Phase 2 GMM Final

Jessala Grijalva

2024-03-20

Introduction

Setup and Preliminaries

Establishes the environment, loads necessary libraries, and sets the seed for reproducibility. Loads and prepares data for analysis, which is a crucial first step in any data analysis process.

Libraries and Seed

```r
# Load necessary libraries
library(pacman)
p_load(mclust, cluster, dplyr, tidyverse, factoextra, flexclust, caret,
       dunn.test, MASS, Rtsne, NbClust, boot, fpc, ggplot2, gridExtra, grid,
       kableExtra, rstatix, PMCMRplus, FSA, DescTools, mclust, stats, purrr,
       fmsb, GGally, tinytex)

# Set seed for reproducibility
set.seed(2500)
```

Data Loading and Preparation

```r
# Set the working directory to the folder where the RDA file is located
load("~/Desktop/Dissertation Analysis/Dissertation Analysis/Data/Processed/lns_clean.rda")

# Check for missing data in the dataset
missing_data_summary <- sapply(lns_subset, function(x) sum(is.na(x)))

# Print the summary of missing data
print(missing_data_summary)
```

```
## CULTURAL_IDENTITY          AMERICAN          LEARNENG          DISTINCT
##                 0                 0                 0                 0
##          KEEPSPAN             BLEND         VOTE_PREF          IDEOLOGY
##                 0                 0                 0                 0
##          FEELPART           PARTYID             SAYSO          GOVTRUST
##                 0                 0                 0                 0
##           INCSUPP            HEALTH          DREAMACT           IMMVIEW
##                 0                 0                 0                 0
##         IMMPOLICY         EDUCATION       NATIONALITY      BIRTH_ORIGIN
##                 0                 0                 0                 0
##        GEN_STATUS               AGE               SEX             RACE1
##                 0                 0                 0                 0
##              HHINC          RELIGION             WEIGHT
##                  0                 0                  0
```

Clustering Preparation

Defines variables for clustering and standardizes them, which is necessary to ensure that all variables contribute equally to the analysis.

Define Clustering Variables

```
# Select variables for K Means clustering
selected_vars <- c("AMERICAN", "CULTURAL_IDENTITY", "KEEPSPAN", "DISTINCT",
                   "LEARNENG")
```

Standardization

```
# Standardize the selected variables. Directly converting the result into a matrix format suitable for 
clustering_data_matrix <- scale(lns_subset[selected_vars])

# Check if the standardization resulted in a matrix and print the dimensions to confirm
print(dim(clustering_data_matrix))
```

```
## [1] 4785    5
```

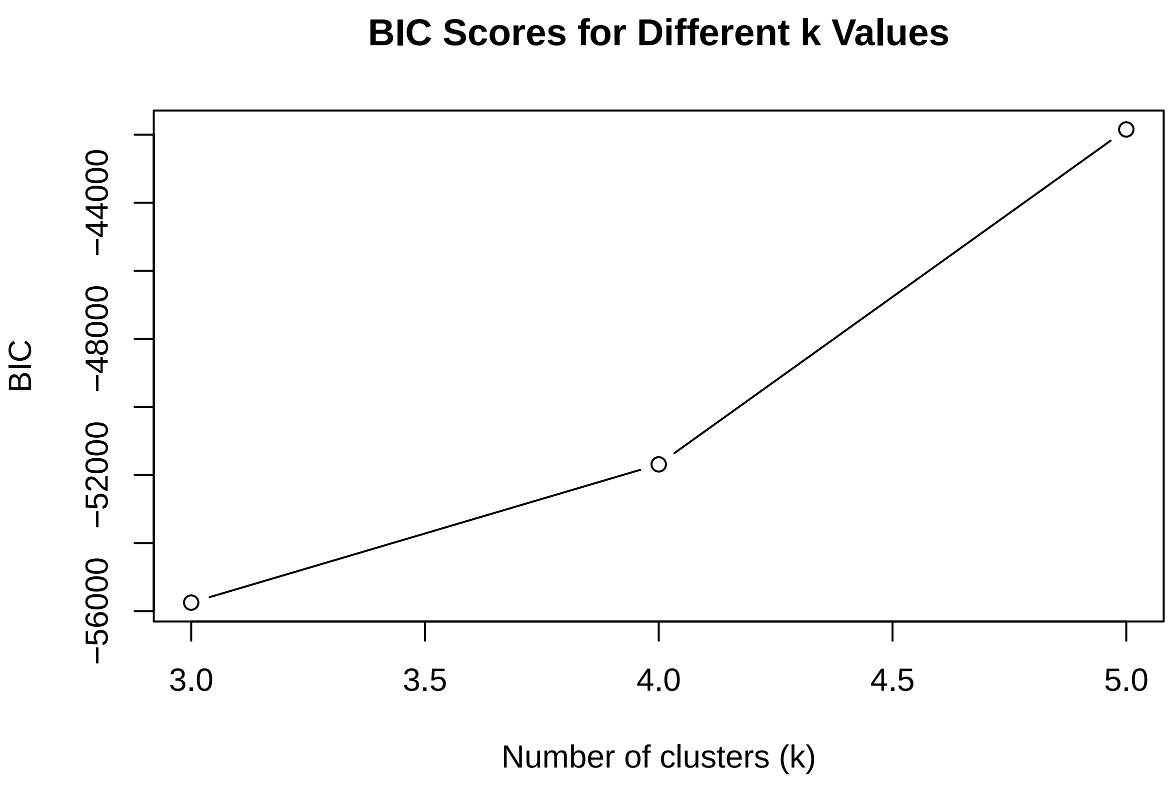
Clustering Evaluation Functions

Provides functions to calculate key clustering metrics. This is important to assess the quality of the clustering solution.

```
# Define the range of cluster numbers to evaluate
k_values <- c(3, 4, 5)

# Initialize a list to store GMM results and evaluation metrics for each k
gmm_results <- list()

for (k in k_values) {
    # Perform GMM clustering
    gmm_model <- Mclust(clustering_data_matrix, G = k)

    # Store the model
    gmm_results[[paste("GMM_k", k)]] <- gmm_model

    # Print basic model information including BIC
    cat("GMM Model for k =", k, "\n")
    print(summary(gmm_model, parameters = FALSE))
}
```

```
## GMM Model for k = 3
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust EEV (ellipsoidal, equal volume and shape) model with 3 components:
##
##  log-likelihood    n df       BIC       ICL
##       -27654.76 4785 52 -55750.13 -57658.22
##
## Clustering table:
##    1    2    3
##  724 2600 1461
## GMM Model for k = 4
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust EEV (ellipsoidal, equal volume and shape) model with 4 components:
##
##  log-likelihood    n df       BIC       ICL
##       -25556.28 4785 68 -51688.75 -52293.62
##
## Clustering table:
##    1    2    3    4
##  433  705  378 3269
## GMM Model for k = 5
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust EEV (ellipsoidal, equal volume and shape) model with 5 components:
##
##  log-likelihood    n df      BIC       ICL
##       -20567.52 4785 84 -41846.8 -42774.64
##
## Clustering table:
##    1    2    3    4    5
##  275  106 3135  604  665
```

```
# Extract and plot BIC scores from the GMM results
bic_values <- sapply(gmm_results, function(x) x$bic)
plot(k_values, bic_values, type = "b", xlab = "Number of clusters (k)", ylab = "BIC", main = "BIC Score
```



GMM Means Summary

```r
# Convert the original selected variables in the dataset to a matrix format for compatibility
original_data_matrix <- as.matrix(lns_subset[selected_vars])

# Initialize a list to store the mean values for each k
cluster_means_list <- list()

for (k in k_values) {
    # Retrieve the model
    gmm_model <- gmm_results[[paste("GMM_k", k)]]

    # Retrieve the cluster assignment from the model
    cluster_assignment <- gmm_model$classification

    # Calculate means for each cluster
    cluster_means <- aggregate(original_data_matrix, by = list(cluster = cluster_assignment), FUN = mea

    # Store the calculated means
    cluster_means_list[[paste("Means_k", k)]] <- cluster_means

    # Print the means for each cluster for the current k
    cat("Cluster Means for k =", k, ":\n")
    print(cluster_means)
}
```

```
## Cluster Means for k = 3 :
##   cluster AMERICAN CULTURAL_IDENTITY KEEPSPAN DISTINCT LEARNENG
## 1       1 3.473757          3.183011 3.261050 2.406077 3.243094
## 2       2 3.737692          3.428846 3.981154 2.701154 3.997308
## 3       3 2.657084          3.591718 3.617385 2.981520 3.995209
## Cluster Means for k = 4 :
##   cluster AMERICAN CULTURAL_IDENTITY KEEPSPAN DISTINCT LEARNENG
## 1       1 3.233256          3.411085 3.635104 2.711316 2.766744
## 2       2 3.880851          3.052482 3.327660 1.797163 3.988652
## 3       3 3.523810          3.158730 2.677249 2.997354 3.947090
## 4       4 3.256959          3.561946 3.996635 2.920465 4.000000
## Cluster Means for k = 5 :
##   cluster AMERICAN CULTURAL_IDENTITY KEEPSPAN DISTINCT LEARNENG
## 1       1 3.149091          3.643636 4.000000 2.767273 2.752727
## 2       2 3.481132          2.858491 3.207547 2.339623 2.764151
## 3       3 3.301435          3.575279 3.955343 2.999681 3.971930
## 4       4 3.344371          3.217715 3.973510 1.847682 4.000000
## 5       5 3.774436          3.022556 2.642105 2.393985 4.000000
```

```
# Convert the original selected variables in the dataset to a matrix format for compatibility
original_data_matrix <- as.matrix(lns_subset[selected_vars])

# Initialize a list to store the mean values and sizes for each k
cluster_means_and_sizes_list <- list()

for (k in k_values) {
    # Retrieve the model
    gmm_model <- gmm_results[[paste("GMM_k", k)]]

    # Retrieve the cluster assignment from the model
    cluster_assignment <- gmm_model$classification

    # Calculate means for each cluster
    cluster_means <- aggregate(original_data_matrix, by = list(cluster = cluster_assignment), FUN = mea

    # Calculate cluster sizes
    cluster_sizes <- table(cluster_assignment)

    # Convert cluster_sizes to a dataframe for easy merging
    cluster_sizes_df <- as.data.frame(cluster_sizes)
    names(cluster_sizes_df) <- c("cluster", "Size")

    # Merge cluster means and sizes
    cluster_means_and_sizes <- merge(cluster_means, cluster_sizes_df, by = "cluster")

    # Store the calculated means and sizes
    cluster_means_and_sizes_list[[paste("MeansAndSizes_k", k)]] <- cluster_means_and_sizes

    # Print the means and sizes for each cluster for the current k
    cat("\nCluster Means and Sizes for k =", k, ":\n")
    print(cluster_means_and_sizes)
}
```

```
##
## Cluster Means and Sizes for k = 3 :
##   cluster AMERICAN CULTURAL_IDENTITY KEEPSPAN DISTINCT LEARNENG Size
## 1       1 3.473757          3.183011 3.261050 2.406077 3.243094  724
## 2       2 3.737692          3.428846 3.981154 2.701154 3.997308 2600
## 3       3 2.657084          3.591718 3.617385 2.981520 3.995209 1461
##
## Cluster Means and Sizes for k = 4 :
##   cluster AMERICAN CULTURAL_IDENTITY KEEPSPAN DISTINCT LEARNENG Size
## 1       1 3.233256          3.411085 3.635104 2.711316 2.766744  433
## 2       2 3.880851          3.052482 3.327660 1.797163 3.988652  705
## 3       3 3.523810          3.158730 2.677249 2.997354 3.947090  378
## 4       4 3.256959          3.561946 3.996635 2.920465 4.000000 3269
##
## Cluster Means and Sizes for k = 5 :
##   cluster AMERICAN CULTURAL_IDENTITY KEEPSPAN DISTINCT LEARNENG Size
## 1       1 3.149091          3.643636 4.000000 2.767273 2.752727  275
## 2       2 3.481132          2.858491 3.207547 2.339623 2.764151  106
## 3       3 3.301435          3.575279 3.955343 2.999681 3.971930 3135
## 4       4 3.344371          3.217715 3.973510 1.847682 4.000000  604
## 5       5 3.774436          3.022556 2.642105 2.393985 4.000000  665
```

Cluster Labeling and Profiles

```
# Step 1: Extract Cluster Assignments for k=4
cluster_assignments_k4 <- gmm_results[["GMM_k 4"]]$classification

# Step 2: Add Cluster Assignments to the Dataset
# Assuming your dataset is named lns_subset
lns_subset$cluster_assignment_k4 <- cluster_assignments_k4

# Verification Step 1: Confirm the column has been added
if("cluster_assignment_k4" %in% colnames(lns_subset)) {
  cat("Verification Passed: 'cluster_assignment_k4' column added.\n")
} else {
  cat("Verification Failed: Column not found.\n")
}
```

```
## Verification Passed: 'cluster_assignment_k4' column added.
```

```
# Verification Step 2: Check Cluster Sizes
cluster_sizes <- table(lns_subset$cluster_assignment_k4)
cat("Cluster Sizes for k=4:\n")
```

```
## Cluster Sizes for k=4:
```

```
print(cluster_sizes)
```

```
##
##    1    2    3    4
##  433  705  378 3269
```

K-fold validation

```
# Generate folds ensuring indices are appropriate for the dataset
set.seed(2500)
folds <- createFolds(1:nrow(clustering_data_matrix), k = 5, list = TRUE)

# Function to perform GMM clustering and calculate silhouette score on test data
gmm_silhouette_cv <- function(train_indices, test_indices, data) {
  train_data <- data[train_indices, ]
  test_data <- data[test_indices, ]

  # Fit GMM on the training data with k=4
  gmm_fit <- Mclust(train_data, G = 4)

  # Obtain cluster predictions for the test data
  test_clusters <- predict(gmm_fit, newdata = test_data)$classification

  # Calculate the silhouette score for the test data
  silhouette_avg <- mean(silhouette(test_clusters, dist(test_data))[, "sil_width"])

  return(silhouette_avg)
}

# Apply the cross-validation function across all folds
cv_silhouette_scores <- sapply(folds, function(indices) {
  gmm_silhouette_cv(-indices, indices, clustering_data_matrix)
})

# Calculate mean and standard deviation of the silhouette scores across all folds
cv_mean_silhouette <- mean(cv_silhouette_scores)
cv_sd_silhouette <- sd(cv_silhouette_scores)

cat("Cross-Validation Mean Silhouette Score:", cv_mean_silhouette, "\n")
```

```
## Cross-Validation Mean Silhouette Score: 0.3566289
```

```
cat("Cross-Validation Silhouette Score SD:", cv_sd_silhouette, "\n")
```

```
## Cross-Validation Silhouette Score SD: 0.008071387
```

Bootstrapping (new)

Visualizations

Bootstrap

Parallel coordinates

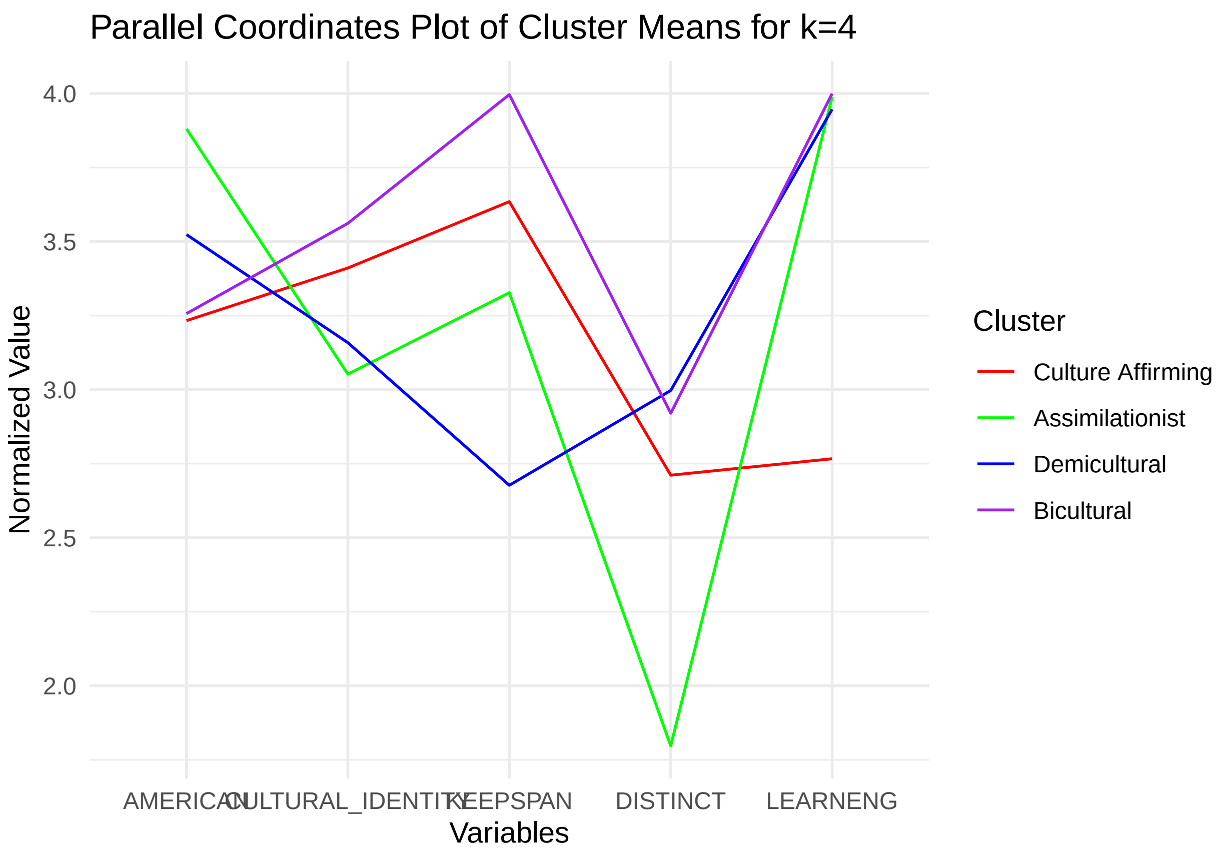
```
# Extract the cluster means for k=4 using the correct key from the list
cluster_means_k4 <- cluster_means_and_sizes_list[["MeansAndSizes_k 4"]]

# Check the structure of cluster_means_k4 to ensure it's as expected
str(cluster_means_k4)
```

```
## 'data.frame':    4 obs. of  7 variables:
##  $ cluster          : num  1 2 3 4
##  $ AMERICAN         : num  3.23 3.88 3.52 3.26
##  $ CULTURAL_IDENTITY: num  3.41 3.05 3.16 3.56
##  $ KEEPSPAN         : num  3.64 3.33 2.68 4
##  $ DISTINCT         : num  2.71 1.8 3 2.92
##  $ LEARNENG         : num  2.77 3.99 3.95 4
##  $ Size             : int  433 705 378 3269
```

```r
# If cluster_means_k4 is not NULL, print the first few rows to verify the content
if (!is.null(cluster_means_k4)) {
  print(head(cluster_means_k4))
} else {
  stop("Cluster means for k=4 not found in the list. Please check the list structure.")
}
```

```
##   cluster AMERICAN CULTURAL_IDENTITY KEEPSPAN DISTINCT LEARNENG Size
## 1       1 3.233256          3.411085 3.635104 2.711316 2.766744  433
## 2       2 3.880851          3.052482 3.327660 1.797163 3.988652  705
## 3       3 3.523810          3.158730 2.677249 2.997354 3.947090  378
## 4       4 3.256959          3.561946 3.996635 2.920465 4.000000 3269
```

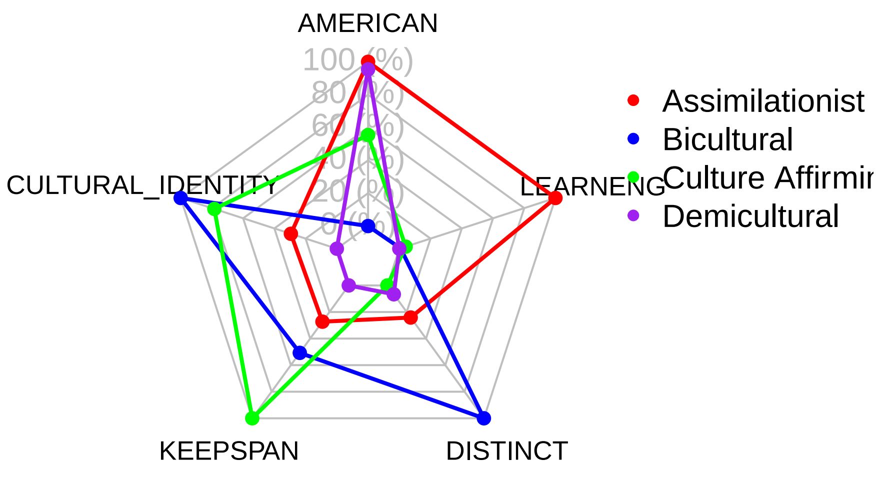
```r
# Ensure the cluster column is correctly set as a factor
cluster_means_k4$cluster <- factor(cluster_means_k4$cluster, levels = c(1, 2, 3, 4),
                                   labels = c("Culture Affirming", "Assimilationist", "Demicultural", "

# Create Parallel Coordinates Plot without the 'order' parameter
p <- ggparcoord(data = cluster_means_k4,
                columns = 2:6, # Indices of the numeric columns to include in the plot
                groupColumn = "cluster", # Name of the cluster column
                scale = "globalminmax") +
  scale_color_manual(values = c("red", "green", "blue", "purple"),
                     labels = c("Culture Affirming", "Assimilationist", "Demicultural", "Bicultural"),
                     name = "Cluster") +
  labs(title = "Parallel Coordinates Plot of Cluster Means for k=4",
       x = "Variables",
       y = "Normalized Value") +
  theme_minimal() +
  theme(legend.position = "right")

# Display plot
print(p)
```



Radar Chart

```
# Creating a data frame for the radar chart with minimum and maximum values
radar_data <- data.frame(
  cluster = factor(c("Culture Affirming", "Assimilationist", "Demicultural", "Bicultural")),
  AMERICAN = c(3.23, 3.88, 3.52, 3.26),
  CULTURAL_IDENTITY = c(3.41, 3.05, 3.16, 3.56),
  KEEPSPAN = c(3.64, 3.33, 2.68, 4),
  DISTINCT = c(2.71, 1.8, 3, 2.92),
  LEARNENG = c(2.77, 3.99, 3.95, 4)
)

# Define the maximum and minimum values for each variable
max_values <- apply(radar_data[, -1], 2, max)
min_values <- apply(radar_data[, -1], 2, min)

# Normalize the radar_data to the scale of the plot
radar_data_normalized <- as.data.frame(mapply(function(x, min, max) (x - min) / (max - min),
                                              radar_data[, -1], min_values, max_values))

# Creating the radar chart data
radar_chart_data <- rbind(
  min = rep(0, times=ncol(radar_data_normalized)),
  max = rep(1, times=ncol(radar_data_normalized)),
  radar_data_normalized
)
```

```r
# Adding cluster labels as row names
rownames(radar_chart_data)[3:6] <- as.character(radar_data$cluster)

# Plot the radar chart
radarchart(radar_chart_data, axistype = 1,
           seg = 5,
           pcol = c("red", "blue", "green", "purple"),
           plty = 1,
           plwd = 2,
           cglcol="grey", cglty=1, axislabcol="grey",
           vlcex=0.8)

# Add a legend outside the plot
legend(x=1.2, y=1, legend = levels(radar_data$cluster), bty = "n", pch=20, col=c("red", "blue", "green"
```



Save the clusters in an RDA file

```r
# Ensure the cluster_assignment_k4 variable is a factor
lns_subset$cluster_assignment_k4 <- factor(lns_subset$cluster_assignment_k4)

# Define the labels for each cluster
cluster_labels <- c('Culture Affirming', 'Assimilationist', 'Demicultural', 'Bicultural')

# Assign names to the levels of the clusters
levels(lns_subset$cluster_assignment_k4) <- cluster_labels

# Rename cluster_assignment_k4 to 'clusters'
names(lns_subset)[names(lns_subset) == 'cluster_assignment_k4'] <- 'clusters'

# Verify the renaming and releveling
str(lns_subset$clusters)
```

```
##  Factor w/ 4 levels "Culture Affirming",..: 4 4 4 1 4 4 4 4 4 3 ...
```

```r
# Check the size of each cluster to ensure it aligns with the given sizes
print(table(lns_subset$clusters))
```

```
##
## Culture Affirming   Assimilationist      Demicultural        Bicultural
##                 433                 705                 378                3269
```

```r
# Save the dataset with the new cluster labels
save(lns_subset, file = "~/Desktop/Dissertation Analysis/Dissertation Analysis/Data/Processed/lns_clust
```

End of Script