Final Exam

Pranay Kumar Kodeboyina

2023-05-07

```
# importing the Necessary libraries
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(caret)
```

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

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

```
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve
3WBa
```

```
library(leaps)
library(dbscan)
```

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

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

```
library(esquisse)
library(readr)
```

```
#Extracting the current working directory
getwd()
```

```
## [1] "/Users/kodeboyina/Documents/Kent State/Sem1/Fundamentals of ML/Final Exam"
```

```
#Loading PUDL csv data Import the data set into R
Fuel <- read.csv("data/fuel_receipts_costs.csv", header = TRUE, sep = ",", stringsAsF
actors = TRUE)

#Observing the first 10 observations of the data set
head(Fuel, n=10L)
```

	ro…	plant_id_eia	report_date	contract_type_code	contract_expiration_date	ene
	<int>	<int>	<fct>	<fct>	<fct>	<fct
1	1	3	2008-01-01	C	2008-04-01	BIT
2	2	3	2008-01-01	C	2008-04-01	BIT
3	3	3	2008-01-01	C		NG
4	4	7	2008-01-01	C	2015-12-01	BIT
5	5	7	2008-01-01	S	2008-11-01	BIT
6	6	7	2008-01-01	S	2008-01-01	BIT
7	7	7	2008-01-01	S		NG
8	8	8	2008-01-01	C	2008-12-01	BIT
9	9	8	2008-01-01	C	2008-03-01	BIT
10	10	8	2008-01-01	C	2010-12-01	BIT

1-10 of 10 rows | 1-7 of 24 columns

```
#Observing the data sets that have missing values more than 50 percent and omitting t
hose columns in the analysis

# calculate the percentage of missing values in each column
missing_pct <- colMeans(is.na(Fuel)) * 100

# create a data frame with the missing percentages as a single column
missing_df <- data.frame(percent_missing = missing_pct)

# print the resulting data frame
print(missing_df)
```

```
##                                               percent_missing
## rowid                                                 0.00000
## plant_id_eia                                          0.00000
## report_date                                           0.00000
## contract_type_code                                    0.00000
## contract_expiration_date                              0.00000
## energy_source_code                                    0.00000
## fuel_type_code_pudl                                   0.00000
## fuel_group_code                                       0.00000
## mine_id_pudl                                         64.40506
## supplier_name                                         0.00000
## fuel_received_units                                   0.00000
## fuel_mmbtu_per_unit                                   0.00000
## sulfur_content_pct                                    0.00000
## ash_content_pct                                       0.00000
## mercury_content_ppm                                  47.56805
## fuel_cost_per_mmbtu                                  32.90369
## primary_transportation_mode_code                      0.00000
## secondary_transportation_mode_code                    0.00000
## natural_gas_transport_code                            0.00000
## natural_gas_delivery_contract_type_code               0.00000
## moisture_content_pct                                 84.88639
## chlorine_content_ppm                                 84.88639
## data_maturity                                         0.00000
```

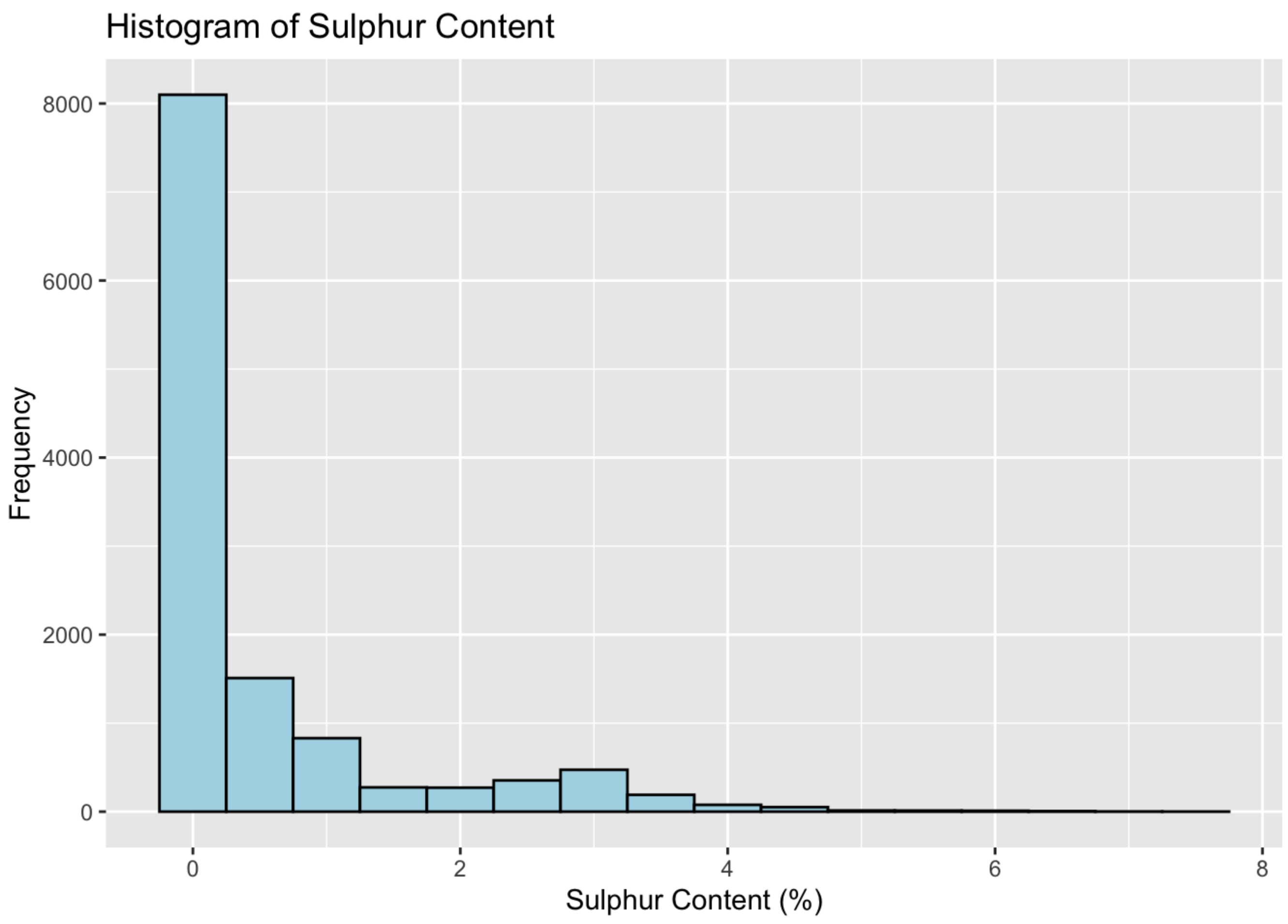
```
#Upon observing the data we can see that the following columns has missing values mor
e than 50% - mine_id_pudl, moisture_content_pct, chlorine_content_ppm
```

```
# set the random seed for reproducibility
set.seed(8627)

# create a new data set without the specified columns and considering the columns wit
hout the specified columns
Fuel_data_ALL <- Fuel[, -c(1,2,3,5,9,10,15,18,19,20,21,22,23)]
head(Fuel_data_ALL)
```

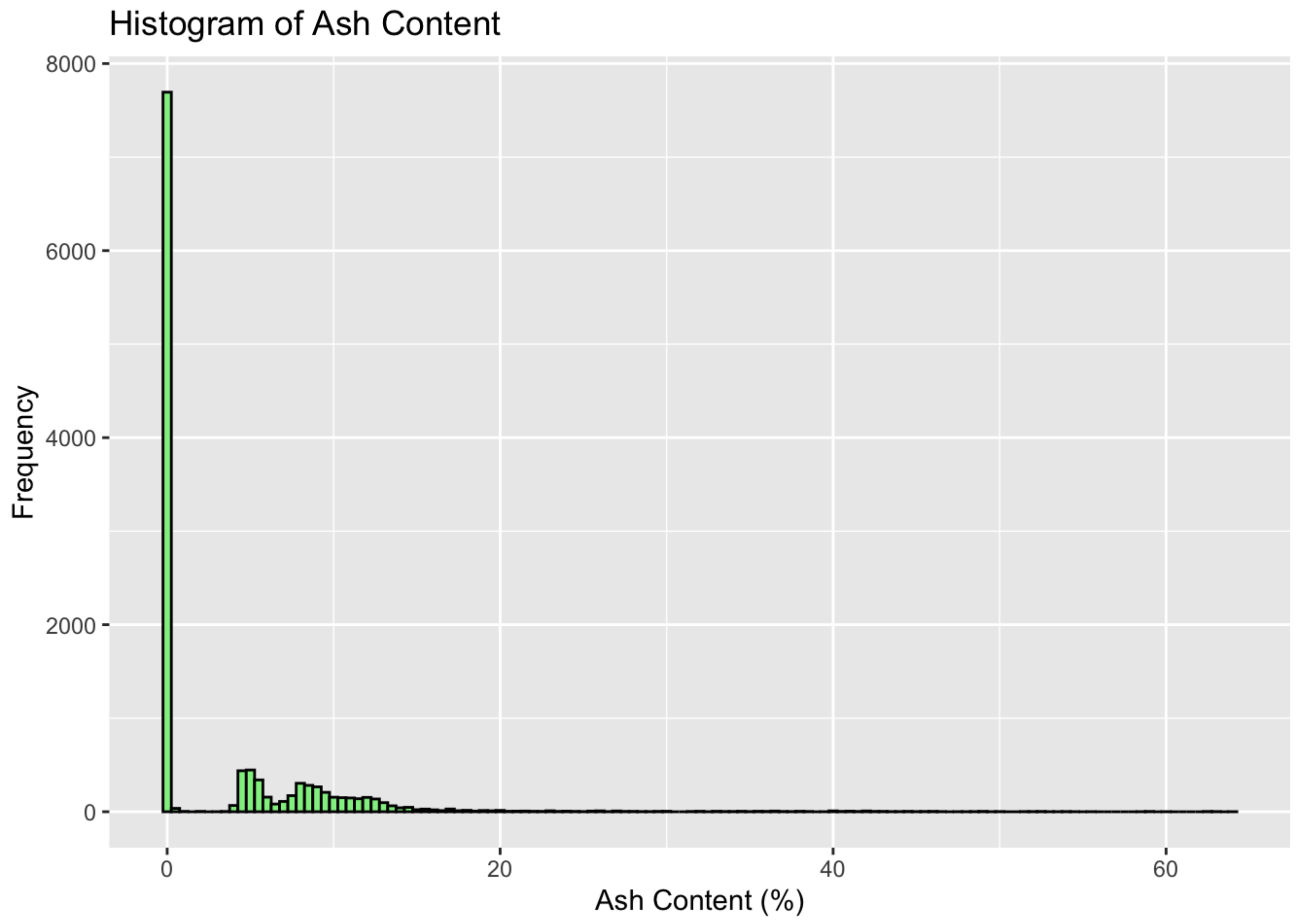
	contract_type_code <fct>	energy_source_code <fct>	fuel_type_code_pudl <fct>	fuel_group_code <fct>
1	C	BIT	coal	coal
2	C	BIT	coal	coal
3	C	NG	gas	natural_gas
4	C	BIT	coal	coal
5	S	BIT	coal	coal
6	S	BIT	coal	coal

6 rows | 1-6 of 11 columns

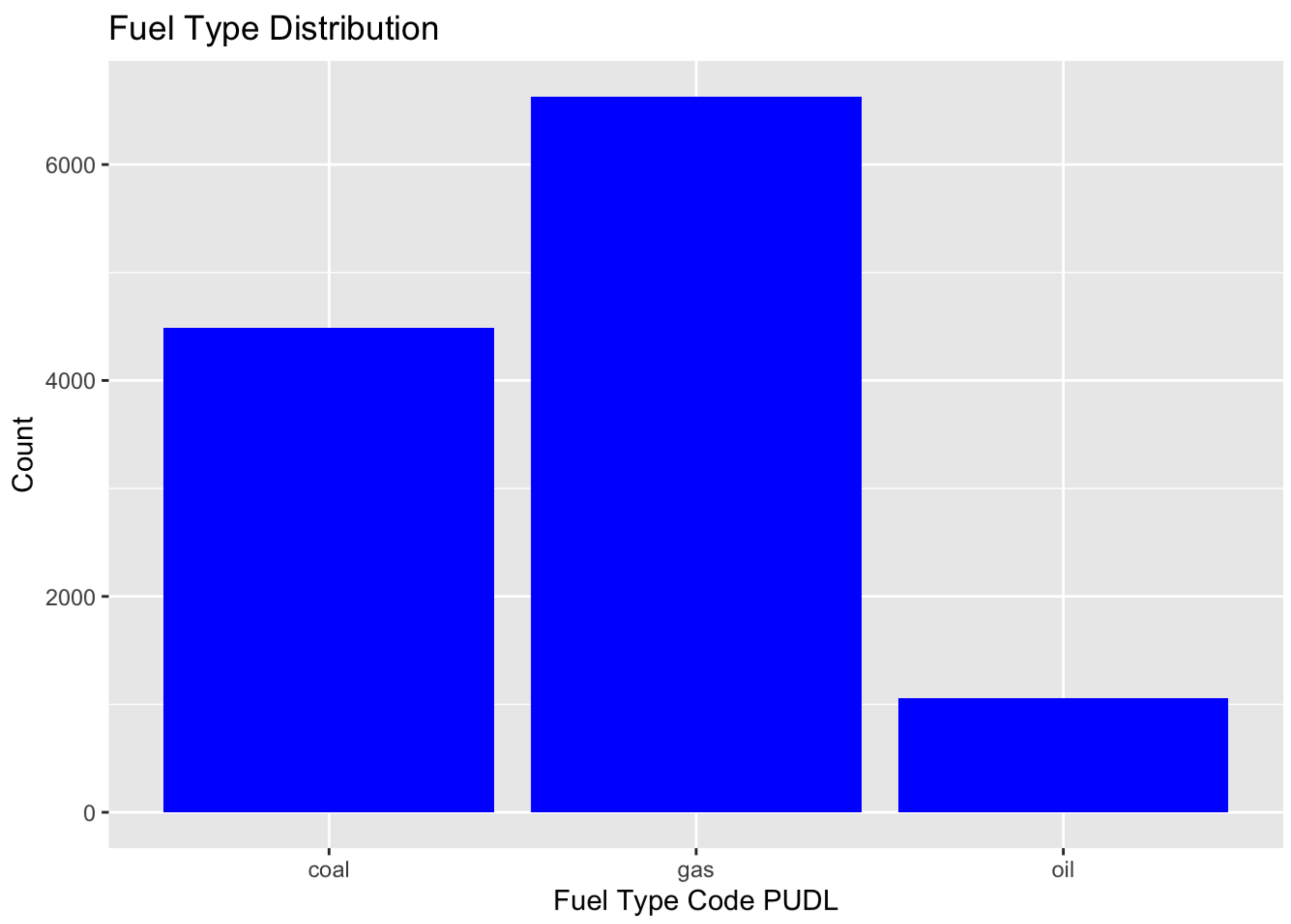
```
#head(Fuel_data_ALL)

# calculate the number of rows to select (2% of the total rows)
nrows <- round(nrow(Fuel_data_ALL) * 0.02)

# randomly select the rows
idx <- sample(nrow(Fuel_data_ALL), nrows, replace = FALSE)

# create the new data set with the randomly selected rows
Fuel_data <- Fuel_data_ALL[idx, ]

#View(Fuel_data)

#Exploratory data Analysis
# Create histograms of sulphur and ash content
ggplot(Fuel_data, aes(x=sulfur_content_pct)) +
  geom_histogram(binwidth=0.5, color="black", fill="lightblue") +
  labs(x="Sulphur Content (%)", y="Frequency", title="Histogram of Sulphur Content")
```



```
ggplot(Fuel_data, aes(x=ash_content_pct)) +
  geom_histogram(binwidth=0.5, color="black", fill="lightgreen") +
  labs(x="Ash Content (%)", y="Frequency", title="Histogram of Ash Content")
```



```
# create a bar graph of the fuel_type_code_pudl categories
ggplot(Fuel_data, aes(x = fuel_type_code_pudl)) +
  geom_bar(fill = "blue") +
  labs(title = "Fuel Type Distribution",
       x = "Fuel Type Code PUDL",
       y = "Count")
```



```
# Create dummy variables for fuel_type_code_pudl to create categorical in to numeric
variables
Fuel_data <- Fuel_data %>%
  mutate(fuel_type_gas = ifelse(fuel_type_code_pudl == "gas", 1, 0),
         fuel_type_coal = ifelse(fuel_type_code_pudl == "coal", 1, 0),
         fuel_type_oil = ifelse(fuel_type_code_pudl == "oil", 1, 0))

# Remove the original fuel_type_code_pudl column
Fuel_data <- select(Fuel_data, -fuel_type_code_pudl)


#We can finding the missing values in fuel_cost_per_mmbtu data and replacing them wit
h the median values for the analysis

# Calculate the median value of fuel_cost_per_mmbtu
median_value <- median(Fuel_data$fuel_cost_per_mmbtu, na.rm = TRUE)

# Replace missing values with median
Fuel_data$fuel_cost_per_mmbtu[is.na(Fuel_data$fuel_cost_per_mmbtu)] <- median_value
```

```r
# Generate a vector of random indices for the data frame and splitiing 75% of data
train_idx <- sample(nrow(Fuel_data), round(nrow(Fuel_data) * 0.75), replace = FALSE)

# Create the training set
train_data <- Fuel_data[train_idx, ]

# Create the test set by excluding the training set indices
test_data <- Fuel_data[-train_idx, ]

nrow(train_data)
```

```
## [1] 9128
```

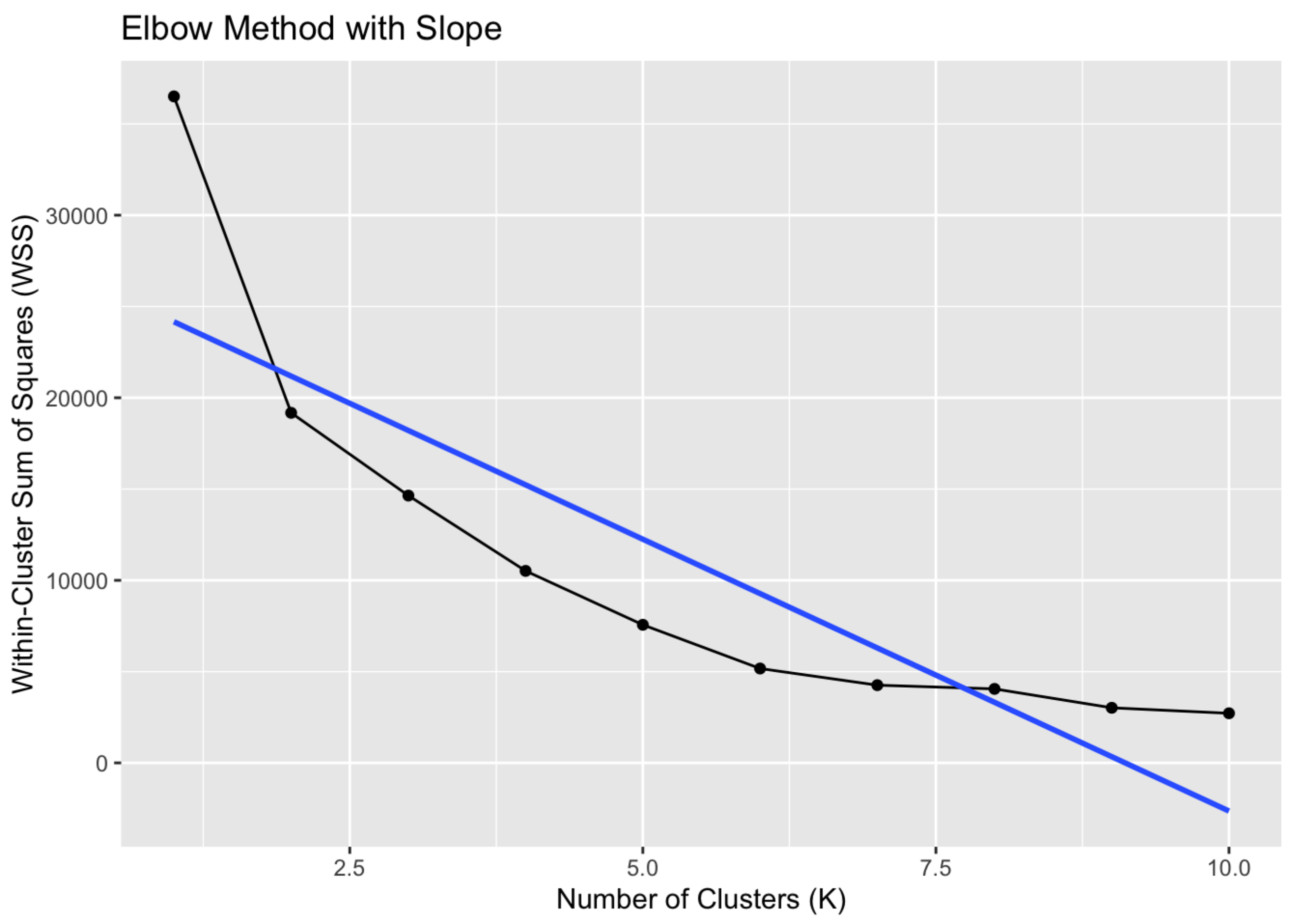
```r
nrow(test_data)
```

```
## [1] 3043
```

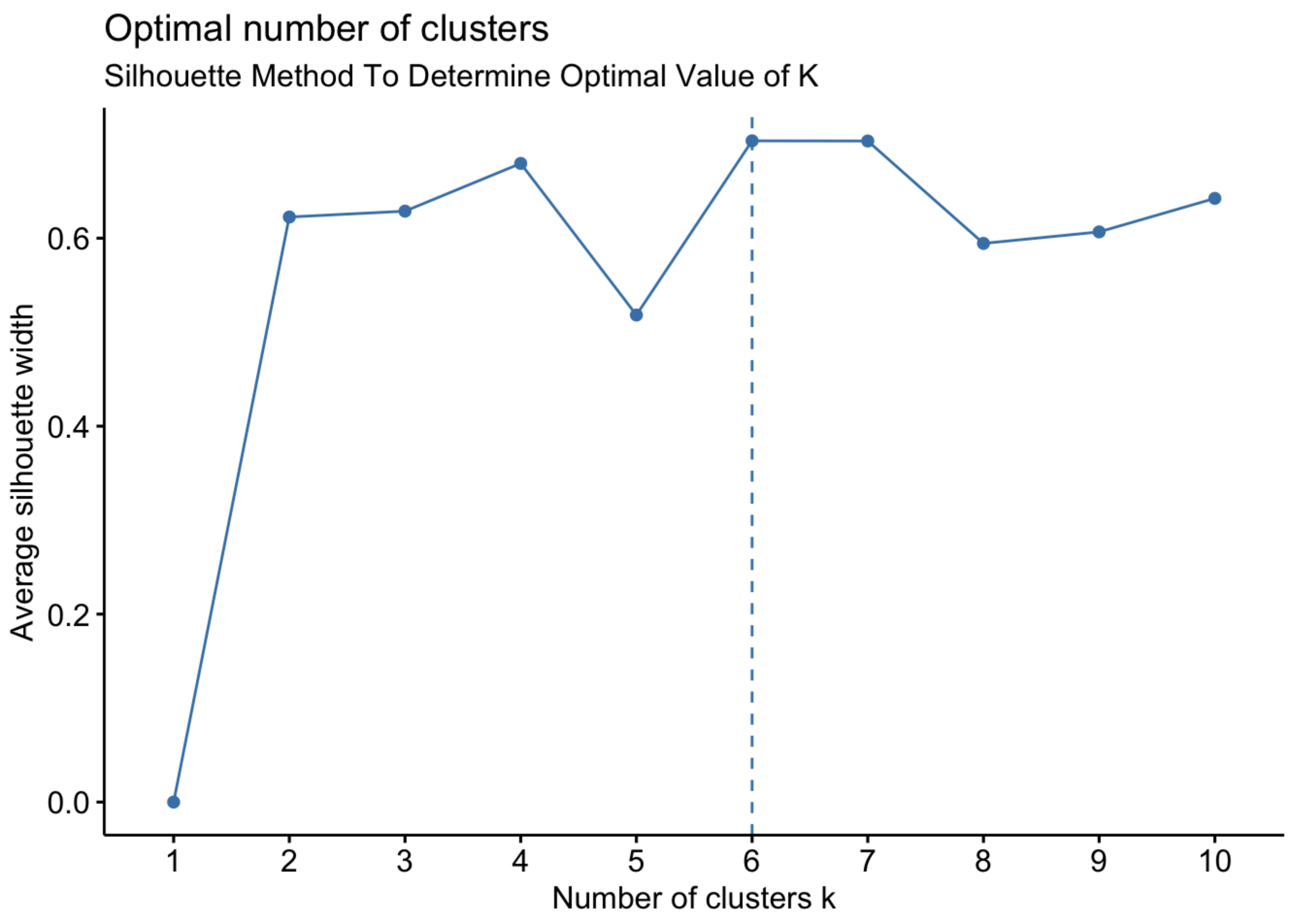
```r
library(caret)

# Create a preprocessing object using the train_data dataset
preproc_obj <- preProcess(train_data[, 4:7], method = c("center", "scale"))

# Use the preprocessing object to normalize the train_data and test_data datasets
train_data_norm <- predict(preproc_obj, train_data)
test_data_norm <- predict(preproc_obj, test_data)



#View(train_data_norm)
```

```
#We have divided the data in to Train and Test set and we are finding Optimal values
of K using Gap stat method and Silhouette method

#In order determine the optimal value of k, employing different methods to determine
k
library(cluster)
library(ggplot2)

wss <- c()
for (i in 1:10) {
  kmeans_fit <- kmeans(train_data_norm[, 4:7], i, nstart = 25)
  wss[i] <- kmeans_fit$tot.withinss
}

elbow_df <- data.frame(K = 1:10, WSS = wss)

ggplot(elbow_df, aes(x = K, y = WSS)) +
  geom_line() +
  geom_point() +
  labs(x = "Number of Clusters (K)", y = "Within-Cluster Sum of Squares (WSS)") +
  ggtitle("Elbow Method with Slope") +
  geom_smooth(method = "lm", se = FALSE)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



```
fviz_nbclust(train_data_norm[, 4:7], kmeans, method = "silhouette") + labs(subtitle =
"Silhouette Method To Determine Optimal Value of K")
```



```
#From the above method we can see that using Elbow method the value of K is 5 and the
value of K is 5 with the silhouette method. I am choosing K value of 5 to cluster the
data as it has produced clusters with clear gap between each other.
```

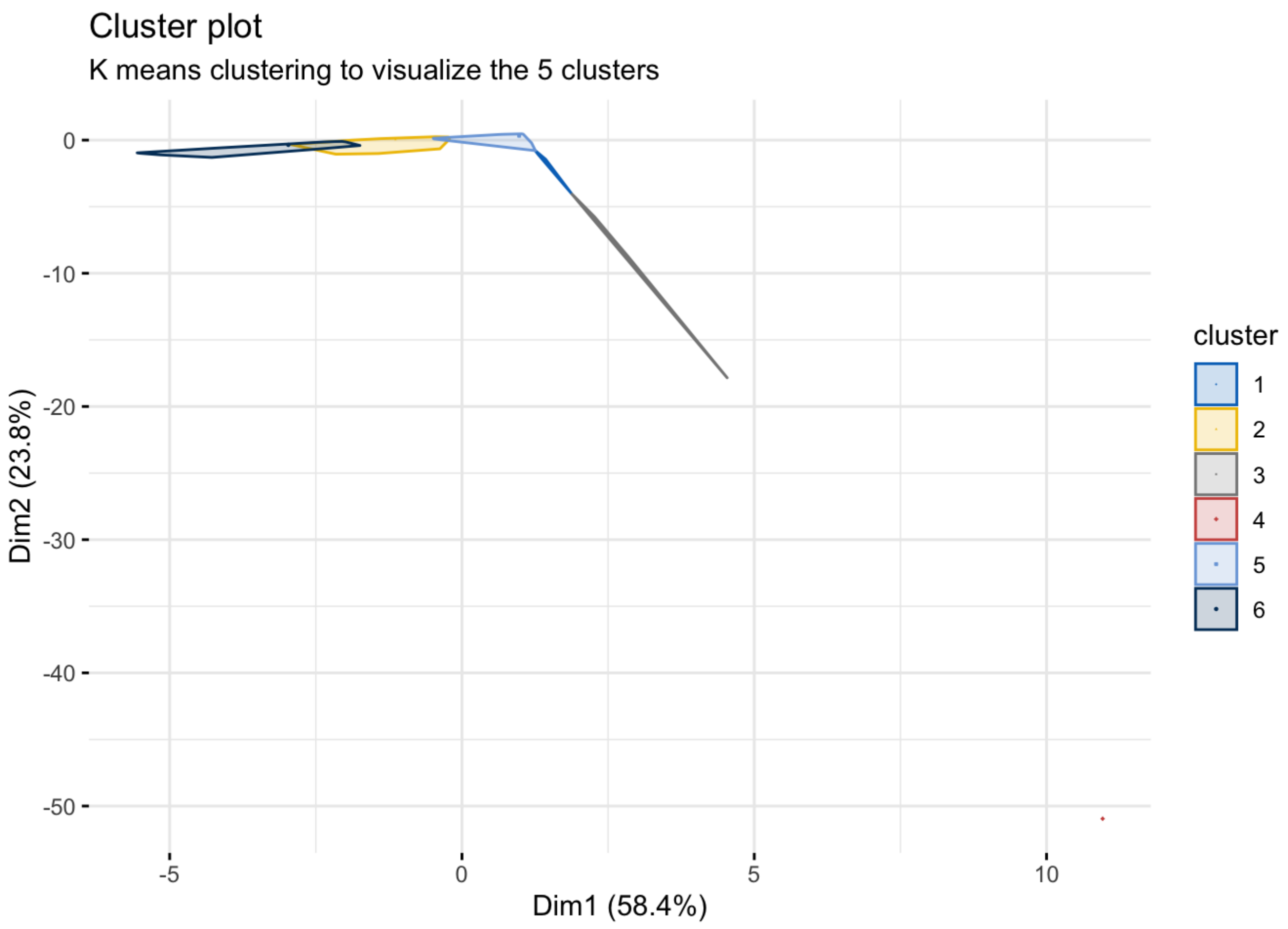
```
kmean <- kmeans(train_data_norm[, 4:7], centers = 6)

## Below finding cluster center for all rows and colomns
kmean$centers
```

```
##   fuel_received_units fuel_mmbtu_per_unit sulfur_content_pct ash_content_pct
## 1           2.1677469          -0.7963608        -0.51998098      -0.5500416
## 2          -0.2211667           1.1873022         0.08285029       0.6310875
## 3           6.4890043          -0.7996130        -0.51998098      -0.5500416
## 4          52.1791803          -0.8911217        -0.51998098      -0.5500416
## 5          -0.1472000          -0.7231877        -0.49352278      -0.5500416
## 6          -0.2424190           1.3939162         2.25568747       1.5256845
```

```
## Number of observation in each cluster
kmean$size
```

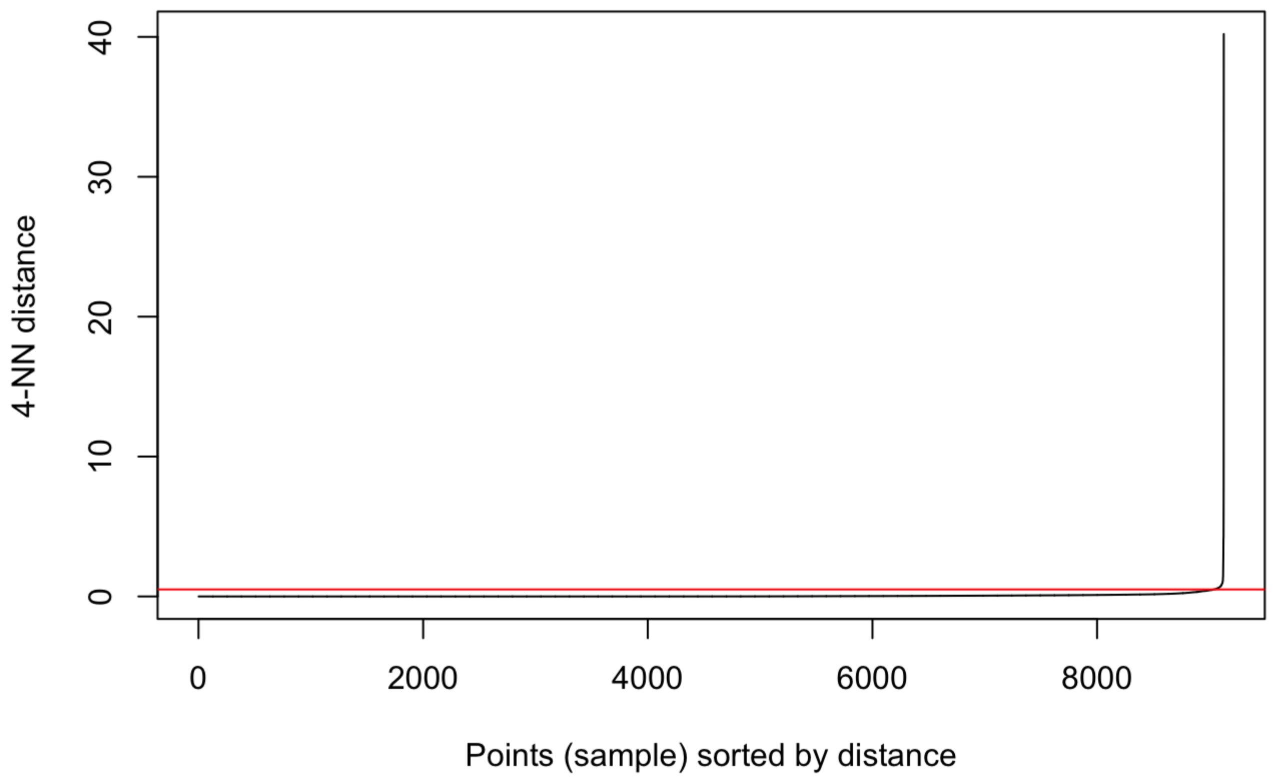
```
## [1]  498 2156   62    1 5219 1192
```

```
library(factoextra)
##For the above observations applyng the K means clustering to visualize the 5 cluste
rs
fviz_cluster(kmean, data=train_data_norm[,4:7],  geom = "convex",
             palette = "jco", ggtheme = theme_minimal()) + labs(subtitle = "K means c
lustering to visualize the 5 clusters")
```



```
# After using K mean algorithm we can see that clusters 3,4 and 5 are overlapping hen
ce it indicates that the data is not well-suited for this type of clustering. In such
cases, we can consider using density-based clustering algorithms such as DBSCAN. Comp
ared to k-means clustering, DBSCAN is more robust to noise and can handle data with v
arying densities and shapes
```

```
library(dbscan)

dbscan::kNNdistplot(train_data_norm[,4:7], k =  4)
abline(h = 0.5,col="red")
```

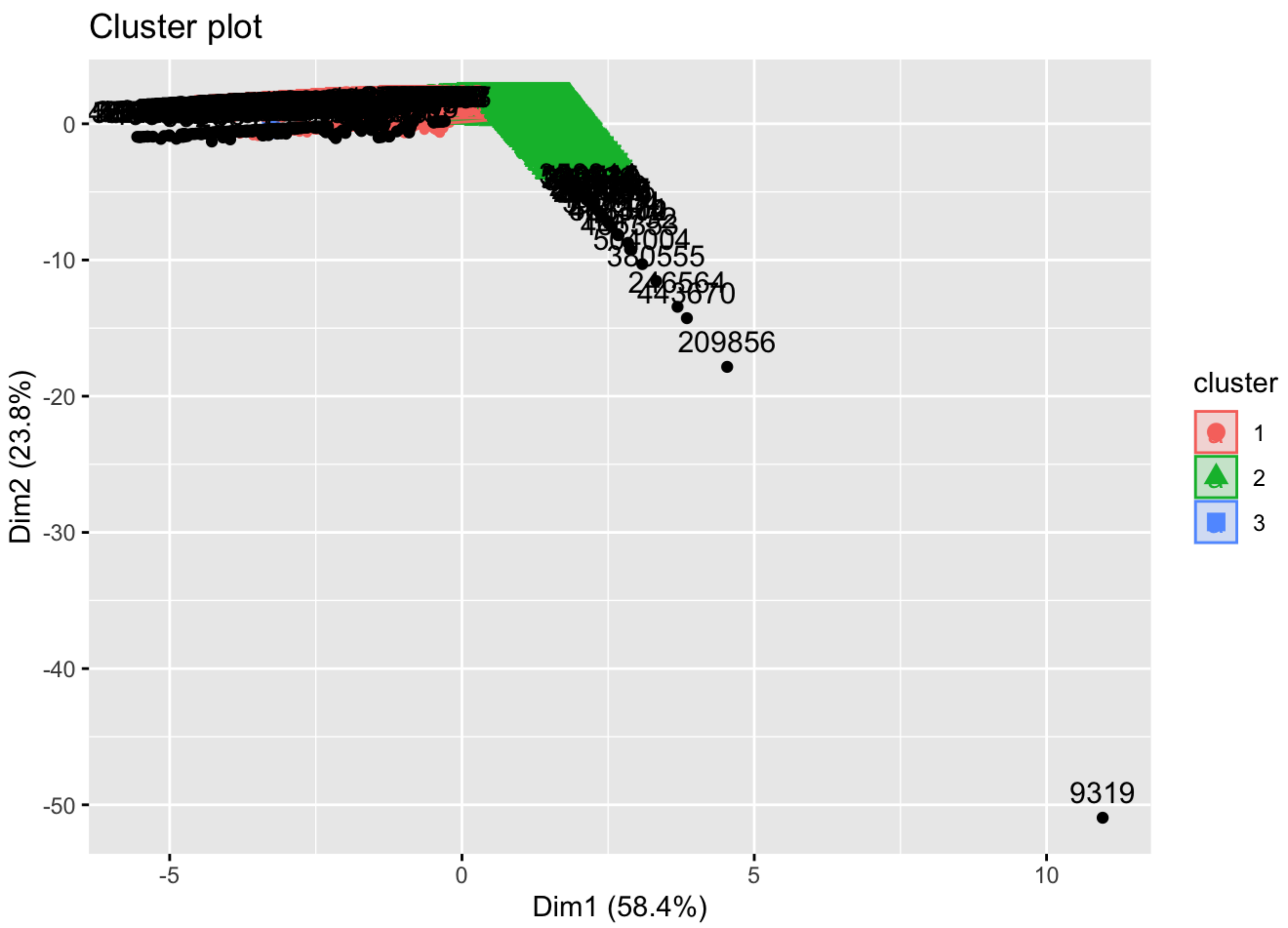


```
#The kNN distplot with k=4 has identified a clear elbow point at 0.5, indicating a re
asonable value for the epsilon parameter in the DBSCAN algorithm. This means that dat
a points that are within a distance of 0.5 from each other are considered to be in th
e same cluster. The appropriate value for minPts is then chosen based on domain knowl
edge and experimentation.
```

```
# Perform DBSCAN clustering

#Selecting numerical data to form clusters:
Training_numerical<-train_data[,c(4:7)]
#Normalizing the data:
Training_norm<-scale(Training_numerical)


db <- dbscan::dbscan(Training_norm, eps = 0.5, minPts = 25)
db
```

```
## DBSCAN clustering for 9128 objects.
## Parameters: eps = 0.5, minPts = 25
## Using euclidean distances and borderpoints = TRUE
## The clustering contains 3 cluster(s) and 294 noise points.
##
##    0    1    2    3
##  294 3072 5737   25
##
## Available fields: cluster, eps, minPts, dist, borderPoints
```

```
#The result of the clustering is that there are 4 clusters and 256 noise points. The
clusters are labeled as 0, 1, 2, and 3, with cluster 0 containing 294 points, cluster
1 containing 3072 points, cluster 2 containing 5737 points, and clusters 3 has 30 poi
nts.

# Visualize the clustering results
library(factoextra)
fviz_cluster(db, train_data_norm[,4:7], stand = TRUE, show.clust.cent = TRUE, ellips
e.type = "norm")
```



```
#Let us explore the clusters formed and try to understand how each attribute is behav
ing in different cluster.

#Assigning clusters to the original data:
train_set<-cbind(train_data,db$cluster)


head(train_set)
```

	contract_type_code <fct>	energy_source_code <fct>	fuel_group_code <fct>	fuel_received_units <dbl>
301641	C	BIT	coal	7603
84826	S	NG	natural_gas	85
581296	S	NG	natural_gas	5911

	contract_type_code <fct>	energy_source_code <fct>	fuel_group_code <fct>	fuel_received_units <dbl>
438751	S	RFO	petroleum	2359
258708	C	WC	coal	17840
592428	C	NG	natural_gas	283417

6 rows | 1-5 of 14 columns

```
#Let us explore the clusters formed and try to understand how each attribute is behav
ing in different cluster.

library(dplyr)
#Finding mean within each cluster to interpret the clusters:
train_set%>%group_by(db$cluster)%>%
  summarize(avg_units=mean(fuel_received_units),
            avg_cost=mean(fuel_cost_per_mmbtu),
            avg_mmbtu=mean(fuel_mmbtu_per_unit))
```

db$cluster <int>	avg_units <dbl>	avg_cost <dbl>	avg_mmbtu <dbl>
0	1008327.02	3.031177	15.285986
1	45303.13	2.691056	21.660984
2	333651.24	7.235618	1.672761
3	7182.60	3.232680	12.457120

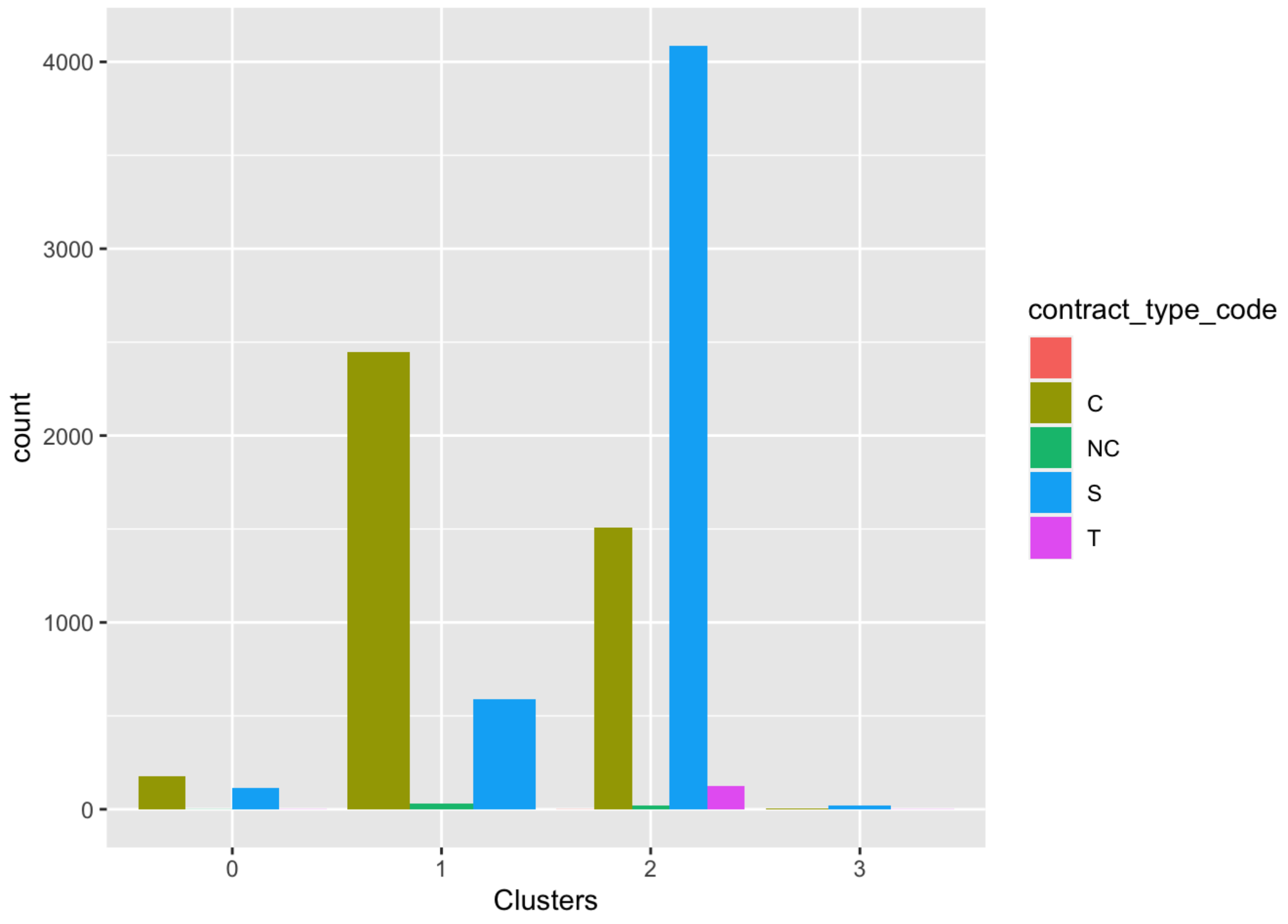
4 rows

```
#From the above observation we can see that Average cost for cluster 2 is the highest
and maximum Average heat is generated by cluster 1 fuel type and least heat is genera
ted by Cluster 2 unit fuel




#Adding the cluster information to original data without dummy variable and let us us
e this for futher analysis.
set.seed(8627)
Part_data<-caret::createDataPartition(Fuel_data$fuel_mmbtu_per_unit,p=0.75,list = FAL
SE)

Data_final<-Fuel_data[train_idx,]

nrow(Data_final)
```

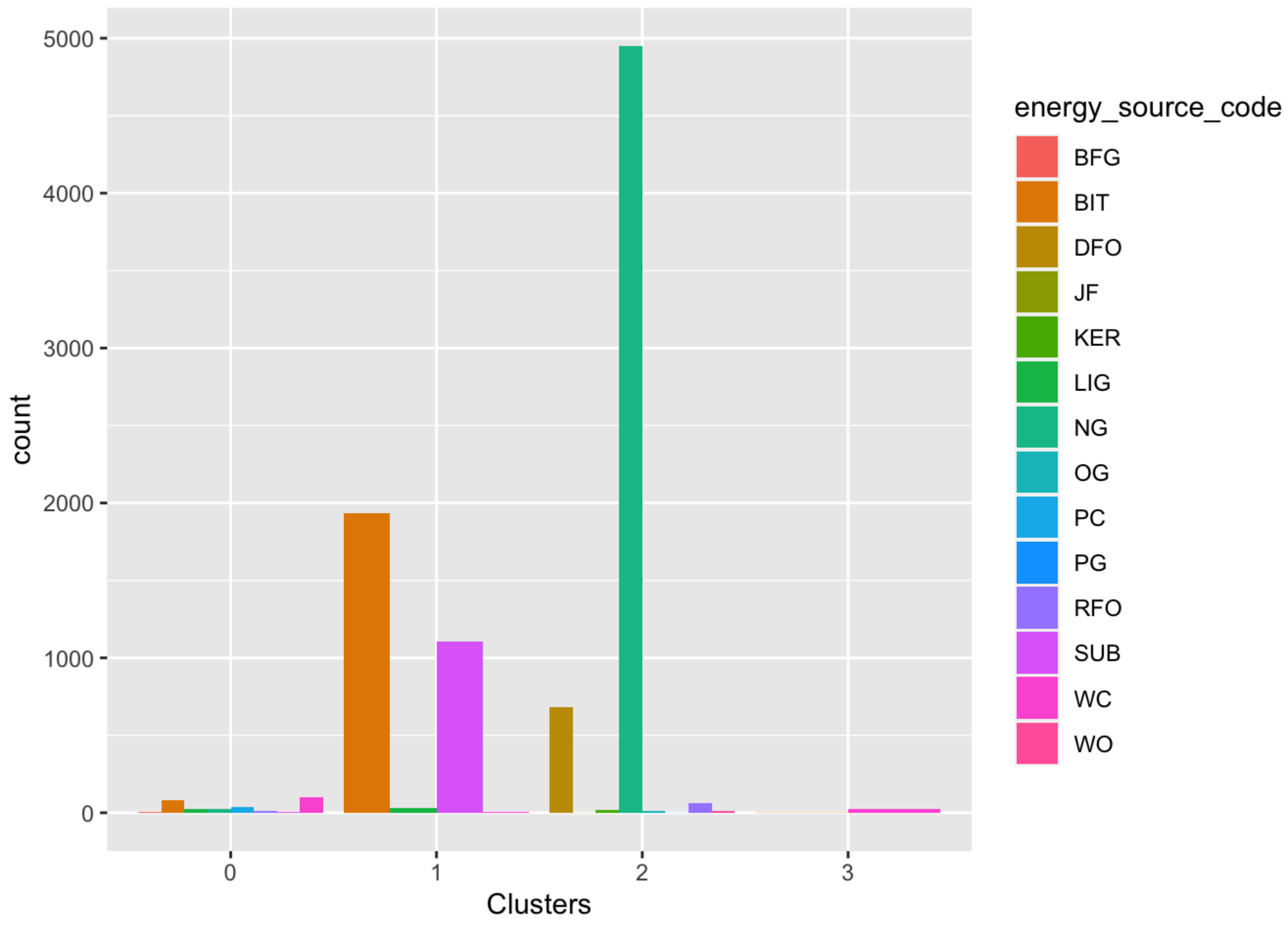
```
## [1] 9128
```

```
# Cluster 1 - Gas
# Cluster 2 - Oil
# Cluster 3 - Coal

Data_final$cluster<-db$cluster
```

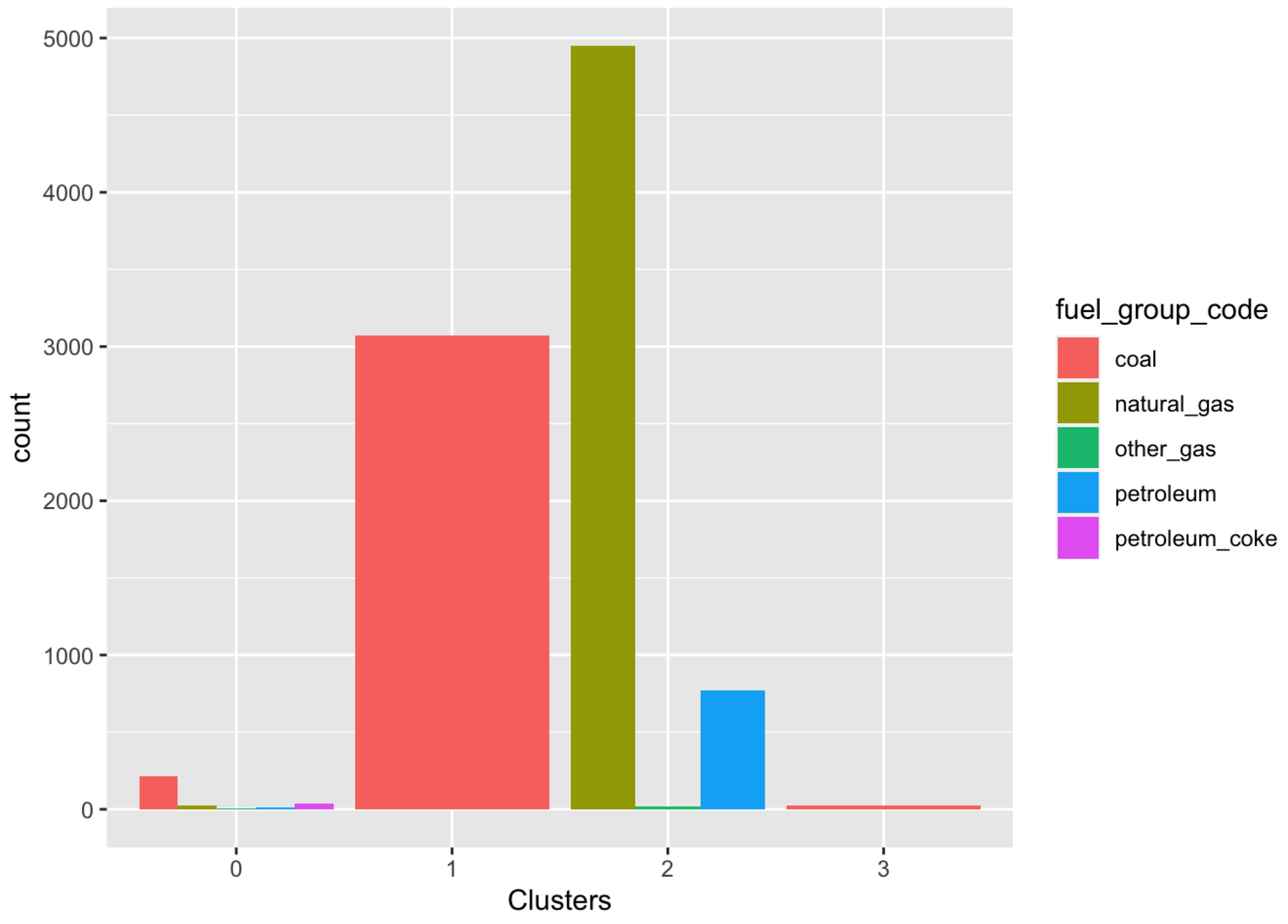
```
library(ggplot2)

#Observing the Distribution of Fuel across Different Contracts
ggplot(Data_final, mapping = aes(factor(cluster), fill =contract_type_code))+geom_bar
(position='dodge')+labs(x ='Clusters')
```



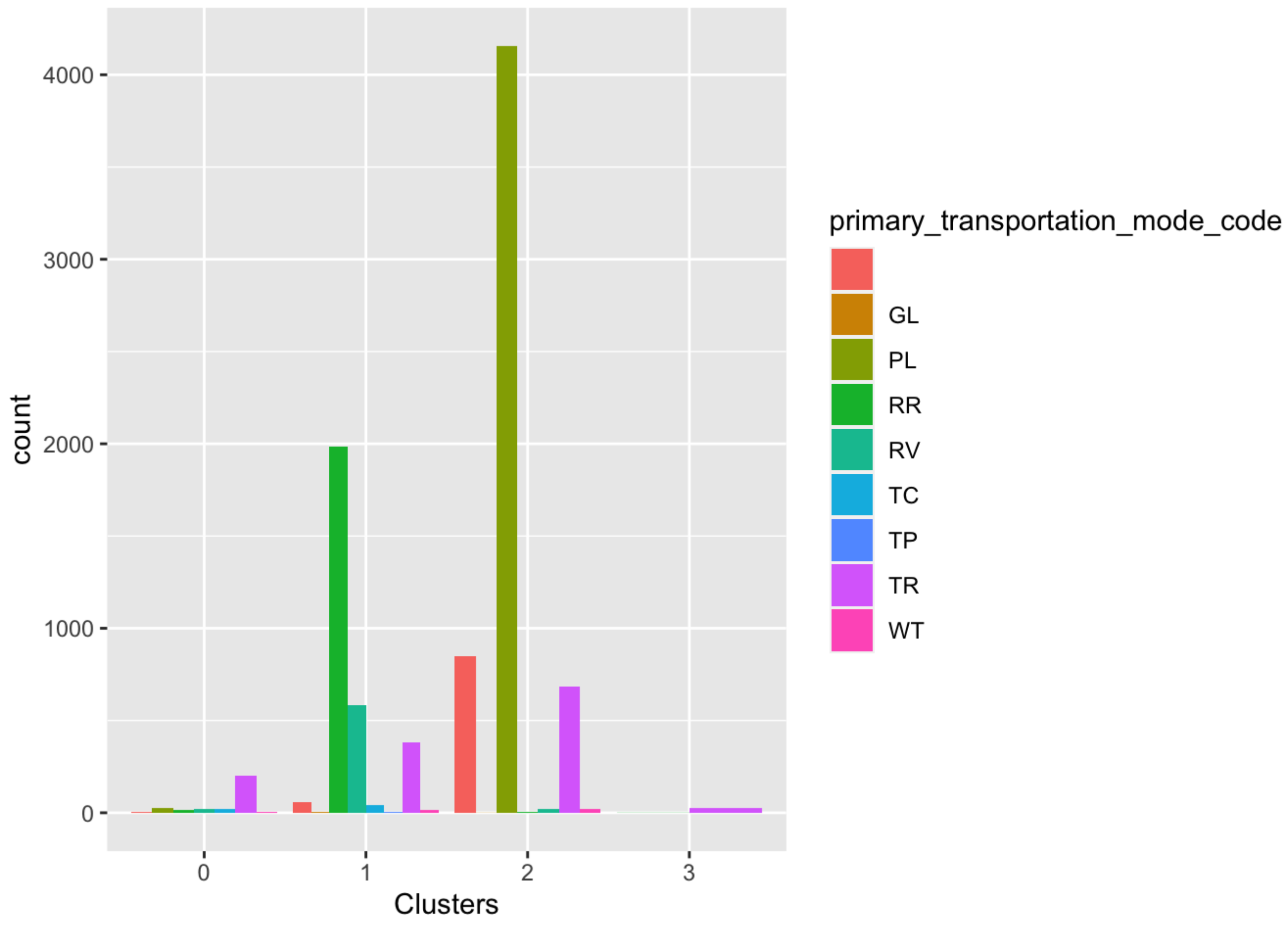
```
#Observing the Distribution of Fuel across Different Energy sources
ggplot(Data_final, mapping = aes(factor(cluster), fill =energy_source_code))+geom_bar
(position='dodge')+labs(x ='Clusters')
```



```
#Observing the Distribution of Fuel across Different Fuel group codes
ggplot(Data_final, mapping = aes(factor(cluster), fill =fuel_group_code))+geom_bar(po
sition='dodge')+labs(x ='Clusters')
```



```
#Observing the Distribution of Fuel across Different Transportation codes
ggplot(Data_final, mapping = aes(factor(cluster), fill =primary_transportation_mode_c
ode))+geom_bar(position='dodge')+labs(x ='Clusters')
```


count
4000
3000
2000
1000
0
0
1
2
3
Clusters
primary_transportation_mode_code
GL
PL
RR
RV
TC
TP
TR
WT

```
### Cluster 1 -  Gas

#The analysis provides insights into the characteristics of Cluster 1, which represen
ts the Gas fuel type.
#The analysis suggests that Gas has the lowest average fuel cost per mmbtu compared t
o other fuel types, which could make it a cost-effective option for energy generatio
n.
#Gas has the lowest average fuel mmbtu per unit, indicating that it generates less he
at content per unit compared to other fuel types.
#Gas does not contain any ash, sulfur, and mercury content, which could make it a cle
an and environmentally friendly fuel option.
#Natural gas is the energy source code for Gas, and pipelines are the most commonly u
sed transportation type to supply this fuel type. This information could be useful in
assessing the infrastructure requirements and associated costs for using Gas as a fue
l type.
#The graph indicates that Gas is mainly purchased on spot rather than through contrac
ts, which could be an important factor to consider for procurement strategies.

### Cluster 2 -  Oil

#Oil is the most expensive type of fuel and is purchased only on spot. This suggests
that the buyers are not willing to enter into long-term contracts for purchasing oil,
possibly due to the high cost and uncertainty of future prices.
#Oil has a relatively low average units received in comparison to gas and coal could
be due to its high cost. Customers may prefer to use other fuel types that are cheape
r and more cost-effective.
#Sulfur content in oil is important as it indicates that oil may have negative enviro
nmental impacts such as air pollution



### Cluster 3 -  Coal
#Coal has the lowest average cost per unit of fuel, it also has the highest average f
uel mmbtu per unit, indicating that it produces more heat energy per unit than gas or
oil. This could explain why it is widely used in the US despite its environmental imp
act.
#Coal is purchased both on spot and on contract, but there are more spot purchases th
an contract purchases.
#Coal include BIT and SUB, which stand for Bituminous Coal and Sub bituminous Coal, r
espectively. These are two types of coal that differ in their energy content and chem
ical properties, and they are both widely used in the US for electricity generation a
nd other industrial purposes.
#Presence of ash, sulfur, and mercury in coal can have significant environmental and
health impacts, including air pollution, acid rain, and water pollution. It is import
ant to note that the use of coal is declining in the US and many other countries due
to these impacts, as well as the increasing availability and affordability of renewab
le energy sources.
```

```
#Use multiple-linear regression to determine the best set of variables to predict fue
l_cost_per_mmbtu. Running the multiple linear regression model to determine the best
set of variables to predict fuel_cost_per_mmbtu by considering variables which were u
sed to form clusters:

Model_data<- lm(train_data$fuel_cost_per_mmbtu~.,data=train_data)

summary(Model_data)
```

```
##
## Call:
## lm(formula = train_data$fuel_cost_per_mmbtu ~ ., data = train_data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -14.1   -3.5   -1.1    0.2 4640.2
##
## Coefficients: (7 not defined because of singularities)
##                                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)                          1.763e+01  6.338e+01   0.278    0.781
## contract_type_codeC                  3.120e+00  5.348e+01   0.058    0.953
## contract_type_codeNC                 3.462e+00  5.398e+01   0.064    0.949
## contract_type_codeS                  4.208e+00  5.347e+01   0.079    0.937
## contract_type_codeT                  6.608e-01  5.368e+01   0.012    0.990
## energy_source_codeBIT               -2.051e+01  3.979e+01  -0.516    0.606
## energy_source_codeDFO               -9.914e+00  3.486e+01  -0.284    0.776
## energy_source_codeJF                -2.109e+01  6.352e+01  -0.332    0.740
## energy_source_codeKER               -1.898e+01  3.691e+01  -0.514    0.607
## energy_source_codeLIG               -1.987e+01  3.732e+01  -0.532    0.595
## energy_source_codeNG                -1.635e+01  3.395e+01  -0.481    0.630
## energy_source_codeOG                -1.849e+01  3.683e+01  -0.502    0.616
## energy_source_codePC                -2.086e+01  4.177e+01  -0.499    0.617
## energy_source_codePG                -8.181e+00  6.368e+01  -0.128    0.898
## energy_source_codeRFO               -1.590e+01  3.519e+01  -0.452    0.651
## energy_source_codeSUB               -2.073e+01  3.715e+01  -0.558    0.577
## energy_source_codeWC                -2.052e+01  3.949e+01  -0.520    0.603
## energy_source_codeWO                -1.804e+01  3.812e+01  -0.473    0.636
## fuel_group_codenatural_gas                  NA         NA      NA       NA
## fuel_group_codeother_gas                    NA         NA      NA       NA
## fuel_group_codepetroleum                    NA         NA      NA       NA
## fuel_group_codepetroleum_coke               NA         NA      NA       NA
## fuel_received_units                 -9.372e-07  6.936e-07  -1.351    0.177
## fuel_mmbtu_per_unit                  9.255e-02  7.333e-01   0.126    0.900
## sulfur_content_pct                  -1.181e-01  1.101e+00  -0.107    0.915
## ash_content_pct                      2.618e-02  2.874e-01   0.091    0.927
## primary_transportation_mode_codeGL -1.922e+00  2.461e+01  -0.078    0.938
## primary_transportation_mode_codePL  2.085e+00  1.993e+00   1.046    0.296
## primary_transportation_mode_codeRR  3.967e-01  5.858e+00   0.068    0.946
## primary_transportation_mode_codeRV -6.066e-02  6.068e+00  -0.010    0.992
## primary_transportation_mode_codeTC  7.305e-01  9.135e+00   0.080    0.936
## primary_transportation_mode_codeTP  7.199e-01  2.461e+01   0.029    0.977
## primary_transportation_mode_codeTR  2.134e-01  5.861e+00   0.036    0.971
## primary_transportation_mode_codeWT  1.682e+00  1.001e+01   0.168    0.867
## fuel_type_gas                               NA         NA      NA       NA
## fuel_type_coal                              NA         NA      NA       NA
## fuel_type_oil                               NA         NA      NA       NA
##
## Residual standard error: 53.46 on 9098 degrees of freedom
## Multiple R-squared:  0.002928,   Adjusted R-squared:  -0.0002497
## F-statistic: 0.9214 on 29 and 9098 DF,  p-value: 0.5867
```

```
#Fuel received units,fuel_type_coal and fuel_type_oil best determine the fuel_cost_pe
r_mmbtu variable.
#Checking the prediction of the above model on Test data


#We could see that the difference between predicted values and actual values is too h
igh,which means that it is too complex and is fitting the noise in the data instead o
f the underlying patterns. This can result in poor performance on new, unseen data.
```