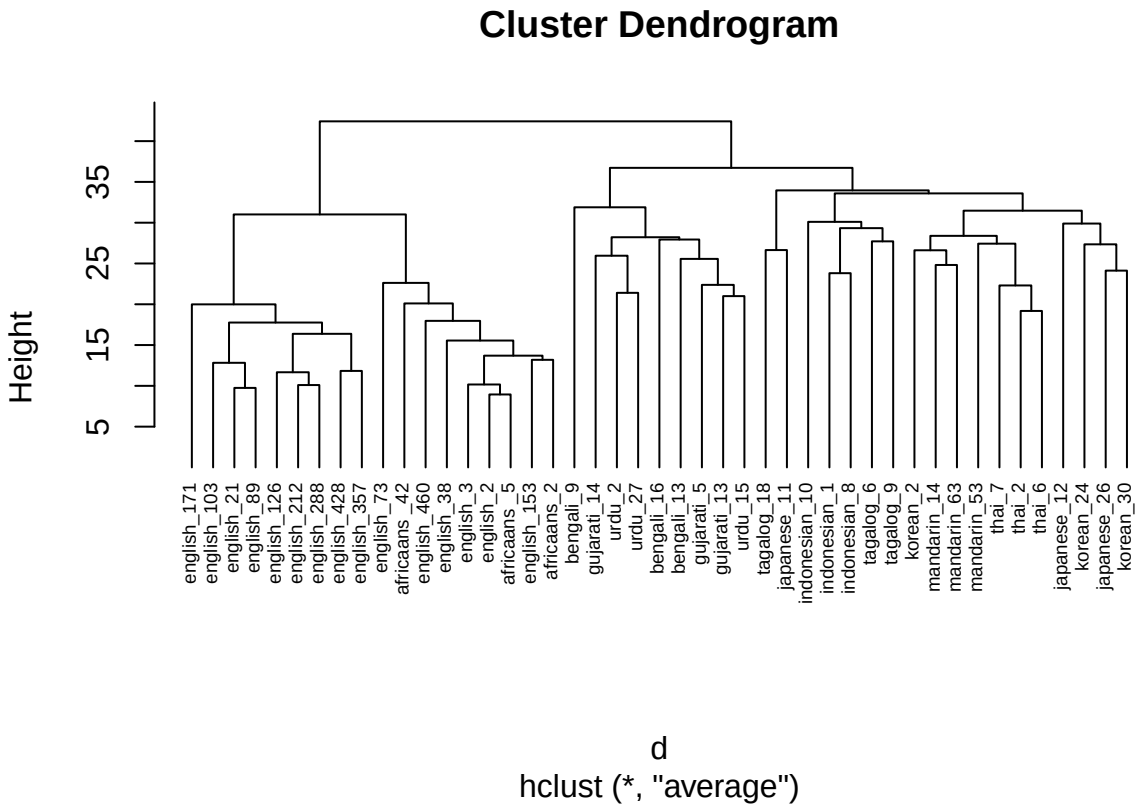
I am going to cluster the data using average link clustering. Average link clustering computes all pairwise dissimilarities between the elements, and considers the average of these dissimilarities as the distance between clusters.

First, we calculate the dissimilarity matrix using euclidean distance.

Second, we comput the clustering with average link.

Third, we plot the cluster solution

```r
# Dissimilarity matrix
d <- dist(clust_data, method = "euclidean")

# Hierarchical clustering using Average Linkage
hc1 <- hclust(d, method = "average" )

# Plot the obtained dendrogram
plot(hc1, cex = 0.6, hang = -1)
```



In the dendrogram displayed above, each leaf corresponds to one observation. As we move up the tree, observations that are similar to each other are combined into branches, which are themselves fused at a higher height.

The height of the fusion, provided on the vertical axis, indicates the (dis)similarity between two observations. The higher the height of the fusion, the less similar the observations are. Note that, conclusions about the proximity of two observations can be drawn only based on the height where branches containing those two observations first are fused. We cannot use the proximity of two observations along the horizontal axis as a criteria of their similarity.

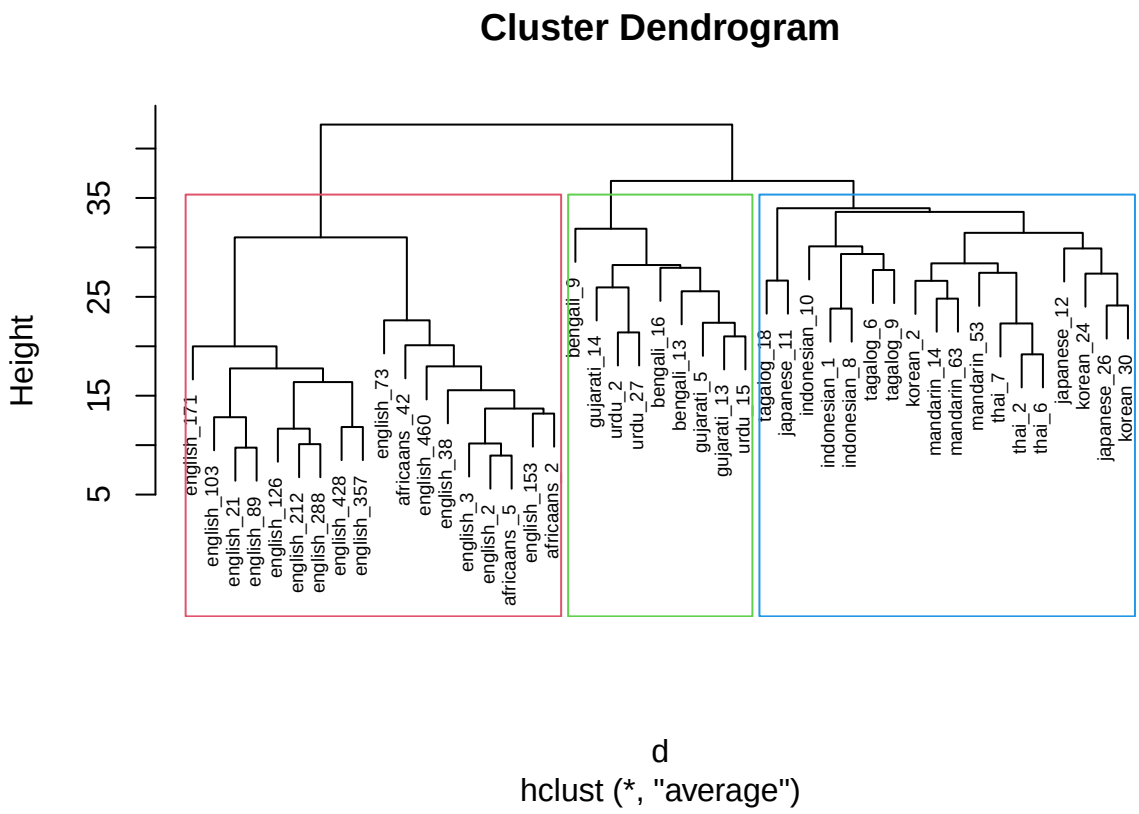
The height of the cut to the dendrogram controls the number of clusters obtained. It plays the same role as the k in k-means clustering. In order to identify sub-groups (i.e. clusters), we can cut the dendrogram with cutree:

```
# Ward's method
hc5 <- hclust(d, method = "average" )

# Cut tree into 3 groups
sub_grp <- cutree(hc5, k = 3)

# Number of members in each cluster
table(sub_grp)
```

```
## sub_grp
##  1  2  3
##  9 18 18
```

Visualize clusters on dendrogram

```
plot(hc5, cex = 0.6)
rect.hclust(hc5, k = 3, border = 2:5)
```



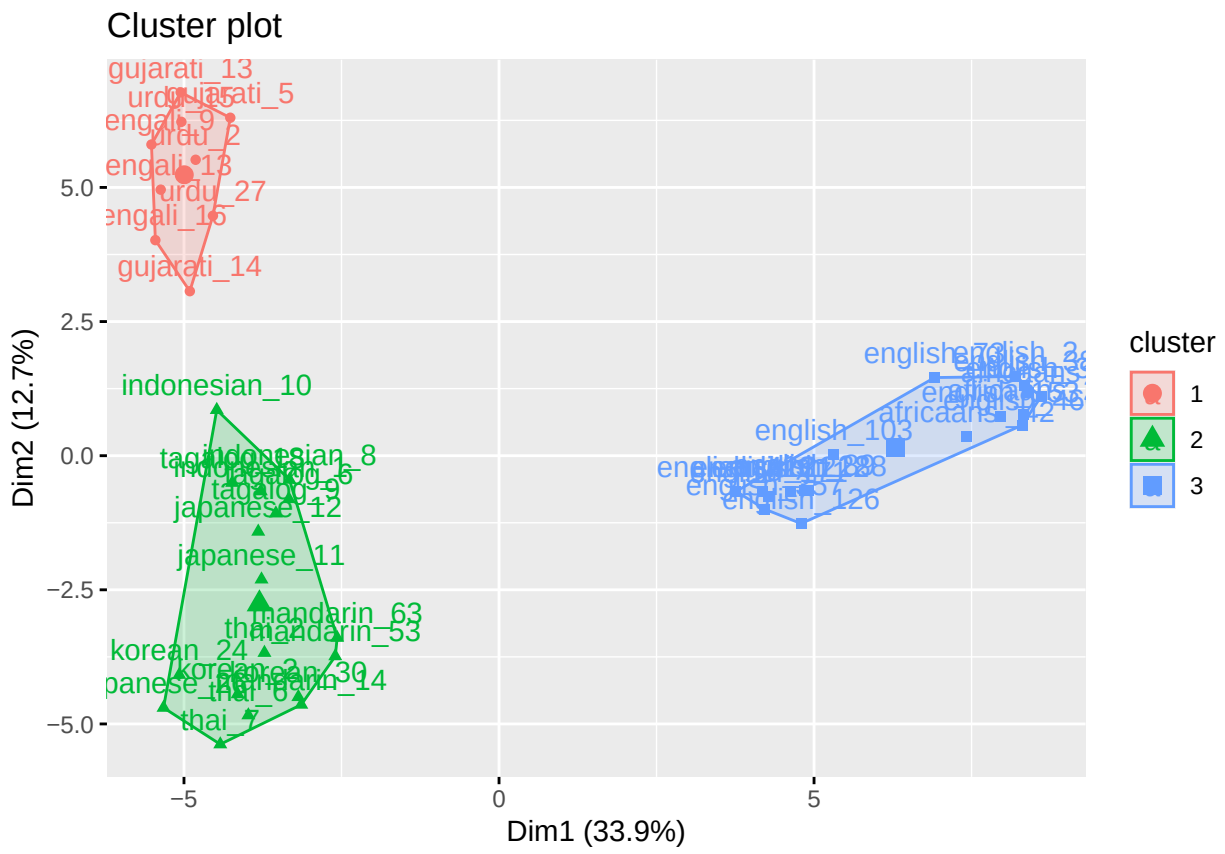
From this, we glean that 3 clusters appear to be adequate. Generally participants grouped speakers into 3 clusters/groups:

- English/African: Clust 1
- Indo/European: Clust 2
- Asian: Clust 3

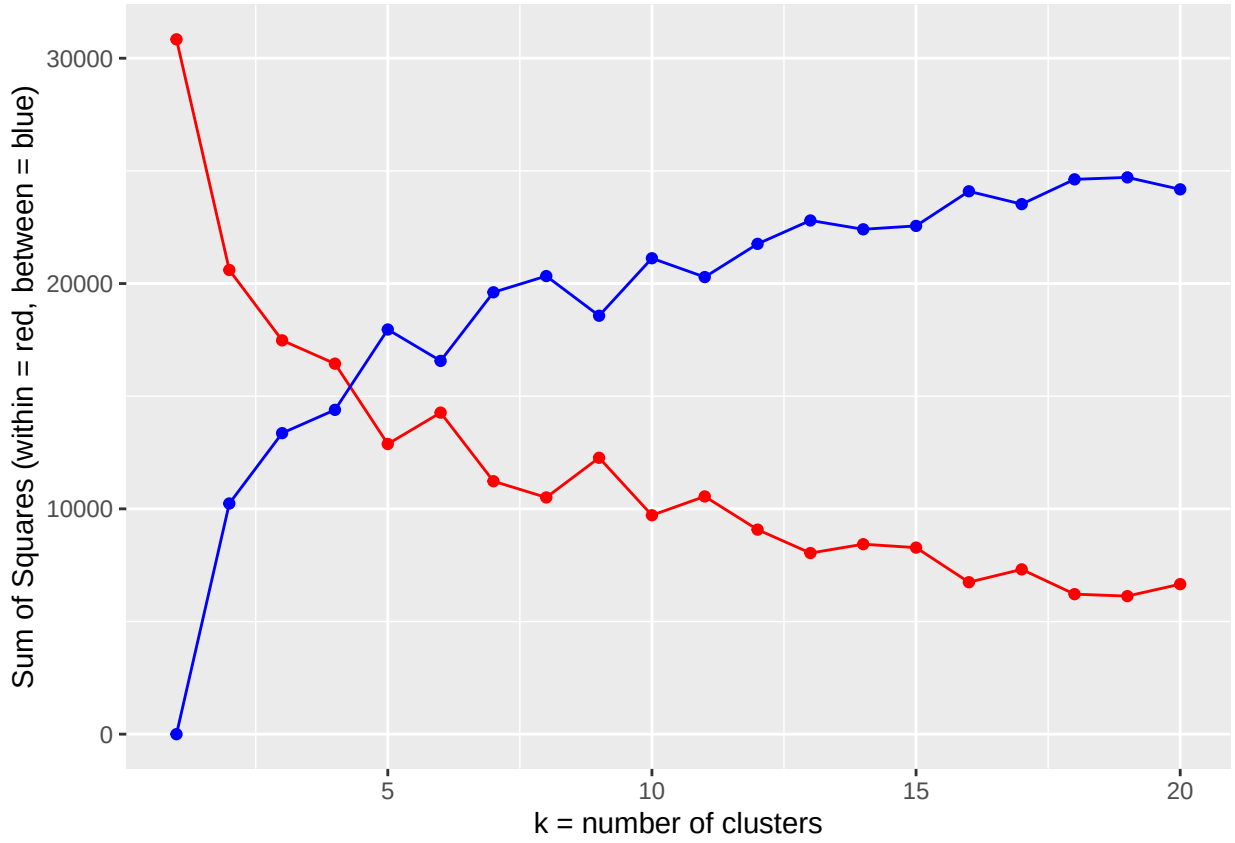
```
clust_data <- clust_data %>%
  mutate(cluster = sub_grp)
```

Visualize Clusters

```
fviz_cluster(list(data = clust_data, cluster = sub_grp))
```




```
#making a empty dataframe
criteria <- data.frame()
#setting range of k
nk <- 1:20
#loop for range of clusters
for (k in nk) {
model <- kmeans(clust_data, k)
criteria <- rbind(criteria,c(k,model$tot.withinss,model$betweenss,model$totss))
}
#renaming columns
names(criteria) <- c("k","tot.withinss","betweenss","totalss")

#scree plot
ggplot(criteria, aes(x=k)) +
  geom_point(aes(y=tot.withinss),color="red") +
  geom_line(aes(y=tot.withinss),color="red") +
  geom_point(aes(y=betweenss),color="blue") +
  geom_line(aes(y=betweenss),color="blue") +
  xlab("k = number of clusters") + ylab("Sum of Squares (within = red, between = blue)")
```



Conclusion

I ran a hierarchical clustering analysis using the average link method to classify talkers in a free classification task. This analysis determined that 3 clusters were adequate to explain the free classification task. A common method for choosing the optimal number of clusters is the "elbow method," which simply means picking the point on the plot where increasing the value of k results in only marginal gains. Here we have determined that 3 clusters were sufficient to explain the data.

Full Code

The full script of executive code contained in this document is reproduced here.

```
df <- read_csv("data/class_wide_1.csv")
library(here)
library(tidyverse)  # data manipulation
library(cluster)    # clustering algorithms
library(factoextra) # clustering visualization
library(dendextend) # for comparing two dendrograms


clust_data <- read_csv(here("data", "class_wide_1.csv")) # read in data

clust_data <- select(clust_data, -X1, -`54`) # remove extra col sub 54 has weird formatting

clust_data <- as.data.frame(clust_data) # turn into df
```

```r
rownames(clust_data) <- clust_data$speaker # make row names speaker

clust_data <- select(clust_data,-speaker) # remove extra col sub 54 has weird formatting

head(clust_data)# show first couple rows

# Dissimilarity matrix
d <- dist(clust_data, method = "euclidean")

# Hierarchical clustering using Average Linkage
hc1 <- hclust(d, method = "average" )

# Plot the obtained dendrogram
plot(hc1, cex = 0.6, hang = -1)


# Ward's method
hc5 <- hclust(d, method = "average" )

# Cut tree into 3 groups
sub_grp <- cutree(hc5, k = 3)

# Number of members in each cluster
table(sub_grp)

plot(hc5, cex = 0.6)
rect.hclust(hc5, k = 3, border = 2:5)


clust_data <- clust_data %>%
  mutate(cluster = sub_grp)

fviz_cluster(list(data = clust_data, cluster = sub_grp))
#making a empty dataframe
criteria <- data.frame()
#setting range of k
nk <- 1:20
#loop for range of clusters
for (k in nk) {
model <- kmeans(clust_data, k)
criteria <- rbind(criteria,c(k,model$tot.withinss,model$betweenss,model$totss))
}
#renaming columns
names(criteria) <- c("k","tot.withinss","betweenss","totalss")

#scree plot
ggplot(criteria, aes(x=k)) +
  geom_point(aes(y=tot.withinss),color="red") +
  geom_line(aes(y=tot.withinss),color="red") +
  geom_point(aes(y=betweenss),color="blue") +
  geom_line(aes(y=betweenss),color="blue") +
  xlab("k = number of clusters") + ylab("Sum of Squares (within = red, between = blue)")
```

References