Fuel Data

Faith Taylor

2025-04-09

```
library(readxl) #packages needed to complete data analysis
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
library(ISLR)
library(tinytex)
library(e1071)
library(readr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(cluster)
library(dbscan)
```

```
##
## Attaching package: 'dbscan'
```

```
## The following object is masked from 'package:stats':
##
##     as.dendrogram
```

```
library(fpc)
```

```
##
## Attaching package: 'fpc'
```

```
## The following object is masked from 'package:dbscan':
##
##     dbscan
```

```
library(factoextra)
```

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

```
library(class)
```

```
fuel_data <- read_xlsx("C:/Users/Faith/OneDrive/Grad School/Semester 1 Working Files/EIA923.xls
x") #read the data file
summary(fuel_data) #give summary statistics
```

```
##   plant_id_eia    plant_id_eia_label fuel_type_code_pudl fuel_received_units
##  Min.   :   3.0   Length:756         Length:756          Min.   :      1
##  1st Qu.: 527.0   Class :character   Class :character    1st Qu.:   1300
##  Median : 728.0   Mode  :character   Mode  :character    Median :  12570
##  Mean   : 808.7                                         Mean   :  89258
##  3rd Qu.:1252.0                                         3rd Qu.:  42025
##  Max.   :1710.0                                         Max.   :4823176
##  fuel_mmbtu_per_unit sulfur_content_pct ash_content_pct  fuel_cost_per_mmbtu
##  Min.   : 0.857      Min.   :0.0000     Min.   : 0.000   Min.   : 0.343
##  1st Qu.: 1.028      1st Qu.:0.0000     1st Qu.: 0.000   1st Qu.: 1.948
##  Median :17.141      Median :0.3200     Median : 5.000   Median : 3.080
##  Mean   :13.349      Mean   :0.7314     Mean   : 4.995   Mean   : 5.786
##  3rd Qu.:23.870      3rd Qu.:0.9400     3rd Qu.: 9.600   3rd Qu.: 8.401
##  Max.   :29.400      Max.   :6.6100     Max.   :20.900   Max.   :29.510
```

```
fuel_data <- fuel_data[, -c(1,2)] #remove the first 2 columns as they will not be utilized in th
e models (plant_id_eia and plant_id_eia_label)
```

```
non_numeric_columns <- !sapply(fuel_data, is.numeric)
unique_values <- sapply(fuel_data[, non_numeric_columns], unique) #identify unique values in the
one non-numeric column that remains
unique_values
```

```
##      fuel_type_code_pudl
## [1,] "gas"
## [2,] "oil"
## [3,] "coal"
```

```
fuel_data[sapply(fuel_data, is.character)] <- lapply(fuel_data[sapply(fuel_data, is.character)], factor) #make fuel_type_code_pud1 into a factor.
summary(fuel_data)
```

```
##  fuel_type_code_pudl fuel_received_units fuel_mmbtu_per_unit sulfur_content_pct
##  coal:422            Min.   :      1     Min.   : 0.857      Min.   :0.0000
##  gas :265            1st Qu.:   1300     1st Qu.: 1.028      1st Qu.:0.0000
##  oil : 69            Median :  12570     Median :17.141      Median :0.3200
##                      Mean   :  89258     Mean   :13.349      Mean   :0.7314
##                      3rd Qu.:  42025     3rd Qu.:23.870      3rd Qu.:0.9400
##                      Max.   :4823176     Max.   :29.400      Max.   :6.6100
##  ash_content_pct  fuel_cost_per_mmbtu
##  Min.   : 0.000   Min.   : 0.343
##  1st Qu.: 0.000   1st Qu.: 1.948
##  Median : 5.000   Median : 3.080
##  Mean   : 4.995   Mean   : 5.786
##  3rd Qu.: 9.600   3rd Qu.: 8.401
##  Max.   :20.900   Max.   :29.510
```

#Part A - Clustering

```
fuel_data.numerical <- fuel_data[ , -c(1)] #isolating to just numeric values to normalize the da
ta for clustering (removing the fuel types)
fuel_data.norm <- scale(fuel_data.numerical) #normalized dataset to use for clustering
summary(fuel_data.norm) #summary of normalized dataset
```

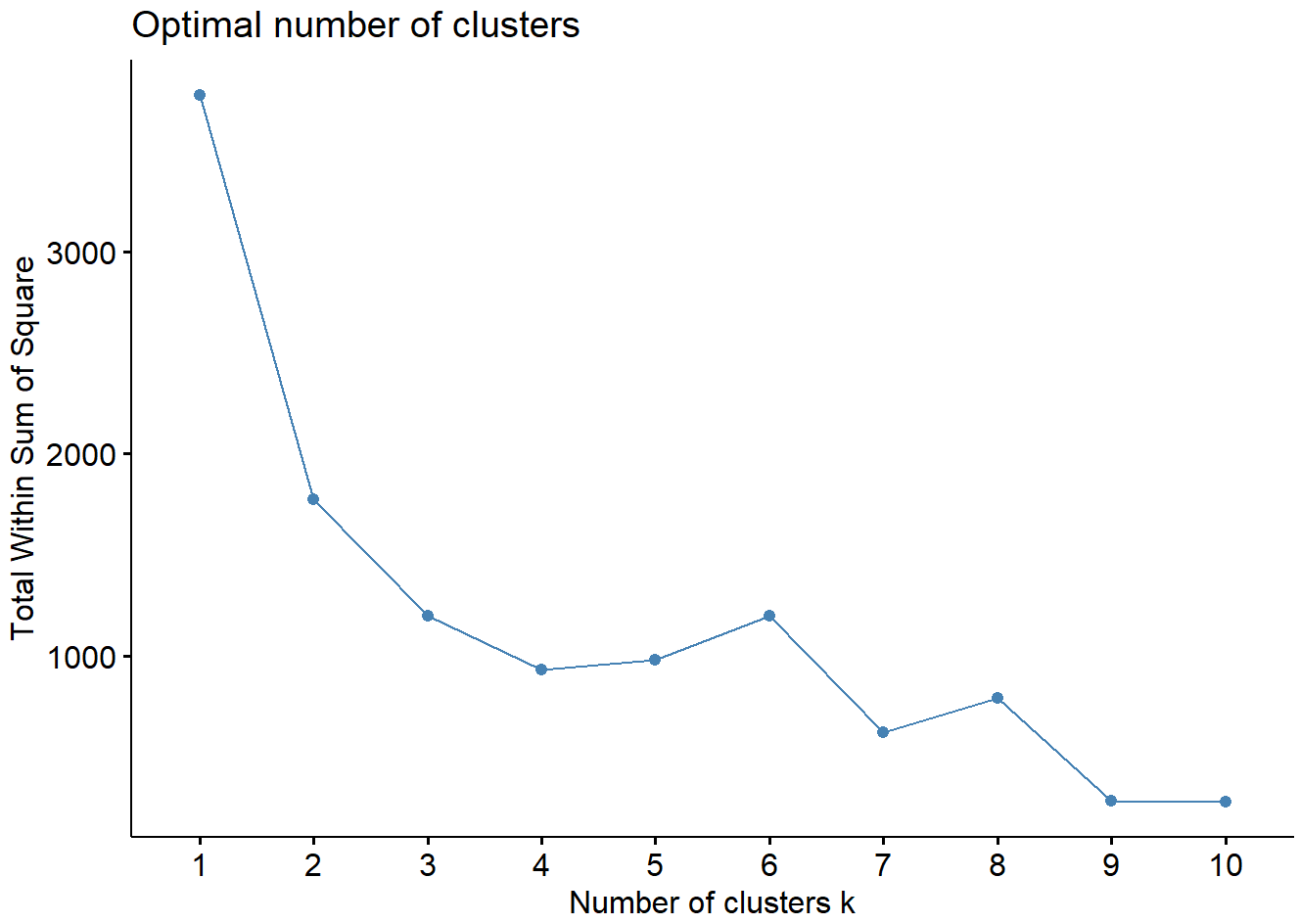
```
##  fuel_received_units fuel_mmbtu_per_unit sulfur_content_pct
##  Min.   :-0.2334     Min.   :-1.1940     Min.   :-0.6988
##  1st Qu.:-0.2300     1st Qu.:-1.1776     1st Qu.:-0.6988
##  Median :-0.2005     Median : 0.3624     Median :-0.3930
##  Mean   : 0.0000     Mean   : 0.0000     Mean   : 0.0000
##  3rd Qu.:-0.1235     3rd Qu.: 1.0055     3rd Qu.: 0.1993
##  Max.   :12.3785     Max.   : 1.5341     Max.   : 5.6168
##  ash_content_pct      fuel_cost_per_mmbtu
##  Min.   :-0.9756003   Min.   :-1.0207
##  1st Qu.:-0.9756003   1st Qu.:-0.7198
##  Median : 0.0009093   Median :-0.5075
##  Mean   : 0.0000000   Mean   : 0.0000
##  3rd Qu.: 0.8992982   3rd Qu.: 0.4902
##  Max.   : 3.1062100   Max.   : 4.4486
```

```
cor_matrix <- data.frame(cor(fuel_data.norm)) #find the correlation matrix of the variables
cor_matrix
```

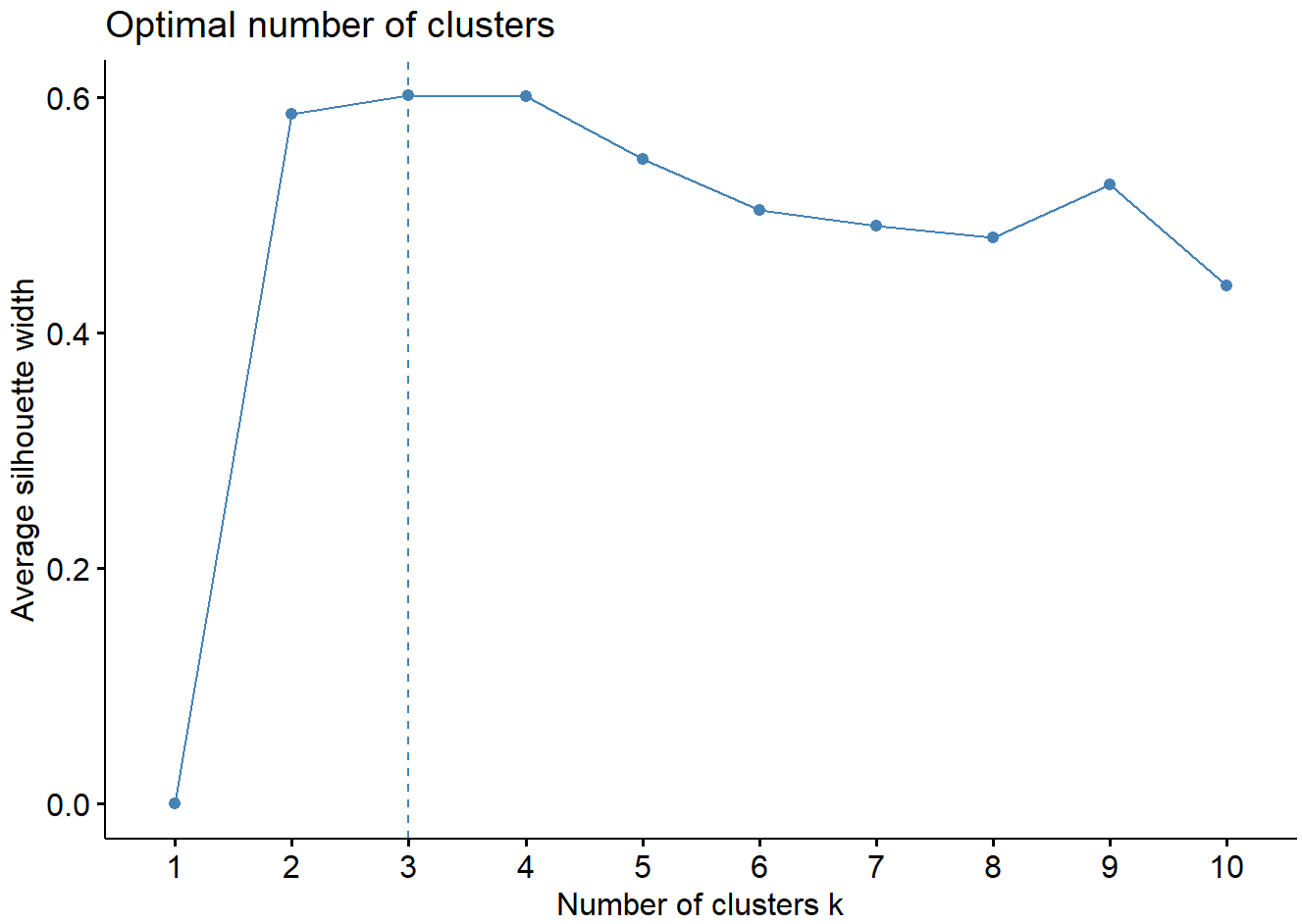
```
##                     fuel_received_units fuel_mmbtu_per_unit sulfur_content_pct
## fuel_received_units          1.00000000          -0.1691732        -0.09760813
## fuel_mmbtu_per_unit         -0.16917323           1.0000000         0.65330830
## sulfur_content_pct          -0.09760813           0.6533083         1.00000000
## ash_content_pct             -0.13784580           0.8790027         0.56710485
## fuel_cost_per_mmbtu          0.05412650          -0.6662766        -0.43643591
##                     ash_content_pct fuel_cost_per_mmbtu
## fuel_received_units      -0.1378458           0.0541265
## fuel_mmbtu_per_unit       0.8790027          -0.6662766
## sulfur_content_pct        0.5671048          -0.4364359
## ash_content_pct           1.0000000          -0.6514981
## fuel_cost_per_mmbtu      -0.6514981           1.0000000
```

#Correlation is positive and strong between mmbtu per unit and ash content at 87.9%. Additionally, there is also a positive correlation between mmbtu per unit and sulfur content at 56.7%.There is a negative correlation between mmbtu per unit and cost per mmbtu. Does this mean the more you buy, the cost goes down? #As sulfur and ash content increase, the cost goes down. This makes since as coal is the cheapest and is the dirtiest.

```
fviz_nbclust(fuel_data.norm, kmeans, method = "wss") #elbow method to find optimal k
```

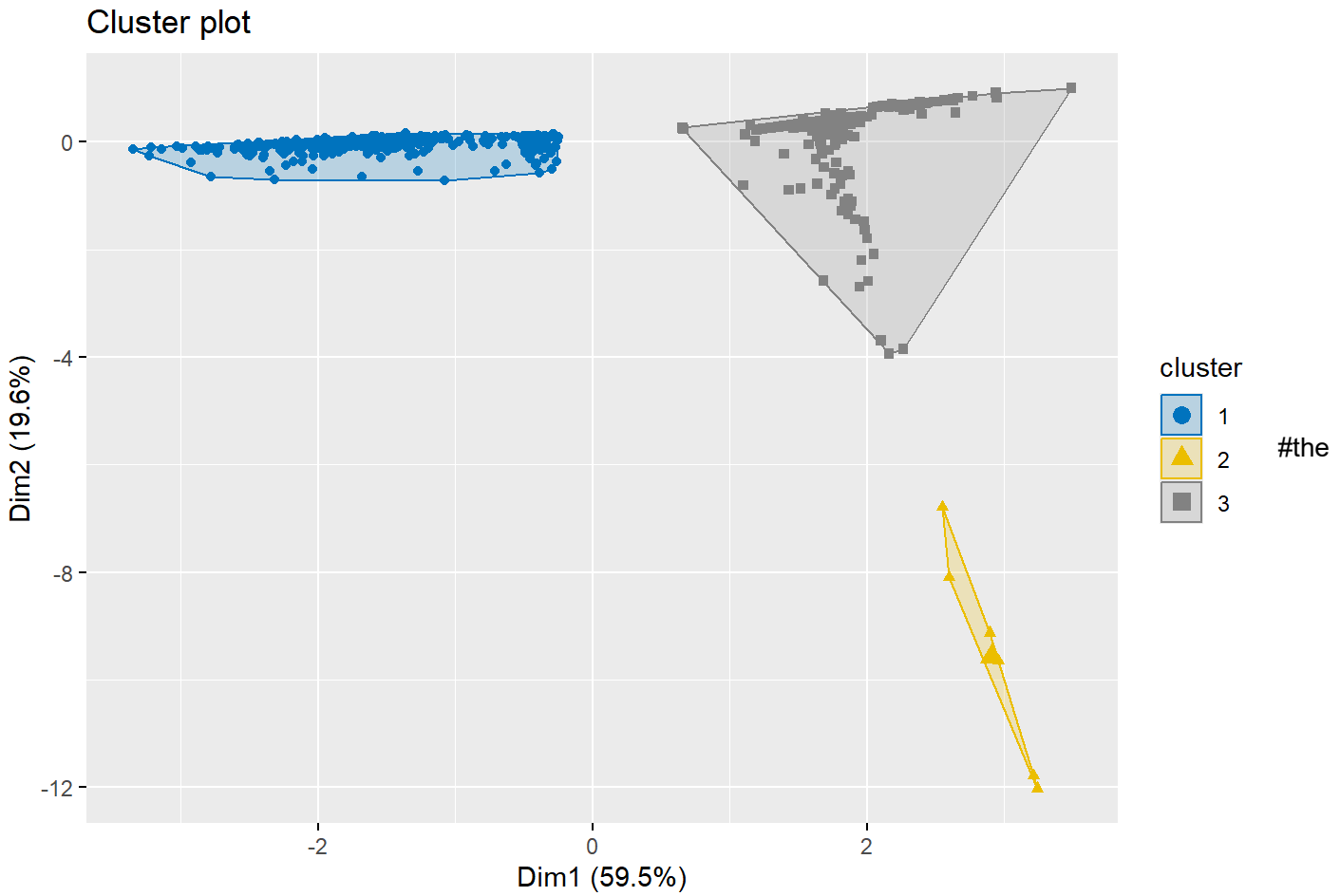



```
fviz_nbclust(fuel_data.norm, kmeans, method = "silhouette") ## Silhouette method to confirm the optimal number of clusters
```



#Silhouette method confirms optimal number of clusters is 3 - this matches the 3 different types of fuel we have in the data.

```
set.seed(123)  # For reproducibility
kmeans.fuel <- kmeans(fuel_data.norm, centers = 3, nstart = 25)  # Optimal k from the previous s
tep is 3.
fviz_cluster(kmeans.fuel, data = fuel_data.norm, geom = "point",
             ellipse.type = "convex", palette = "jco") #used to visualize the clusters
```



patterns here seem to be densely packed - maybe DBSCAN will be more helpful with this dataset.

```
kmeans.fuel$centers #view cluster centers (mean values of the attributes for each cluster)
```

```
##   fuel_received_units fuel_mmbtu_per_unit sulfur_content_pct ash_content_pct
## 1        -0.137419076           0.8580624          0.5230871       0.7721576
## 2         9.886407956          -1.1775916         -0.6987812      -0.9756003
## 3        -0.004047554          -1.0824292         -0.6602136      -0.9756003
##   fuel_cost_per_mmbtu
## 1          -0.6845684
## 2           0.5898067
## 3           0.8699666
```

#Cluster 1 - does not take a lot, to give a lot, has high sulfur and ash content, so more destructive environmentally, but very cheap. Looks like coal. #Cluster 2 - More of those has been received, but it is not as efficient as coal (cluster 1). Cost is much higher than cluster 1, but not as high as cluster 3. Seems like oil, because not as densely packed, and oil was only 69 rows in the dataset for the summary. #Cluster 3 - Highest cost, least units received. Efficiency, sulfur content, and ash content similar to cluster 2. Looks to be gas.

```
kmeans.fuel$size #to view size of the clusters
```

```
## [1] 422   6 328
```

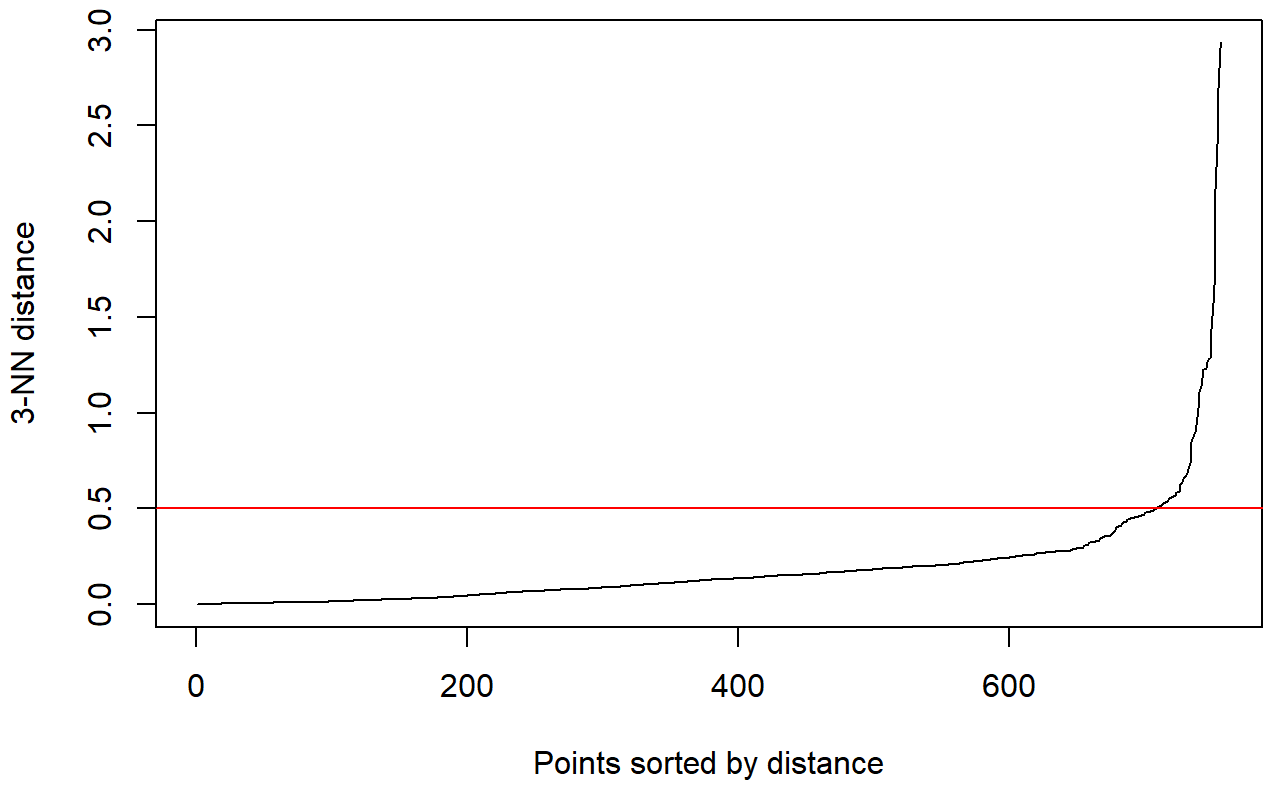
#coal is off by itself, with 422 matching the summary to start; oil and gas sharing some similarities, so the clustering is not as distinct

```
kmeans.fuel$withinss #View within-cluster sum of squares (to evaluate the tightness of the clust
ers)
```

```
## [1] 727.00643  22.05767 449.21681
```

#Cluster 2 has the smallest sum of squares. So, that one is the most compact, with those samples most similar to each other. Cluster 3 is not as compact, and cluster 1 is the least compact. This can also been seem visually in the cluster plot.

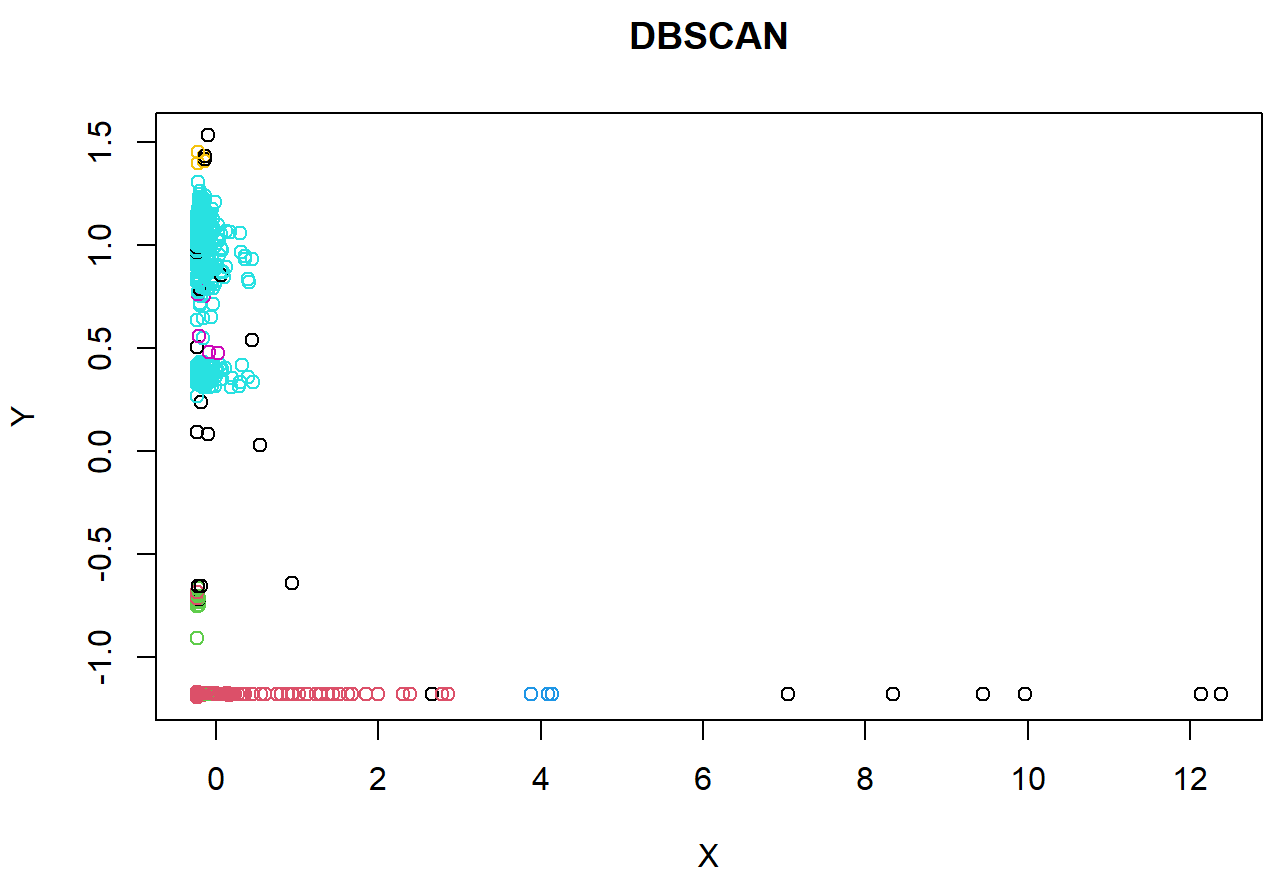
```
kNNdistplot(fuel_data.norm, k=3) #used to figure out the epsilon - in this case it is 0.5 as vis
ualized with the red line; we need this to apply DBSCAN.
abline(h=0.5, col="red")
```




```
epsilon_value <- 0.5 # Adjust based on kNNdistplot output
minPts_value <- 3  # Set based on the silhouette method to prep for k-means above (k=3)

set.seed(123) #for reproducibility
DBscan_fuel <- dbscan::dbscan(fuel_data.norm, eps=epsilon_value, minPts=minPts_value) #applying
DBSCAN
DBscan_fuel #print to show cluster details
```

```
## DBSCAN clustering for 756 objects.
## Parameters: eps = 0.5, minPts = 3
## Using euclidean distances and borderpoints = TRUE
## The clustering contains 6 cluster(s) and 26 noise points.
##
##   0   1   2   3   4   5   6
##  26 254  65   3 398   6   4
##
## Available fields: cluster, eps, minPts, metric, borderPoints
```

```
plot(DBscan_fuel, fuel_data.norm, main="DBSCAN", frame= TRUE, xlab = "X", ylab = "Y") #used to v
isualize the DBSCAN
```




```
DBscan_fuel$cluster # Create a new column in the data to store cluster assignments
```

```
##   [1] 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [75] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 0 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1
## [112] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1
## [149] 1 1 1 1 2 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 0 1 1
## [186] 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1
## [223] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 2 1 1 1 1 1 1 1
## [260] 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [297] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 0 1 2 2 2 2 2 2 2 1 2 1 2 1 2 0 0
## [334] 0 0 0 0 0 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
## [371] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
## [408] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 0 0 4 5 4 4 4 4
## [445] 4 4 5 4 4 5 4 4 4 4 0 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 0 4 4 4 5 4 4 4 4 4
## [482] 4 0 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
## [519] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 0 4 4 4 4 4 4 4 4 4
## [556] 4 4 4 4 0 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
## [593] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
## [630] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
## [667] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
## [704] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
## [741] 4 4 4 4 4 4 4 4 4 6 6 0 6 0 6 0
```

```
fuel_data$cluster <- DBscan_fuel$cluster

# Calculate the means for each cluster (ignoring noise points where cluster == 0)
centroids <- aggregate(. ~ cluster, data = fuel_data[DBscan_fuel$cluster > 0,], FUN = mean)

# View centroids
print(centroids)
```

```
##   cluster fuel_type_code_pudl fuel_received_units fuel_mmbtu_per_unit
## 1       1            2.019685           86878.516            1.114453
## 2       2            2.923077            3173.938            5.388800
## 3       3            2.000000         1634654.667            1.026667
## 4       4            1.000000           35952.191           22.345593
## 5       5            1.000000           35204.000           20.107667
## 6       6            1.000000           17840.750           28.245000
##   sulfur_content_pct ash_content_pct fuel_cost_per_mmbtu
## 1        0.002401575        0.000000            8.083496
## 2        0.127538462        0.000000           19.645185
## 3        0.000000000        0.000000            8.450000
## 4        1.183165829        8.793216            2.148947
## 5        0.863333333       19.200000            2.007333
## 6        5.597500000        0.540000            2.160500
```

#since we know there are 3 groupings of fuel types, this method is not the most helpful as it has separated the data into 6 clusters. This will not meet the need.

```
df <- fuel_data.norm #AGNES method for hierarchical clustering
hc_singleA <- agnes(df, method = "single")
hc_completeA <- agnes(df, method = "complete")
hc_averageA <- agnes(df, method = "average")
print(hc_singleA$ac)
```

```
## [1] 0.9628749
```

```
print(hc_completeA$ac)
```

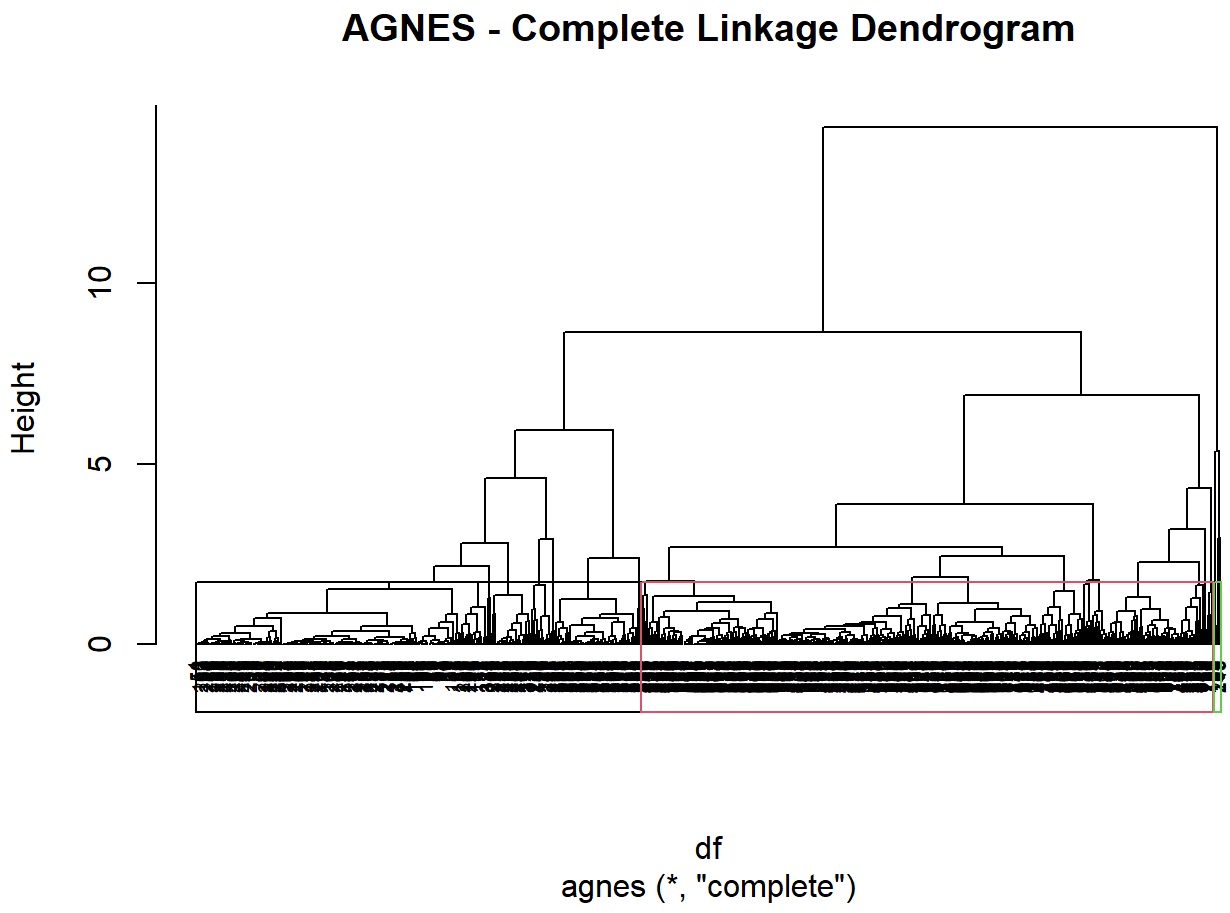
```
## [1] 0.9911642
```

```
print(hc_averageA$ac)
```

```
## [1] 0.9881404
```

#complete has the highest coefficient at 99.11, so utilizing that moving forward this hierarchical clustering

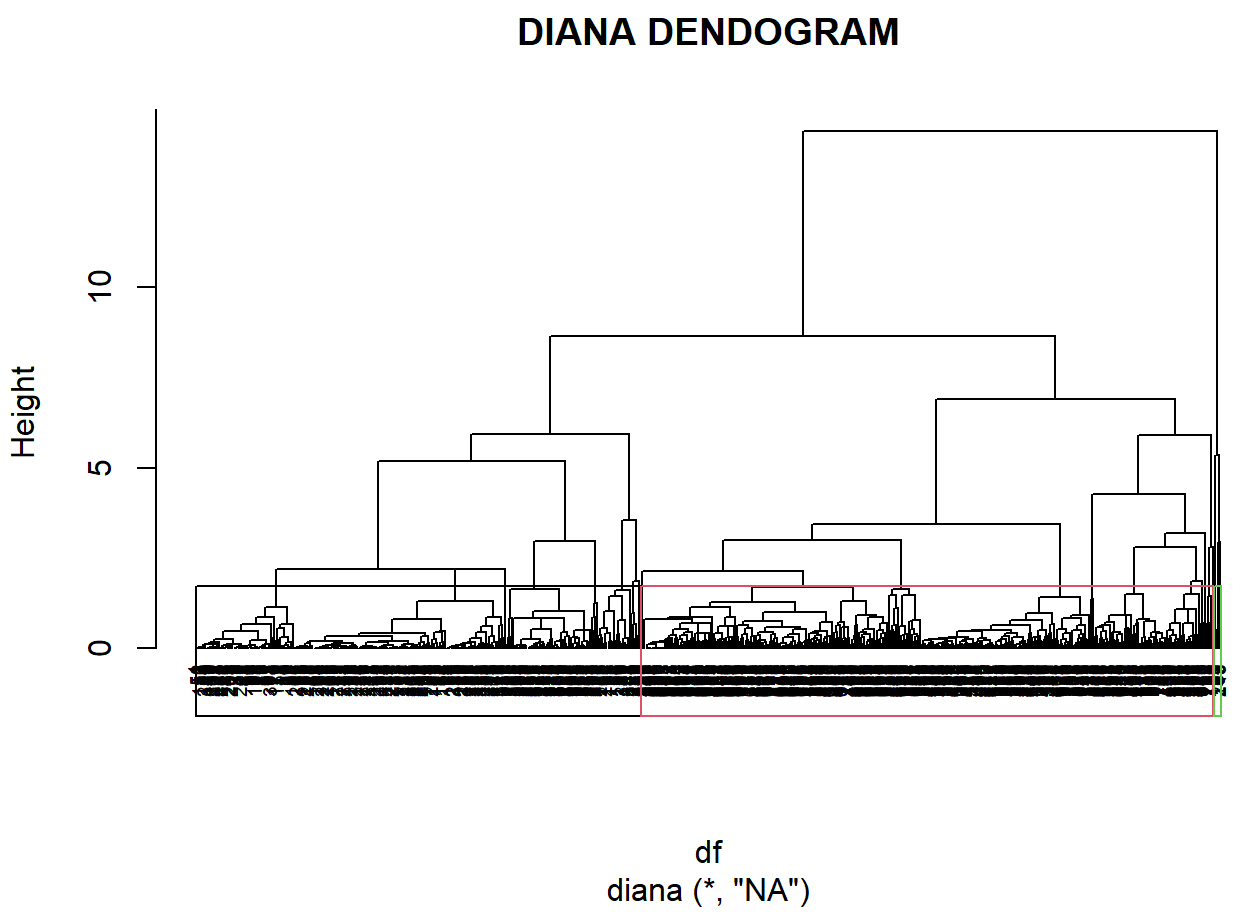
```
pltree(hc_completeA, cex=0.6, hang=-1, main = "AGNES - Complete Linkage Dendrogram") #dendrogram for AGNES - Complete linkage chosen because had higher coefficient with AGNES method at 99.11%
rect.hclust(hc_completeA, k = 3, border = 1:5) #put border around the clusters
```



```
hc_diana <- diana(df) #DIANA method for hierarchical clustering and print coefficient
hc_diana$dc
```

```
## [1] 0.9896694
```

```
pltree(hc_diana, cex=0.6,hang=-1,main="DIANA DENDOGRAM") #dendrogram for DIANA method and provid
e border around the clusters
rect.hclust(hc_diana, k=3, border=1:5)
```




```
silhouette_agnes <- silhouette(cutree(hc_completeA, k = 3), dist(fuel_data.norm)) #silhouette sc
ore for AGNES method
mean(silhouette_agnes[, 3])
```

```
## [1] 0.6014165
```

```
silhouette_kmeans <- silhouette(kmeans.fuel$cluster, dist(fuel_data.norm)) #silhouette score for
K-Means
mean(silhouette_kmeans[, 3])
```

```
## [1] 0.6014165
```

#The coefficient for AGNES method was higher than DIANA, at 99.11 vs. 98.97%. However, when looking at the silhouette score for accuracy, both AGNES and K-Means have the same silhouette score at .6014. Visually, K-Means is more useful than AGNES. There are too many values (rows of data) for hierarchical clustering to be useful. Additionally, DBSCAN was excluded just from the fact that there were 6 clusters, and we know there are 3 groups (coal, oil, and gas).

#B) Classification

```
set.seed(123)
Index_Train <- createDataPartition(fuel_data$fuel_type_code_pudl, p=.7, list = FALSE) # separati
ng data into Training and Testing data for classification modeling
Fuel_Train <- fuel_data[Index_Train,]
Fuel_Test <-  fuel_data[-Index_Train,]
nrow(Fuel_Train) #how many rows of data for the training and testing datasets
```

```
## [1] 531
```

```
nrow(Fuel_Test)
```

```
## [1] 225
```

```
Fuel_Train.Norm <- Fuel_Train[,-1] #remove the categorical variable to do classification
Fuel_Test.Norm <- Fuel_Test[,-1]

norm.values <- preProcess(Fuel_Train[, -1], method = c("center", "scale"))
Fuel_Train.Norm <- predict(norm.values, Fuel_Train[, -1])
Fuel_Test.Norm <- predict(norm.values, Fuel_Test[, -1])
```

```
set.seed(123)
accuracy.df <- data.frame(k = seq(1, 15, 1), overallaccuracy = rep(0, 15))

for(i in 1:15) {
knn.pred <- knn(train = Fuel_Train.Norm,
                  test = Fuel_Test.Norm,
                  cl = Fuel_Train$fuel_type_code_pudl,
                  k = i)

cm <- confusionMatrix(knn.pred, as.factor(Fuel_Test$fuel_type_code_pudl))

accuracy.df$overallaccuracy[i] <- cm$overall['Accuracy']
}

print(accuracy.df)
```

```
##     k overallaccuracy
## 1   1       0.9955556
## 2   2       0.9955556
## 3   3       0.9955556
## 4   4       0.9955556
## 5   5       0.9955556
## 6   6       0.9955556
## 7   7       0.9955556
## 8   8       0.9911111
## 9   9       0.9911111
## 10 10       0.9911111
## 11 11       0.9911111
## 12 12       0.9911111
## 13 13       0.9911111
## 14 14       0.9911111
## 15 15       0.9911111
```

#K 1-7 all have the same accuracy of .9956. A k of 1 would lead to overfitting.Choosing a k of 5 as a balance between bias and variance.

```
knn.pred <- knn(train = Fuel_Train.Norm,
                test = Fuel_Test.Norm,
                cl = Fuel_Train$fuel_type_code_pudl,
                k = 5)
summary(knn.pred)
```

```
## coal  gas  oil
##  126   78   21
```

```
fuel_counts <- table(knn.pred)
fuel_percentages <- (fuel_counts/sum(fuel_counts))*100 #calculating percentage of each fuel type in the prediction dataset using K-NN method
fuel_percentages
```

```
## knn.pred
##      coal       gas       oil
## 56.000000 34.666667  9.333333
```

```
original_counts <- c(422, 265,69)
original_percentages <- (original_counts/sum(original_counts))*100 #calculating percentage of each fuel type in the original dataset
original_percentages
```

```
## [1] 55.820106 35.052910  9.126984
```

#The prediction is spot on for the test data with regards to choosing the appropriate fuel type as the percentages are pretty similar to the original dataset.

```
Actual <- Fuel_Test$fuel_type_code_pudl  #actual values in the testing dataset
Predicted_KNN <- knn.pred #predicted values for the fuel types using K-NN method

# Create a confusion matrix
confusion_matrix_KNN <- table(actual = Actual, predicted = Predicted_KNN) #confusion matrix to v
iew actual vs. predicted value and see accuracy
print(confusion_matrix_KNN)
```

```
##       predicted
## actual coal gas oil
##   coal  126   0   0
##   gas     0  78   1
##   oil     0   0  20
```

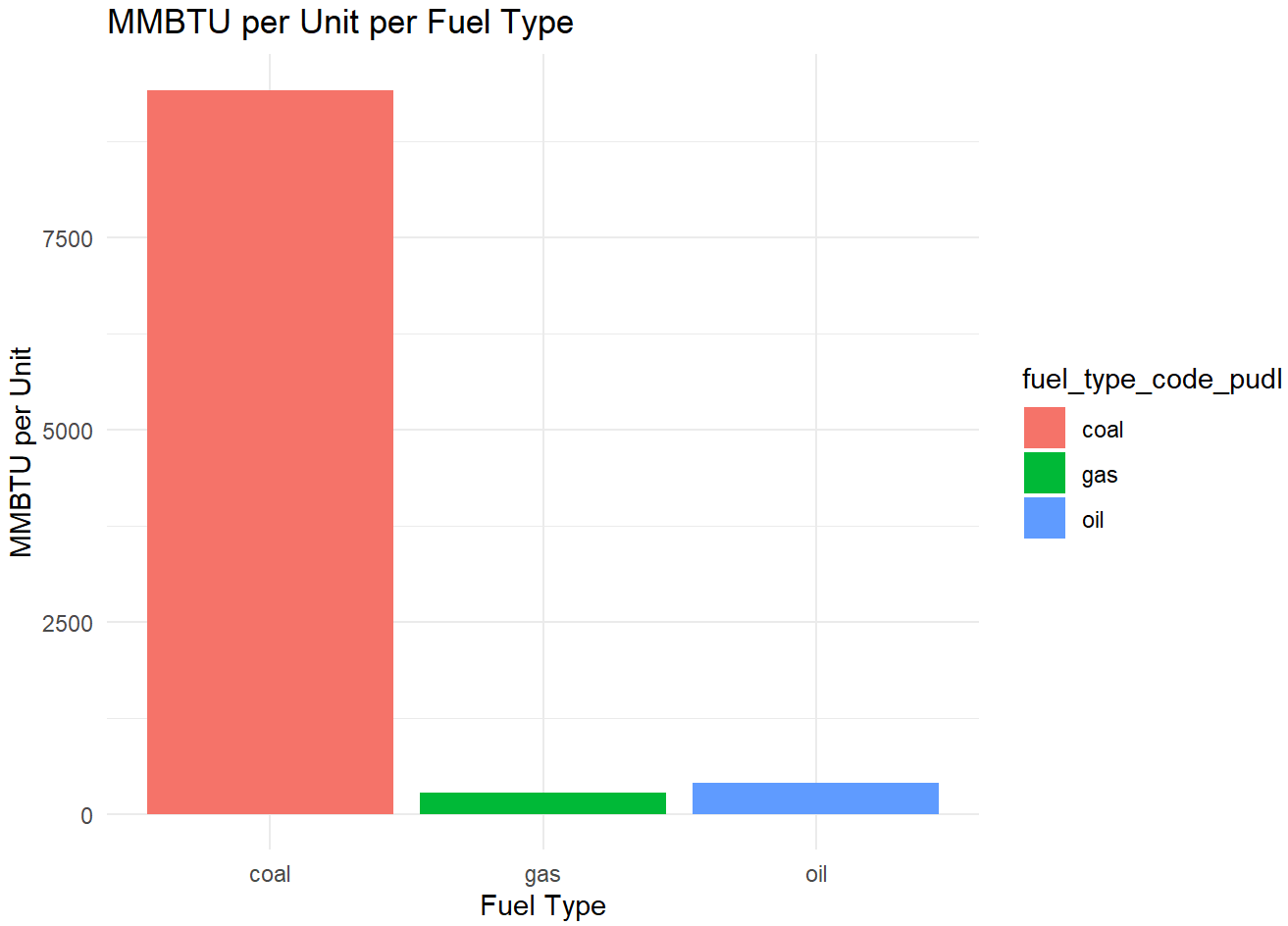
#Confusion matrix above shows accuracy, with misclassification of only 1 value, where gas was incorrectly predicted to be oil. 1 misclassifcation out of 255 values = 99.61% accuracy with K-NN.

```
set.seed(123)  # Set seed for reproducibility
train.index <- sample(1:nrow(fuel_data), 0.7 * nrow(fuel_data))  # Use numeric indices to split
into training and testing data
test.index <- setdiff(1:nrow(fuel_data), train.index)  # Get remaining indices for testing

train.df <- fuel_data[train.index, ] # Create training and testing datasets
test.df <- fuel_data[test.index, ]

nb_model <- naiveBayes(fuel_type_code_pudl ~ fuel_mmbtu_per_unit + ash_content_pct + fuel_cost_p
er_mmbtu,
                       data = train.df) #build Naive Bayes model using the training data

test_pred <- predict(nb_model, newdata = test.df) #generate predictions on the test data

Actual_NB <- test.df$fuel_type_code_pudl  # Actual fuel types in the test set
Predicted_NB <- test_pred  # Predicted fuel types from the Naive Bayes model

confusion_matrix_NB <- table(actual = Actual_NB, predicted = Predicted_NB) #create and print con
fusion matrix to see model accuracy
print(confusion_matrix_NB)
```

```
##       predicted
## actual coal gas oil
##   coal  126   1   0
##   gas     0  75   0
##   oil     0   3  22
```

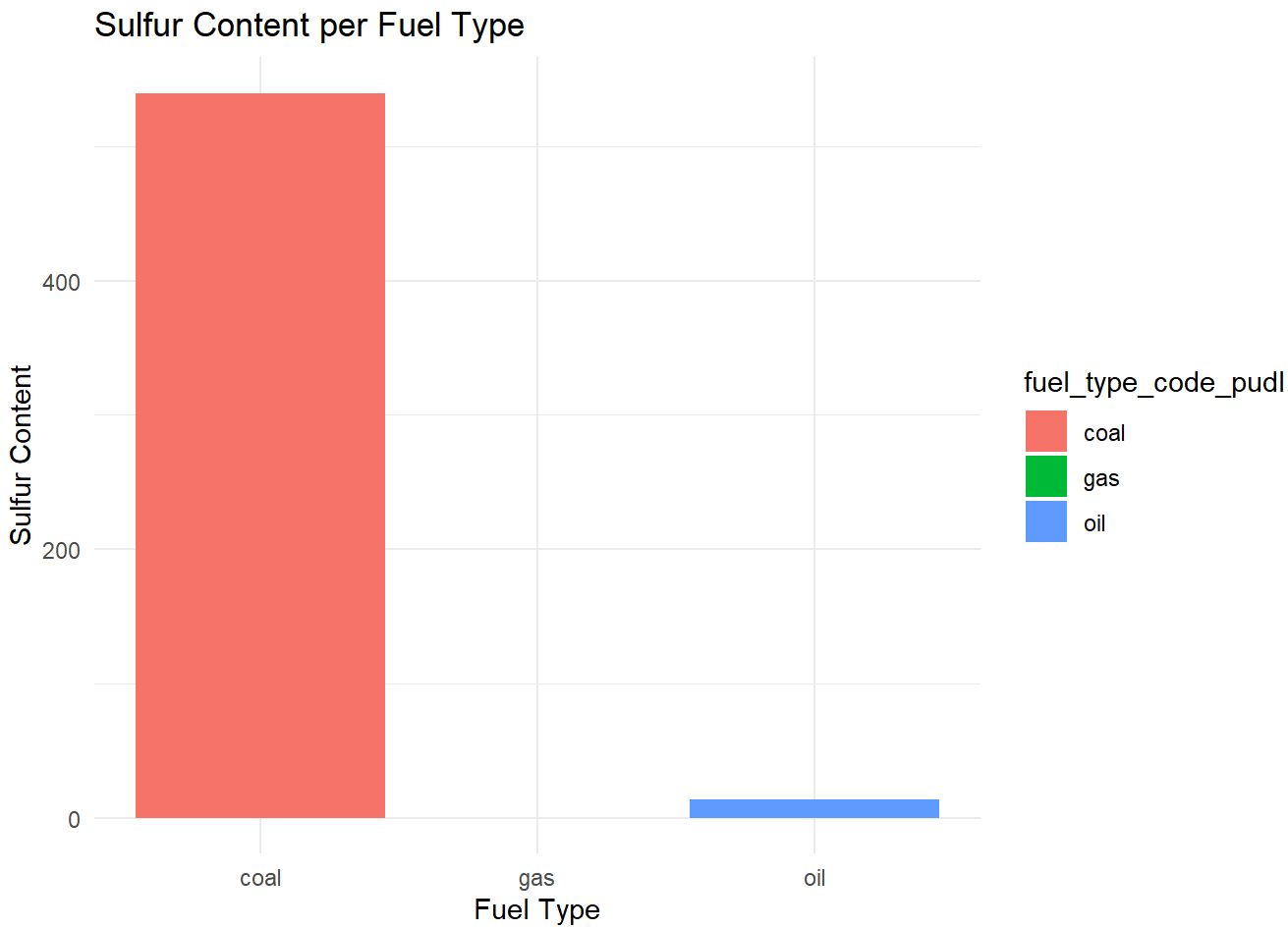
#Naive Bayes was not as accurate as K-NN. All of the gas was correctly identified. However, 1 coal value was misidentified as gas and 3 oil values were misidentified as gas. This model had an accuracy of 98.24%. While this model is still accurate, it is just not as accurate as the K-NN method employed previously.

```
ggplot(fuel_data, aes(x = fuel_type_code_pudl, y = fuel_mmbtu_per_unit, fill = fuel_type_code_pu
dl)) +
  geom_bar(stat = "identity") +
  labs(
    title = "MMBTU per Unit per Fuel Type",
    x = "Fuel Type",
    y = "MMBTU per Unit"
  ) +
  theme_minimal() #bar plot for MMBTU per unit
```



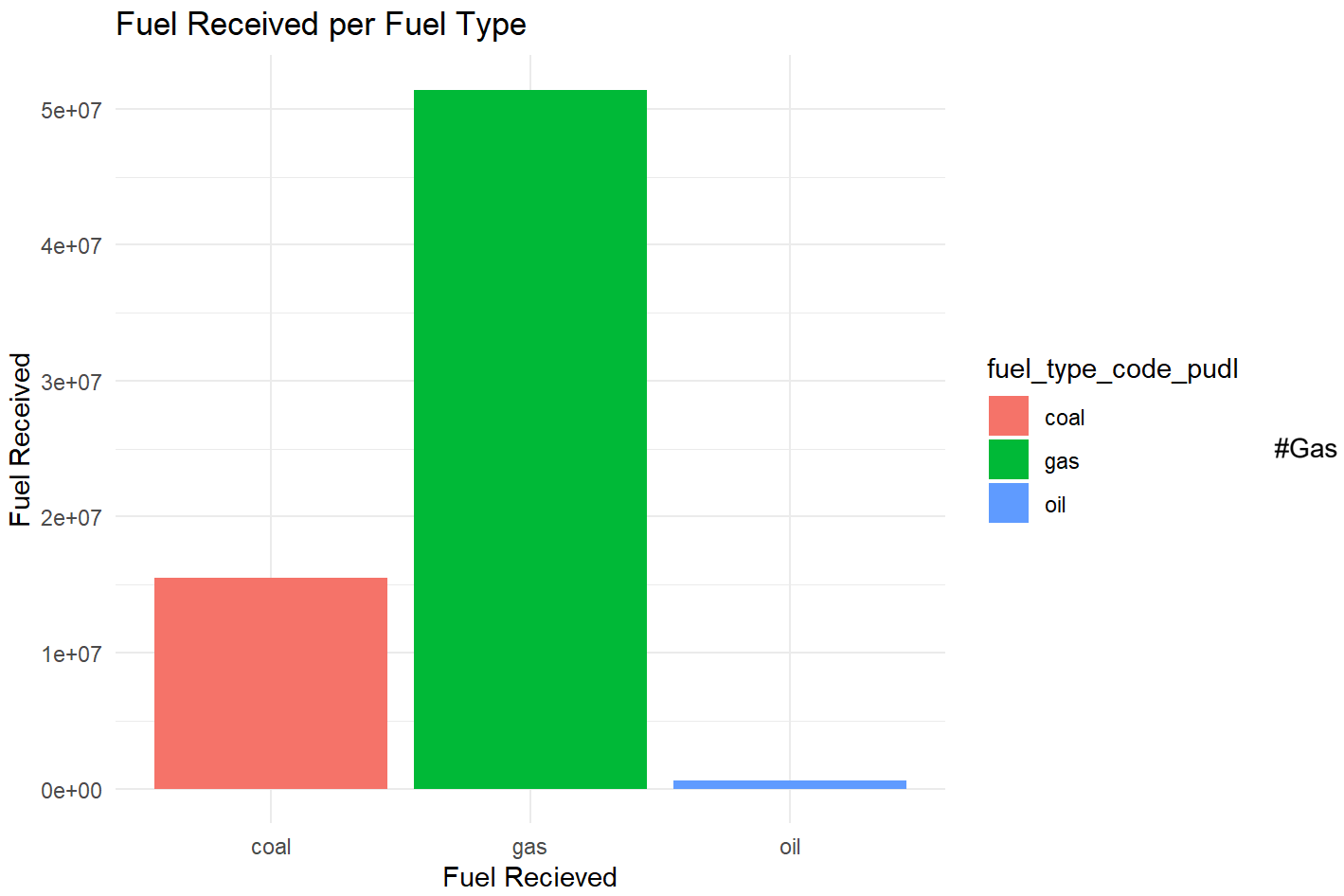
#Coal is drastically higher in MMBTU per unit than the other fuel types. Oil is much lower, but marginally higher than gas.

```
ggplot(fuel_data, aes(x = fuel_type_code_pudl, y =sulfur_content_pct, fill = fuel_type_code_pud
l)) +
  geom_bar(stat = "identity") +
  labs(
    title = "Sulfur Content per Fuel Type",
    x = "Fuel Type",
    y = "Sulfur Content"
  ) +
  theme_minimal() #bar plot for sulfur content
```



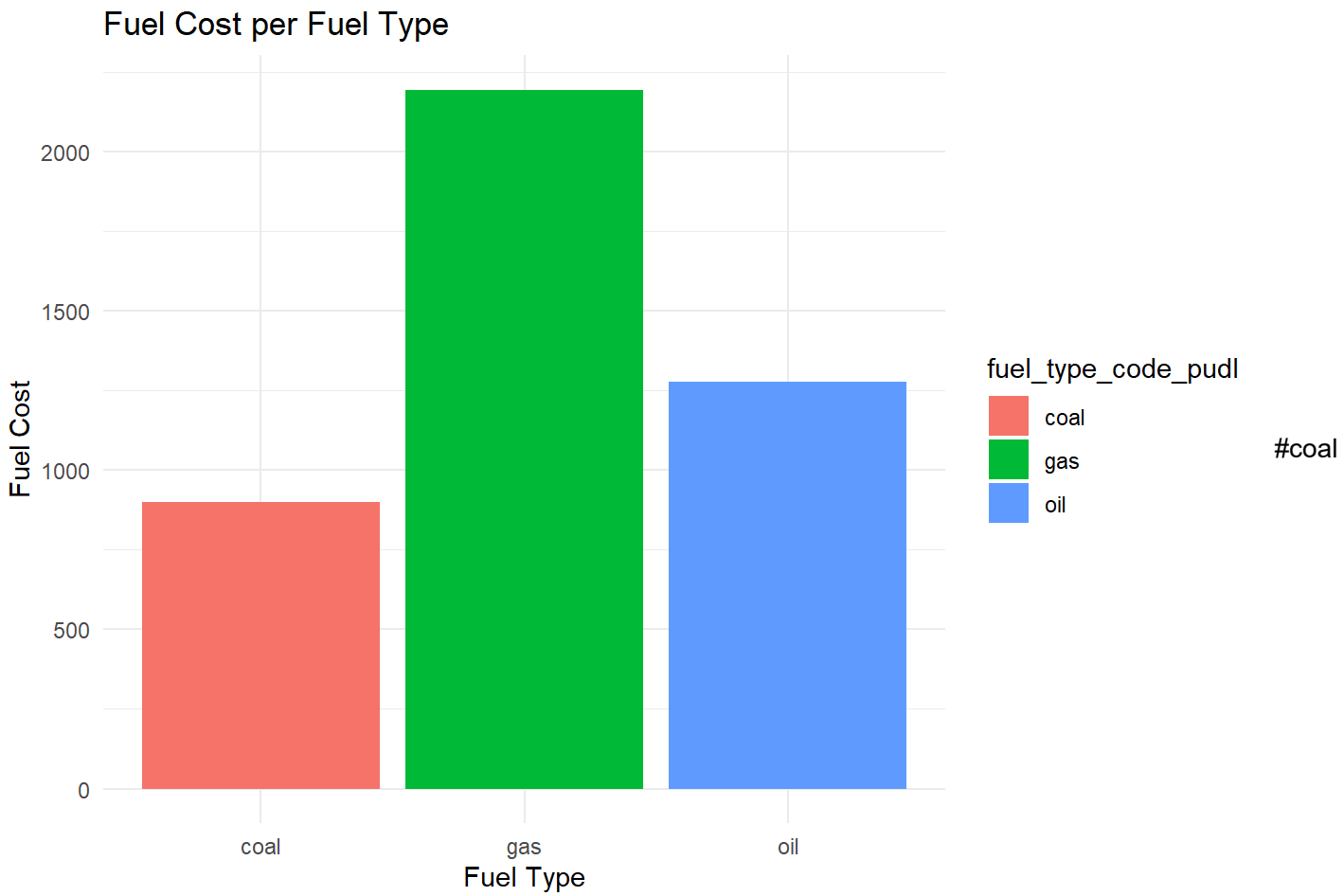
#Sulfur content drastically higher in coal. Oil is slightly higher than gas in sulfur content.

```
ggplot(fuel_data, aes(x = fuel_type_code_pudl, y = fuel_received_units, fill = fuel_type_code_pu
dl)) +
  geom_bar(stat = "identity") +
  labs(
    title = "Fuel Received per Fuel Type",
    x = "Fuel Recieved",
    y = "Fuel Received"
  ) +
  theme_minimal()#bar plot for fuel received
```



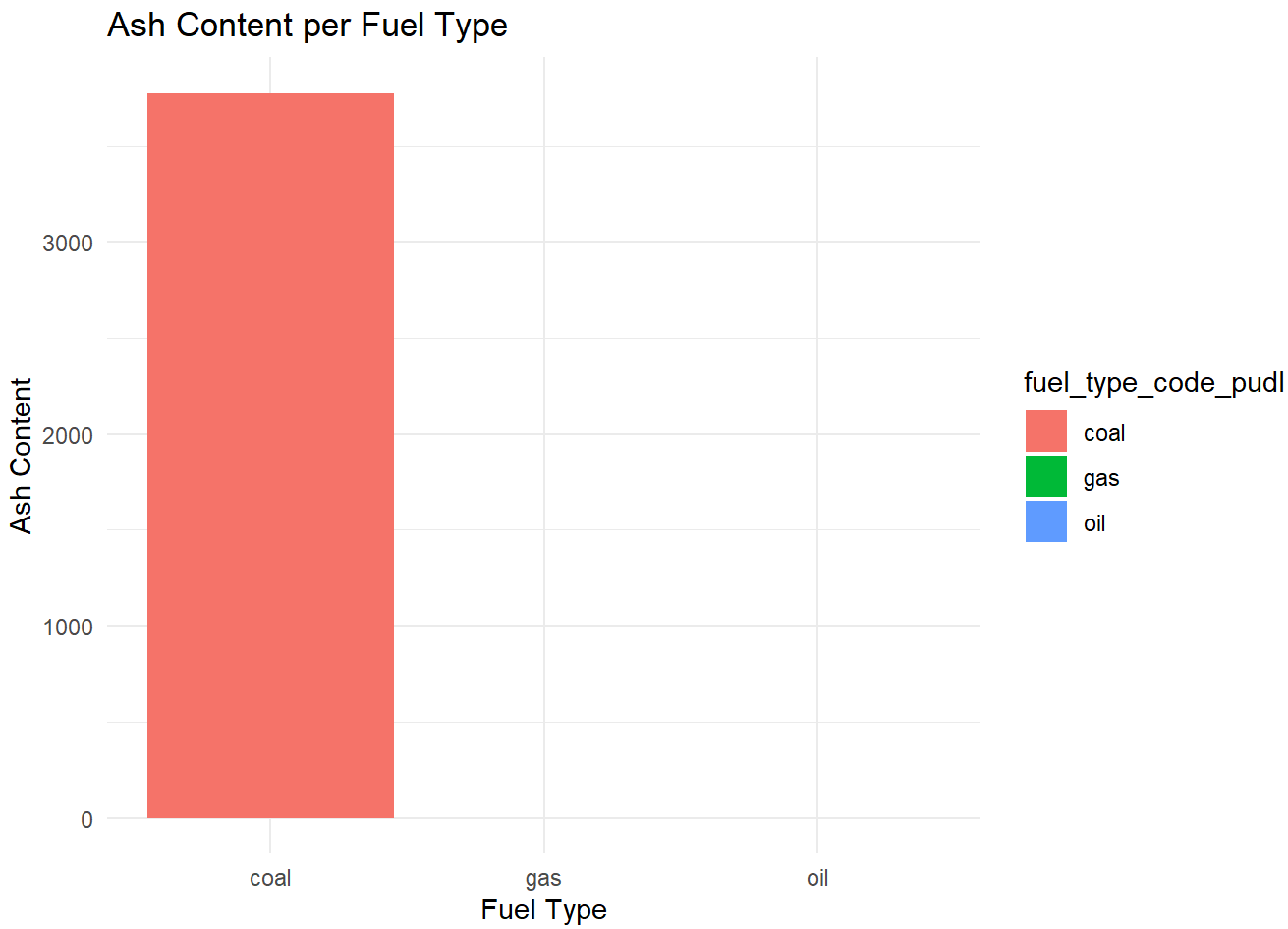
is the fuel type most received. Coal is second most, with oil much smaller.

```
ggplot(fuel_data, aes(x = fuel_type_code_pudl, y = fuel_cost_per_mmbtu, fill = fuel_type_code_pu
dl)) +
  geom_bar(stat = "identity") +
  labs(
    title = "Fuel Cost per Fuel Type",
    x = "Fuel Type",
    y = "Fuel Cost"
  ) +
  theme_minimal() #bar plot for fuel cost
```



is the cheapest, then oil, and gas is almost twice as expensive as oil.

```
ggplot(fuel_data, aes(x = fuel_type_code_pudl, y = ash_content_pct, fill = fuel_type_code_pudl))
+
  geom_bar(stat = "identity") +
  labs(
    title = "Ash Content per Fuel Type",
    x = "Fuel Type",
    y = "Ash Content"
  ) +
  theme_minimal() #bar plot for ash content
```


Ash Content per Fuel Type
Ash Content
3000
2000
1000
0
coal
gas
oil
Fuel Type
fuel_type_code_pudl
coal
gas
oil