Math_574M_Hw7

Saheed Adisa Ganiyu

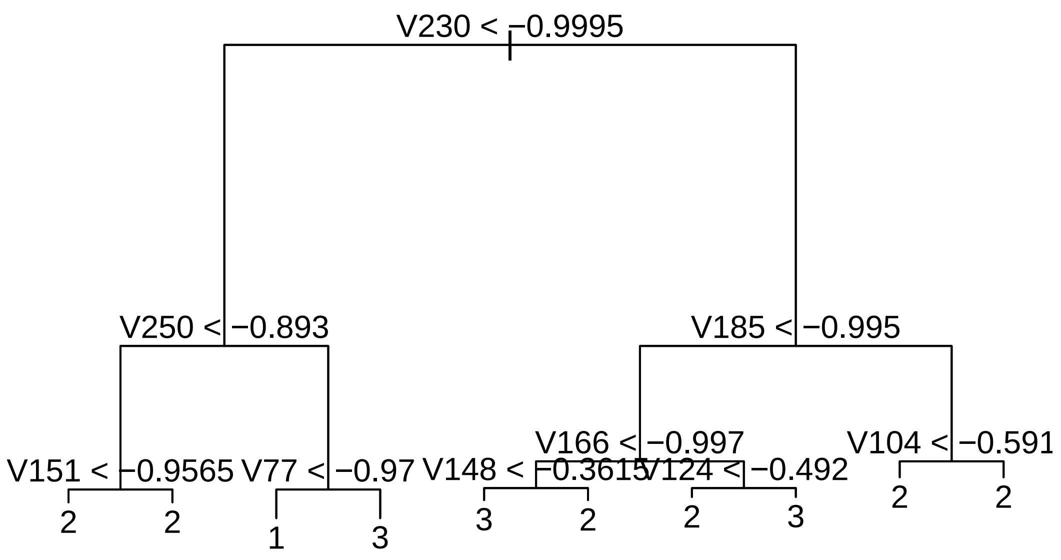
2023-12-04

6. Download the Zip code and classify the three digits " 1 ",” 2 " and " 3 ". In R, install the package”tree”.

(a) Fit a classification tree based on the training set, using “Deviance” as a split criterion. Plot the unpruned tree, and report the size (total number of terminal nodes) of the tree.

```r
library(tree)
set.seed(123)

# Loading the dataset
train_data <- read.table("zip.train", sep = "")
test_data <- read.table("zip.test", sep = "")

# taking the subsets of the data with class Y in {1, 2, 3}
class_labels <- c(1, 2, 3)
train_data <- train_data[train_data$V1 %in% class_labels, ]
test_data <- test_data[test_data$V1 %in% class_labels, ]
train_data$V1 <- as.factor(train_data$V1)
test_data$V1 <- as.factor(test_data$V1)
```

```r
# Fitting the model with Deviance split
tree_model <- tree(V1 ~ ., data = train_data, split = "deviance")

# Plotting
plot(tree_model)
text(tree_model, pretty = 0)
```



```r
# Size of the unpruned tree
unpruned_size <- (summary(tree_model))$size
cat("Size of the unpruned tree:", unpruned_size, "\n")
```

```
## Size of the unpruned tree: 10
```

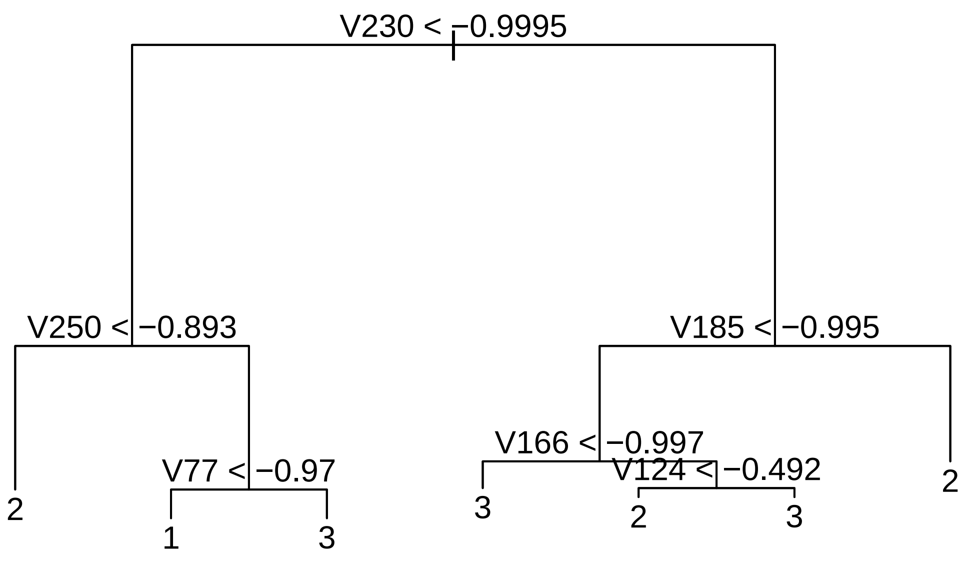
(b) Tune the size of tree, using 10 -fold cross validation and the "misclassification rate" as a criteria. Find the size of the smallest tree size with low deviance.

```r
# Tune tree size using 10-fold cross-validation and misclassification rate
cv_tree <- cv.tree(tree_model, FUN = prune.misclass, K=10)

# Size of the smallest tree with low deviance
min_idx <- which.min(cv_tree$dev)
best_size <- cv_tree$size[min_idx]
cat("Best size from cross-validation:", best_size, "\n")
```

```
## Best size from cross-validation: 7
```

(c) Prune the tree to the best size from (b). Plot the pruned tree, generate its confusion table for the test data, and compute the training and test errors.

```r
# fitting the model with best size from cross-validation
pruned_tree <- prune.misclass(tree_model, best = best_size)

plot(pruned_tree)
text(pruned_tree, pretty = 0)
```



```r
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```r
# Predictions on test data
predictions <- predict(pruned_tree, newdata = test_data, type = "class")

# Confusion table for the test data
conf_table <- table(predictions, test_data$V1)
confusionMatrix(conf_table)
```

```
## Confusion Matrix and Statistics
##
##
## predictions   1   2   3
##           1 248   2   1
##           2  13 178  27
##           3   3  18 138
##
## Overall Statistics
##
##                Accuracy : 0.8981
##                  95% CI : (0.8717, 0.9206)
##     No Information Rate : 0.4204
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.8446
##
##  Mcnemar's Test P-Value : 0.01247
##
## Statistics by Class:
##
```

```
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            0.9394   0.8990   0.8313
## Specificity            0.9918   0.9070   0.9545
## Pos Pred Value         0.9880   0.8165   0.8679
## Neg Pred Value         0.9576   0.9512   0.9403
## Prevalence             0.4204   0.3153   0.2643
## Detection Rate         0.3949   0.2834   0.2197
## Detection Prevalence   0.3997   0.3471   0.2532
## Balanced Accuracy      0.9656   0.9030   0.8929
```

```r
cat( "\n")
```

```r
# Computing training and test errors
train_error <- 1 - sum(predict(pruned_tree, type = "class") == train_data$V1) / nrow(train_data)
test_error <- 1 - sum(predictions == test_data$V1) / nrow(test_data)

cat("Training Error:", train_error, "\n")
```

```
## Training Error: 0.05137845
```

```r
cat("Test Error:", test_error, "\n")
```

```
## Test Error: 0.1019108
```

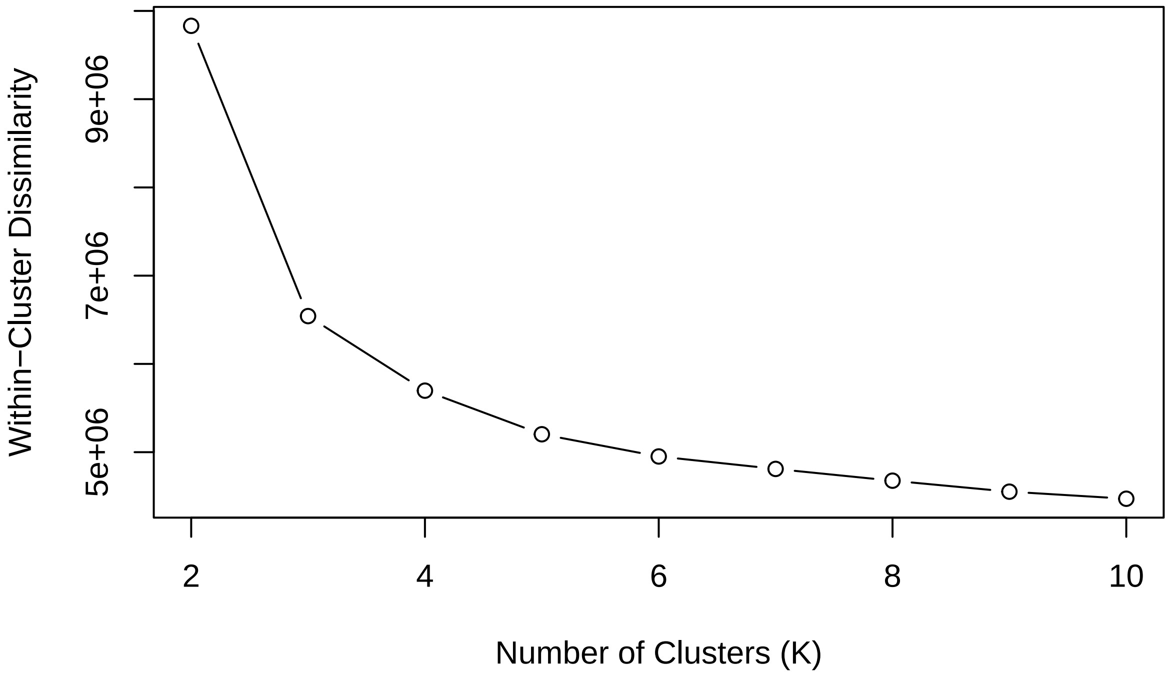
7. Download the Phoneme dataset from the textbook website. Perform cluster analysis using columns $x.1$ to x.256.

(a) For $K = 2, 3, \ldots, 10$, use kmeans to cluster the data points into K clusters. For each K, use ten different sets of initial centroids by setting nstart $= 10$ and record the within cluster dissimilarity (WC). Plot WC against K. Choose an appropriate K based on your plot.

```r
# Loading the dataset from the book website+
phoneme_data <- read.table(
"https://hastie.su.domains/ElemStatLearn/datasets/phoneme.data", sep = ",", header = TRUE, row.names = 
```

```r
library(stats)
phoneme_subset <- phoneme_data[, -c(ncol(phoneme_data)-1, ncol(phoneme_data))]

# Function to perform k-means clustering with different K values
kmeans_analysis <- function(K) {
  set.seed(123)
  kmeans_result <- kmeans(phoneme_subset, centers = K, nstart = 10)
  return(kmeans_result$tot.withinss)
}

# Applying the function for K = 2 to 10
K_values <- 2:10
WC_values <- sapply(K_values, kmeans_analysis)
```

```r
# Plot within cluster dissimilarity (WC) against K
plot(K_values, WC_values, type = "b", xlab = "Number of Clusters (K)", ylab = "Within-Cluster Dissimila
```



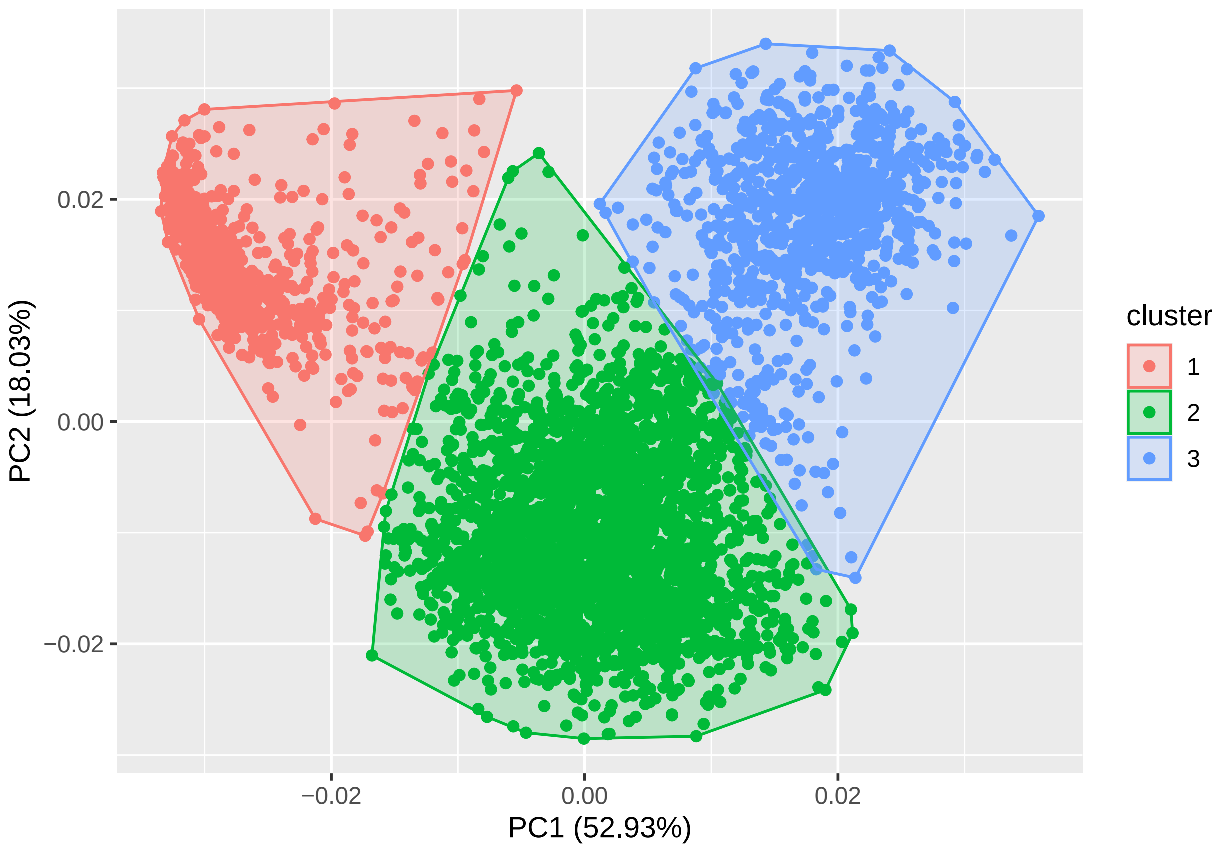
(b) Find the cluster assignment corresponding to the chosen K in part (a). Create a confusion matrix for the cluster assignment and phoneme (column g). Interpret the K clusters based on the confusion matrix.

```r
set.seed(13)
chosen_K <- 3
kmeans_result <- kmeans(phoneme_subset, centers = chosen_K, nstart = 10)
cluster_assignments <- kmeans_result$cluster
table(phoneme_data$g, cluster_assignments)
```

```
##      cluster_assignments
##          1    2    3
##   aa     0  690    5
##   ao     8 1013    1
##   dcl  717   36    4
##   iy     6 1052  105
##   sh     0    6  866
```

Comment: From the computed confusion matrix above, we can see that almost of “aa” are fall in cluster 2, same as “ao” as well, it just only its 8 members fall to cluster 1. Most of “sh” members also fall to cluster 3, where only its 6 member fall to cluster 2. Generally, cluster 2 has the highest members.

```r
library(ggfortify)
autoplot(kmeans_result, phoneme_subset, frame= TRUE)
```



(c) Perform hierarchical clustering based on the Euclidean distance and complete linkage. Find out the cluster assignments corresponding to 4 clusters and 5 clusters. Create a confusion matrix for the two cluster assignments. Create another confusion matrix for the 4- and 5-cluster assignments based on kmeans. Summarize the main difference between the results from two clustering algorithms.

```r
# computing hierarchical clustering
hierarchical_result <- hclust(dist(phoneme_subset), method = "complete")

# Cutting the dendrogram to get cluster assignments for 4 and 5 clusters
cut_4_clusters <- cutree(hierarchical_result, k = 4)
cut_5_clusters <- cutree(hierarchical_result, k = 5)

table(phoneme_data$g, cut_4_clusters)
```

```
##      cut_4_clusters
##         1   2   3   4
##   aa  109 586   0   0
##   ao  463 558   1   0
##   dcl  78   2 677   0
##   iy  480 681   1   1
##   sh   69  58   0 745
```

```r
cat("\n\n")
```

```r
table(phoneme_data$g, cut_5_clusters)
```

```
##      cut_5_clusters
##         1   2   3   4   5
##   aa    0 586   0 109   0
##   ao    0 558   1 463   0
##   dcl  42   2 677  36   0
##   iy    6 681   1 474   1
##   sh   69  58   0   0 745
```

```r
cat("\n\n")
```

```r
# for k-means clustering (chosen K)
table(phoneme_data$g, cluster_assignments)
```

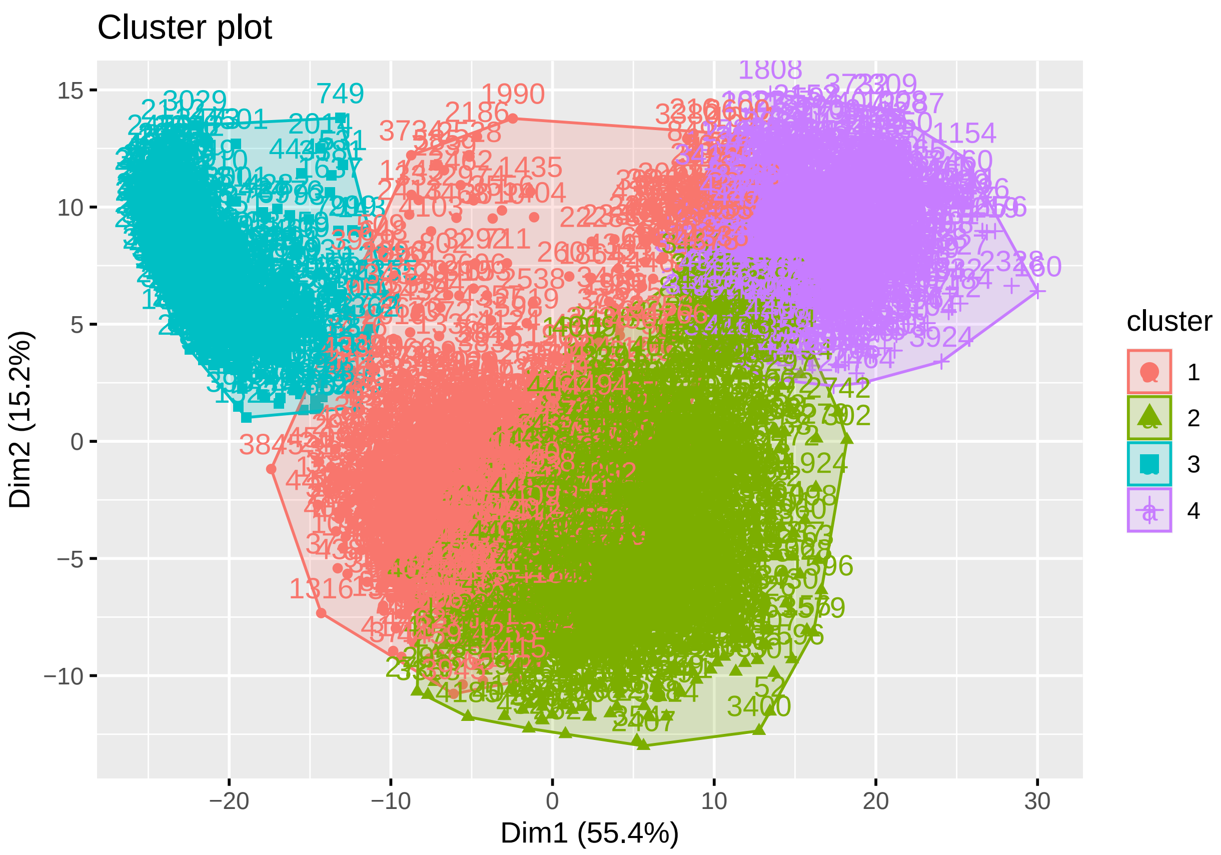
```
##      cluster_assignments
##          1    2    3
##   aa     0  690    5
##   ao     8 1013    1
##   dcl  717   36    4
##   iy     6 1052  105
##   sh     0    6  866
```

Comment: Based on our results, we observed that K-means clustering perform better than hierarchical clustering, as k-means has clear and better clusters with little overlapping unlike the case of hierarchical clustering which can be confirmed from the above confusion matrices and the following two graphs compared to the one (cluster plot) for k-means clustering.

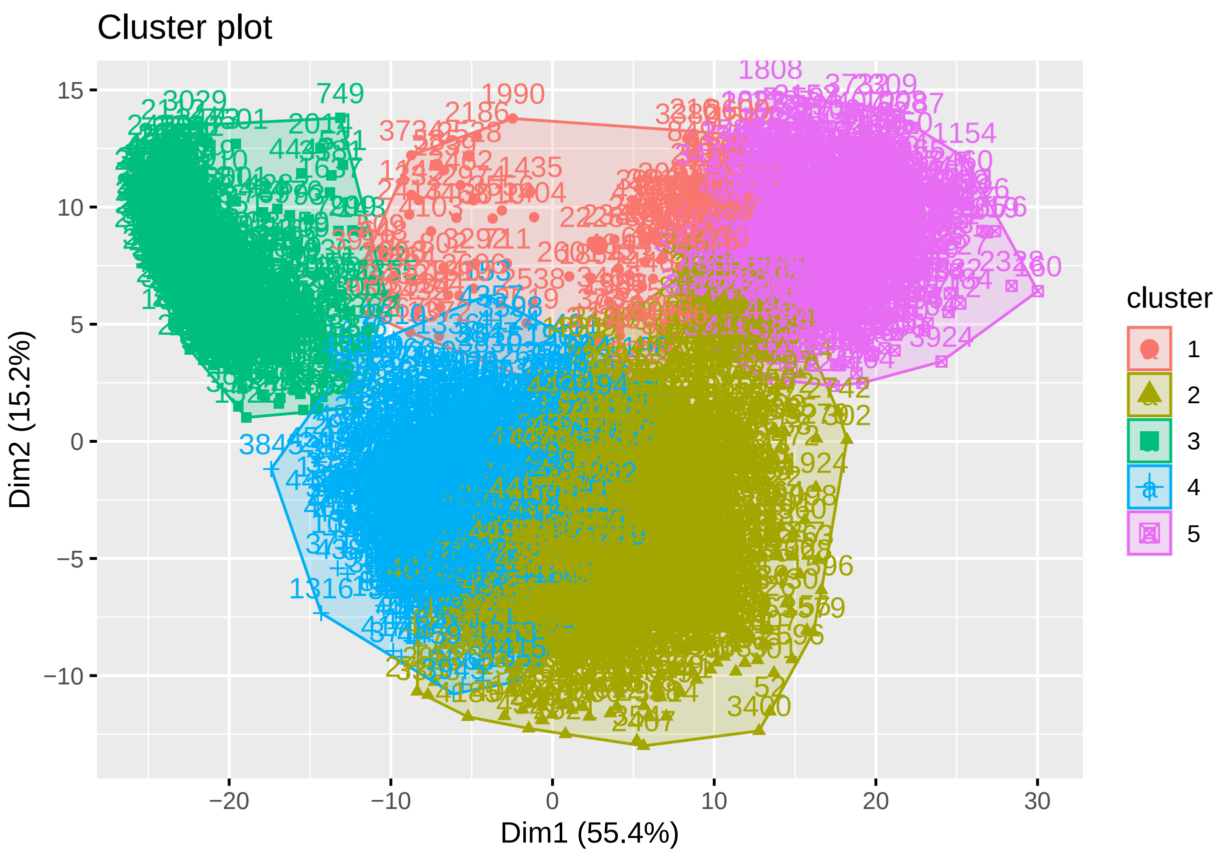
```r
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```r
fviz_cluster(list(data=phoneme_subset, cluster= cut_4_clusters))
```



```r
fviz_cluster(list(data=phoneme_subset, cluster= cut_5_clusters))
```



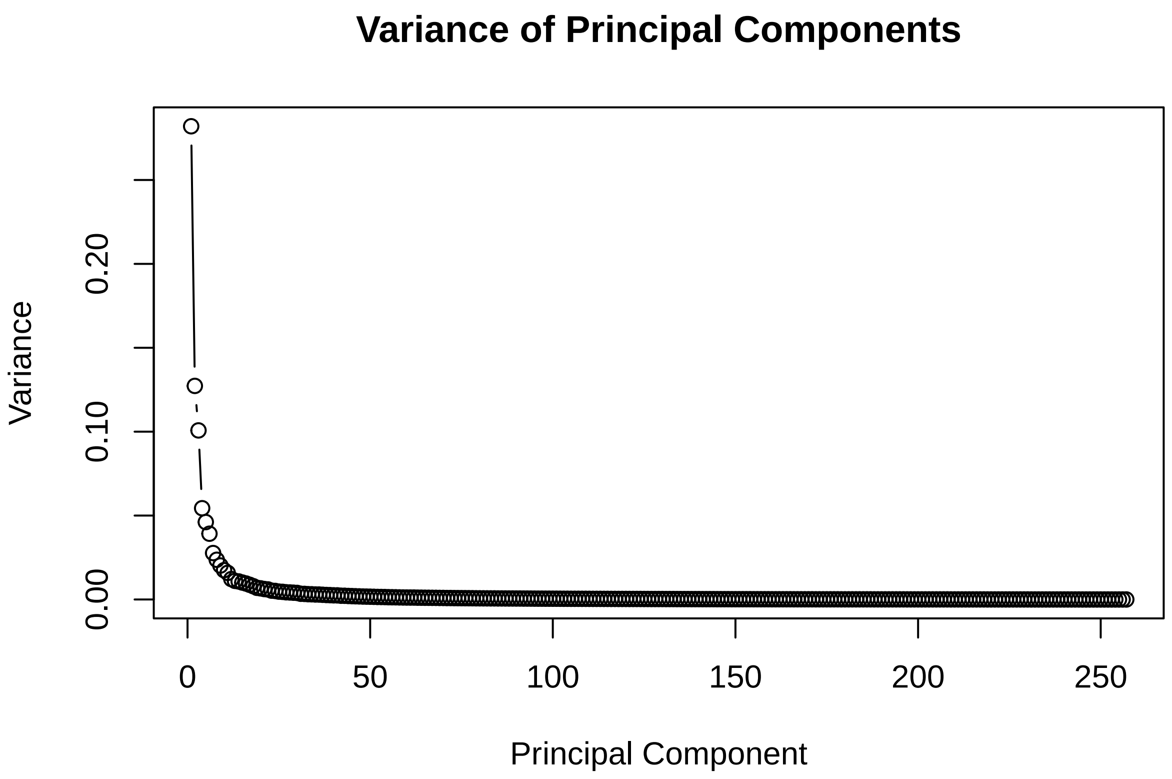
8. Consider the data points corresponding to digit 0 in the zip code training data. Perform principal component analysis (PCA) on the data points.\

(a) Plot the variances of principal components (PCs) against the indices of the PCs.

```
library(stats)
set.seed(123)

# Loading the dataset
train_data2 <- read.table("zip.train", sep = "")
test_data2 <- read.table("zip.test", sep = "")

# taking the subsets of the data with class Y in {1, 2, 3}
class_labels2 <- c(0)
train_data2 <- train_data2[train_data2$V1 %in% class_labels2, ]
```

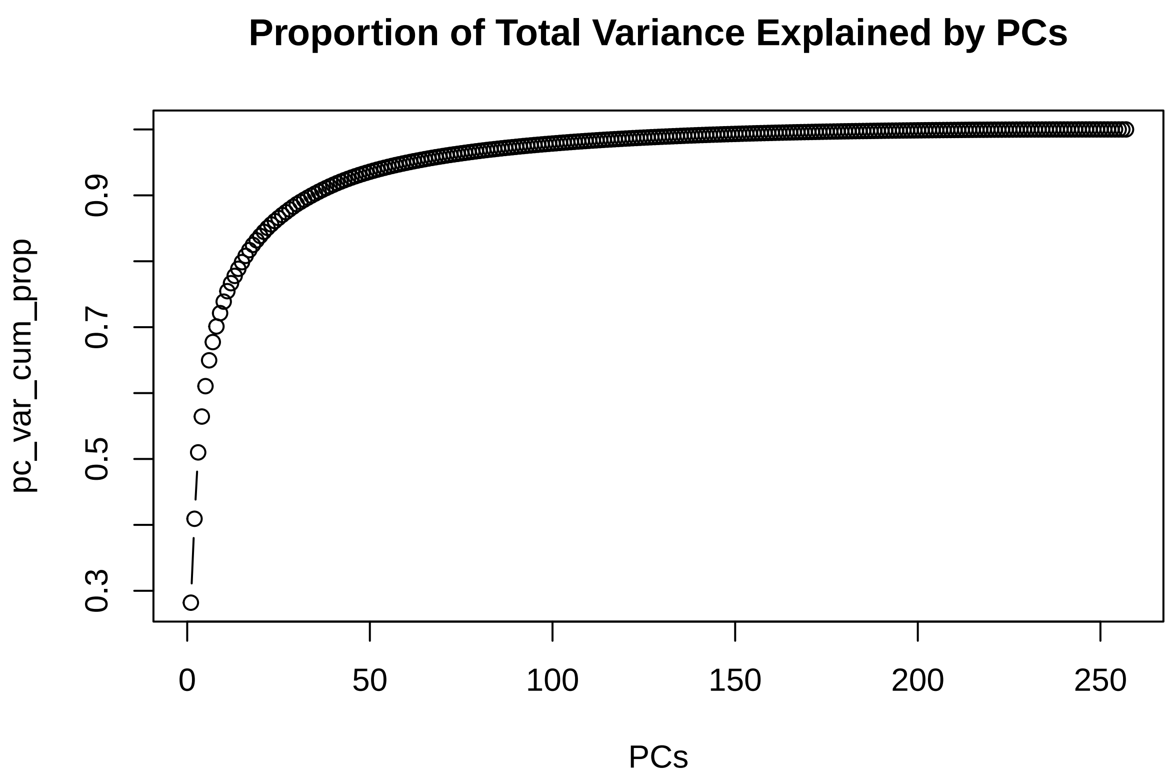
```
# Perform PCA
pca_Zip <- prcomp(train_data2)

# variance of each principal components
pc_var <- pca_Zip$sdev^2
pc_var_prop <- pc_var / sum(pc_var)
```

```r
# Plotting variances of principal components against indices of PCs
plot(pc_var_prop, type = "b",
     xlab = "Principal Component", ylab = "Variance",
     main = "Variance of Principal Components")
```



(b) For $j = 1, \ldots, 256$, compute the proportion of total variance explained by the first j components. Plot the proportions against j.

```r
# Compute proportion of total variance explained by the first j components
cumulative_var <- cumsum(pc_var) / sum(pc_var)

# Plot proportions against j
plot(1:length(cumulative_var), cumulative_var, type = "b",
     xlab = "PCs",
     ylab = "pc_var_cum_prop",
     main = "Proportion of Total Variance Explained by PCs")
```



(c) Based on (b), how many PCs are needed at least to explain 80% of the total variance in the data points? How many PCs are needed at least for 95% of the total variance?

```
# the number of PCs needed for 80% and 95% of the total variance
pc_80 <- which(cumulative_var >= 0.8)[1]
pc_95 <- which(cumulative_var >= 0.95)[1]

cat("Number of PCs needed for 80% of total variance:", pc_80, "\n")
```

```
## Number of PCs needed for 80% of total variance: 16
```

```
cat("Number of PCs needed for 95% of total variance:", pc_95, "\n")
```

```
## Number of PCs needed for 95% of total variance: 61
```