DANA 4840 Project - Hierarchical Clustering

Aryan Mukherjee, Maryam Gadimova, Patricia Tating, Roman Shrestha

1. Research Statement on Cars Dataset

The automobile industry continually seeks to understand the characteristics and performance of various car models in order to meet the varied needs of its customers. The given dataset contains numerous characteristics that can be used to categorize cars into different groups, including weight (wt), horsepower (hp), engine displacement (disp), and miles per gallon (mpg).

The objective of this analysis is to segment the car models into clusters that represent different types of vehicles. This can help in understanding the underlying patterns in the data, such as identifying clusters of high-performance sports cars, fuel-efficient vehicles, or family-oriented models. Additionally, by using this clustering, marketers and producers can more successfully target particular customer categories.

2. Preliminaries

Before diving into the cluster analysis, let's first thoroughly examine and understand our data. This preliminary step will allow us to identify key patterns and characteristics within the dataset, ensuring a solid foundation for accurate analysis. By doing so, we can address any potential data quality issues and refine our approach for more meaningful results.

```
library("tidyverse")
library("factoextra")
library("dendextend")
library("hopkins")
library("corrplot")
library("cluster")
library("patchwork")
library("clValid")
library("gridExtra")
```

2.1. Reading the Data

```
mtcars <- read.csv("data/mtcars.csv", header = T, sep = ",")
head(mtcars)
```

```
##               model  mpg cyl disp  hp drat    wt  qsec vs am gear carb
## 1         Mazda RX4 21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## 2     Mazda RX4 Wag 21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## 3        Datsun 710 22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## 4    Hornet 4 Drive 21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## 5 Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## 6           Valiant 18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

2.1.1. Checking Data Structure

```
dim(mtcars)
```

```
## [1] 32 12
```

```
str(mtcars)
```

```
## 'data.frame':    32 obs. of  12 variables:
##  $ model: chr  "Mazda RX4" "Mazda RX4 Wag" "Datsun 710" "Hornet 4 Drive" ...
##  $ mpg  : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl  : int  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp : num  160 160 108 258 360 ...
##  $ hp   : int  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat : num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt   : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec : num  16.5 17 18.6 19.4 17 ...
##  $ vs   : int  0 0 1 1 0 1 0 1 1 1 ...
##  $ am   : int  1 1 1 0 0 0 0 0 0 0 ...
##  $ gear : int  4 4 4 3 3 3 3 4 4 4 ...
##  $ carb : int  4 4 1 1 2 1 4 2 2 4 ...
```

We can see that our data comprises 32 observations of different car models and 12 automobile features of mixed (numeric, integer, character) data types.

2.2. Feature Explanation

The 'mtcars' dataset includes 12 features as detailed below:

- 'model' (categorical): Model of the vehicle
- 'mpg' (numerical): Miles/(US) gallon, a measure of fuel efficiency
- 'cyl' (categorical-ordinal): Number of cylinders in the engine
- 'disp' (numerical): Displacement in cubic inches
- 'hp' (numerical): Gross horsepower
- 'drat' (numerical): Rear axle ratio
- 'wt' (numerical): Weight of the car (1000 lbs)
- 'qsec' (numerical): 1/4 mile time, a measure of acceleration
- 'vs' (binary): Engine type (0 = V-shaped, 1 = straight)
- 'am' (binary): Transmission type (0 = automatic, 1 = manual)
- 'gear' (categorical-ordinal): Number of forward gears
- 'carb' (categorical-ordinal): Number of carburetor

2.3 Data Pre-processing

```
mtcars_categorical <- data.frame(
  cyl = mtcars$cyl,
  vs = mtcars$vs,
  am = mtcars$am,
```

```
  gear = mtcars$gear,
  carb = mtcars$carb
)

mtcars_numerical <- data.frame(
  mpg = mtcars$mpg,
  disp = mtcars$disp,
  hp = mtcars$hp,
  drat = mtcars$drat,
  wt = mtcars$wt,
  qsec = mtcars$qsec
)

mtcars_numerical_scaled <- data.frame(scale(mtcars_numerical))

mtcars_joined <- cbind(mtcars_numerical_scaled, mtcars_categorical)
rownames(mtcars_joined) <- mtcars$model
mtcars <- mtcars_joined
head(mtcars)
```

```
##                           mpg        disp         hp       drat           wt
## Mazda RX4           0.1508848 -0.57061982 -0.5350928  0.5675137 -0.610399567
## Mazda RX4 Wag       0.1508848 -0.57061982 -0.5350928  0.5675137 -0.349785269
## Datsun 710          0.4495434 -0.99018209 -0.7830405  0.4739996 -0.917004624
## Hornet 4 Drive      0.2172534  0.22009369 -0.5350928 -0.9661175 -0.002299538
## Hornet Sportabout -0.2307345  1.04308123  0.4129422 -0.8351978  0.227654255
## Valiant           -0.3302874 -0.04616698 -0.6080186 -1.5646078  0.248094592
##                          qsec cyl vs am gear carb
## Mazda RX4           -0.7771651   6  0  1    4    4
## Mazda RX4 Wag       -0.4637808   6  0  1    4    4
## Datsun 710           0.4260068   4  1  1    4    1
## Hornet 4 Drive       0.8904872   6  1  0    3    1
## Hornet Sportabout   -0.4637808   8  0  0    3    2
## Valiant              1.3269868   6  1  0    3    1
```

Our numerical features have different scale of measurements, so we standardized the data to ensure each variable contributes equally to the distance calculations, preventing variables with larger scales to have more weight in the clustering results. We also do not standardize the categorical data.

2.4 Exploratory Data Analysis

2.4.1. Checking Missing Values

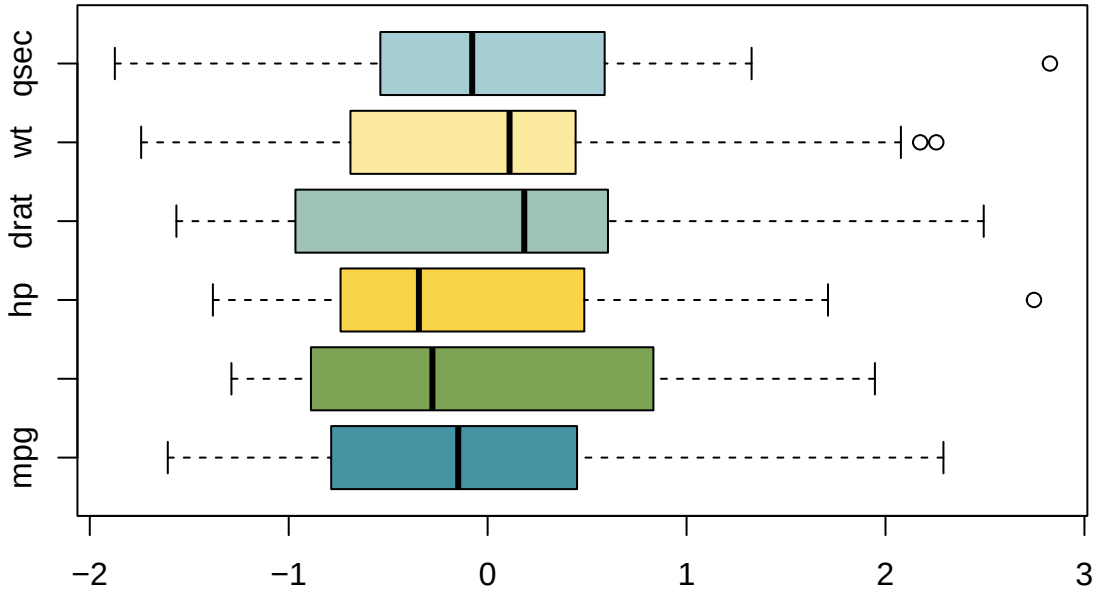
```
missing_mtcars <- sapply(mtcars, function(x) sum(is.na(x)))
missing_mtcars
```

```
##  mpg disp   hp drat   wt qsec  cyl   vs   am gear carb
##    0    0    0    0    0    0    0    0    0    0    0
```

From this, we can see that our data does not have the presence of any missing values.

2.4.2. Boxplots for Different Features

```
color_palette <- c("#4494a4", "#7ca454", "#f9d448", "#9fc4b7", "#fcea9e", "#a6ccd4")

boxplot(mtcars_numerical_scaled, horizontal = T, col = color_palette)
```



The graph presents standardized boxplots for several numerical variables, facilitating a comparative analysis of their distributions. The variables 'qsec', 'hp', and 'disp' exhibit lower medians and narrower ranges compared to 'wt', 'drat', and 'mpg'.

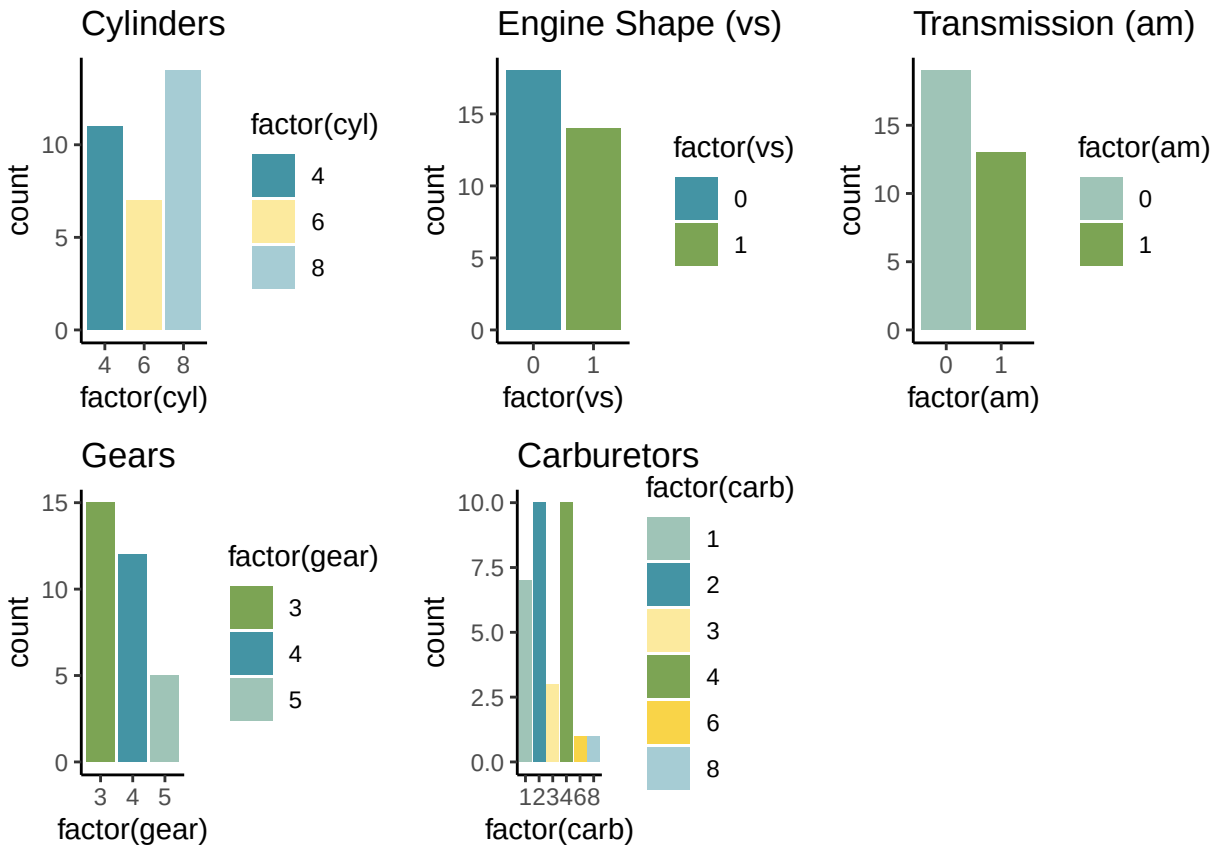
Notably, the variables 'qsec', 'hp', and 'wt' have outliers on the positive side, indicating some data points that are significantly higher than the majority of the data for these variables.

Besides this we can also see that the distribution of most of the numeric data are skewed either to the right or left with 'qsec', 'hp', and 'disp' being skewed to the right, 'wt' and 'drat' to the left, and 'mpg' appearing to be symmetric.

2.4.3. Bar Plot for Categorical Variables

```
set.seed(42)

plot_cyl <- ggplot(mtcars_categorical, aes(x = factor(cyl), fill = factor(cyl))) +
  geom_bar() +
  scale_fill_manual(values = sample(color_palette, 3)) +
```

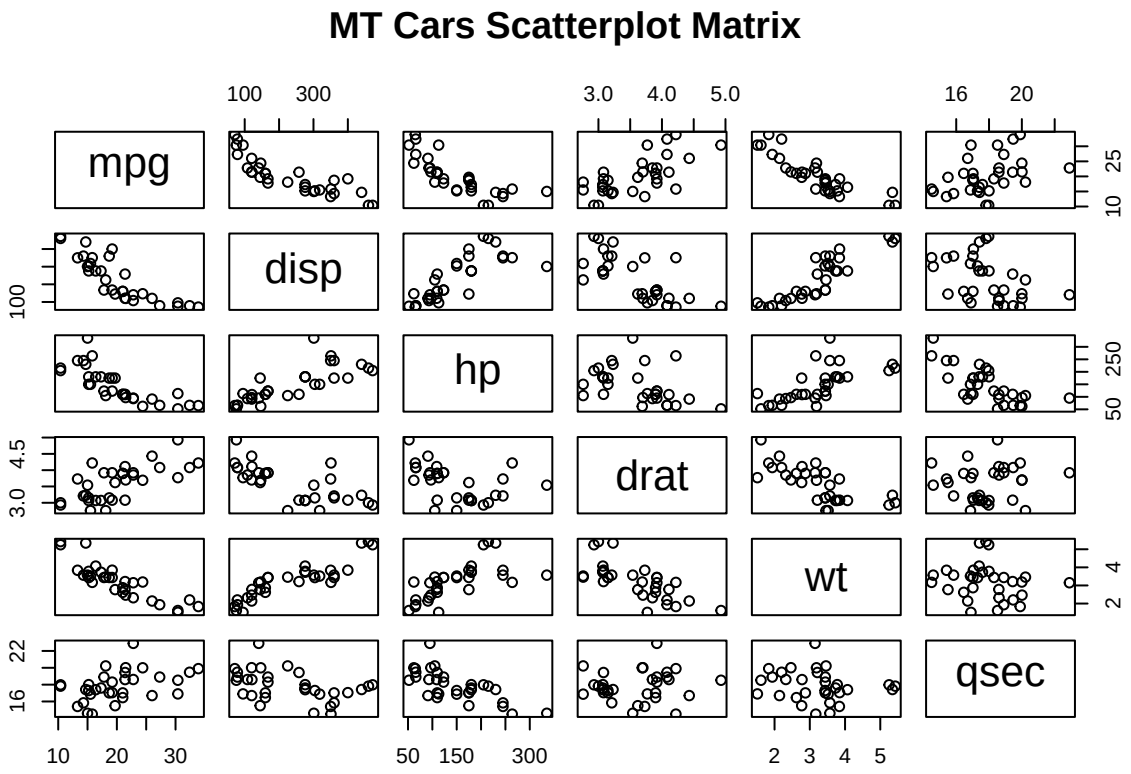
```
  theme_classic() +
  ggtitle("Cylinders")

plot_vs <- ggplot(mtcars_categorical, aes(x = factor(vs), fill = factor(vs))) +
  geom_bar() +
  scale_fill_manual(values = sample(color_palette, 2)) +
  theme_classic() +
  ggtitle("Engine Shape (vs)")

plot_am <- ggplot(mtcars_categorical, aes(x = factor(am), fill = factor(am))) +
  geom_bar() +
  scale_fill_manual(values = sample(color_palette, 2)) +
  theme_classic() +
  ggtitle("Transmission (am)")

plot_gear <- ggplot(mtcars_categorical, aes(x = factor(gear), fill = factor(gear))) +
  geom_bar() +
  scale_fill_manual(values = sample(color_palette, 3)) +
  theme_classic() +
  ggtitle("Gears")

plot_carb <- ggplot(mtcars_categorical, aes(x = factor(carb), fill = factor(carb))) +
  geom_bar() +
  scale_fill_manual(values = sample(color_palette, 6)) +
  theme_classic() +
  ggtitle("Carburetors")

grid.arrange(plot_cyl, plot_vs, plot_am, plot_gear, plot_carb, ncol = 3)
```



- **Cylinders (cyl):** Cars with 8 cylinders are the most common, followed by those with 4 and 6 cylinders.
- **Engine Shape (vs):** This plot categorizes cars based on engine shape, where 0 represent a V-shaped engine and 1 a straight engine. More cars have the V-shaped type of engine shape compared to the straight type.
- **Transmission (am):** This plot indicates the type of transmission, with 0 representing automatic and 1 manual. There are more cars with the automatic type of transmission than the manual type.
- **Gears (gear):** This plot shows the count of cars based on the number of gears. Cars with 3 gears are the most prevalent, followed by those with 4 and 5 gears.
- **Carburetors (carb):** Cars with 2 carburetors are the most common, followed by those with 4 and 1. There are fewer cars with 6 and 8 carburetors.

2.4.5. Scatterplot matrix

```
pairs(mtcars_numerical,
   main="MT Cars Scatterplot Matrix")
```



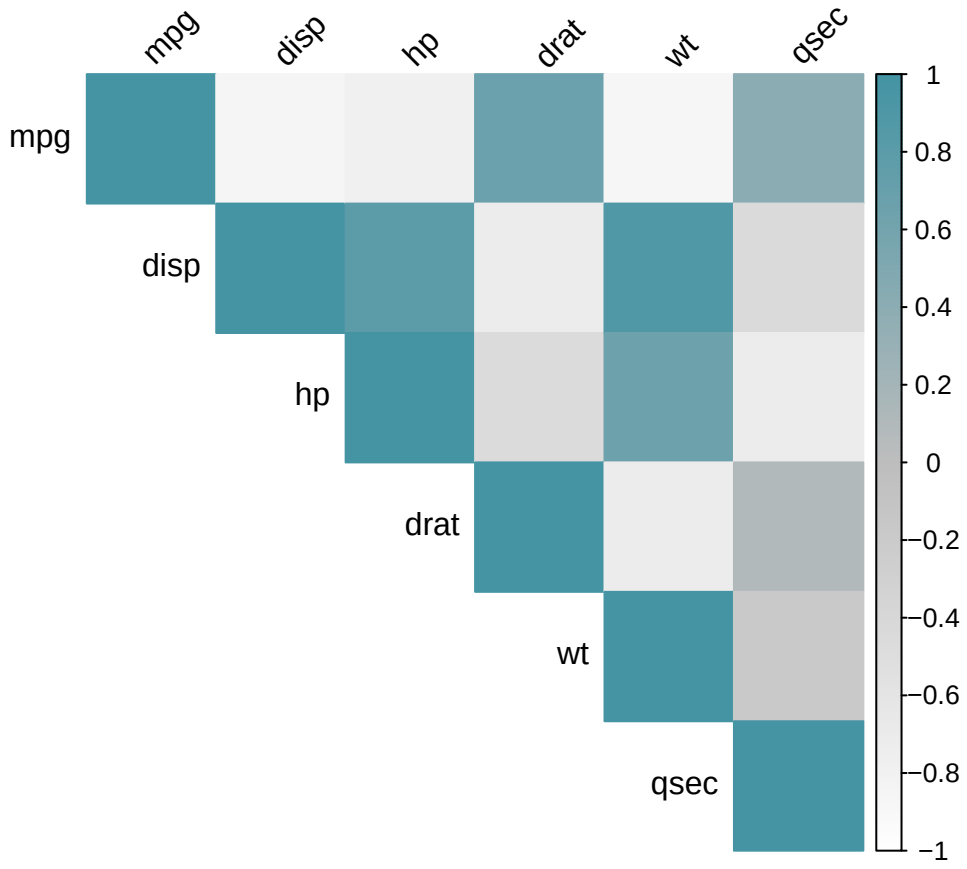
2.4.6. Correlation Matrix

```
cor_matrix <- cor(mtcars_numerical, use = "complete.obs")
cor_matrix
```

```
##             mpg       disp         hp        drat         wt        qsec
## mpg   1.0000000 -0.8475514 -0.7761684  0.68117191 -0.8676594  0.41868403
## disp -0.8475514  1.0000000  0.7909486 -0.71021393  0.8879799 -0.43369788
## hp   -0.7761684  0.7909486  1.0000000 -0.44875912  0.6587479 -0.70822339
## drat  0.6811719 -0.7102139 -0.4487591  1.00000000 -0.7124406  0.09120476
## wt   -0.8676594  0.8879799  0.6587479 -0.71244065  1.0000000 -0.17471588
## qsec  0.4186840 -0.4336979 -0.7082234  0.09120476 -0.1747159  1.00000000
```

2.4.7. Correlation Heatmap

```
corrplot(cor_matrix,
         method = "color",
         type = "upper",
         col = colorRampPalette(c("white", "grey", "#4494a4"))(200),
         tl.col = "black",
         tl.srt = 45)
```



- **Fuel Efficiency (mpg):** Strongly negatively correlated with engine displacement and vehicle weight, as expected. Moderately positively correlated with rear axle ratio and acceleration time.
- **Engine Displacement (disp):** Strongly positively correlated with horsepower and vehicle weight. Moderately negatively correlated with rear axle ratio and acceleration time.
- **Horsepower (hp):** Strongly correlated with engine displacement and moderately with vehicle weight. Moderately negatively correlated with rear axle ratio and acceleration time.
- **Rear Axle Ratio (drat):** Strongly negatively correlated with engine displacement and moderately with horsepower. Weight (wt): Strongly negatively correlated with fuel efficiency and moderately with horsepower. Very weakly negatively correlated with acceleration time.
- **Acceleration Time (qsec):** Moderately correlated with fuel efficiency and inversely correlated with engine displacement, horsepower, and weight.

3. Pre-clustering Assessment

Before performing clustering analysis, it is crucial to conduct a pre-clustering assessment to evaluate the dataset's cluster tendency and determine the optimal clustering approach. Tools like the Hopkins statistic and VAT can help assess whether the data points possess significant clustering tendencies. Once cluster tendency is established, the next step involves finding the optimal number of clusters. This can be achieved using methods such as the Elbow Method, Silhouette Analysis, or the Gap Statistic, each providing insights into the most meaningful way to partition the data.

3.1. Assessing Cluster Tendency

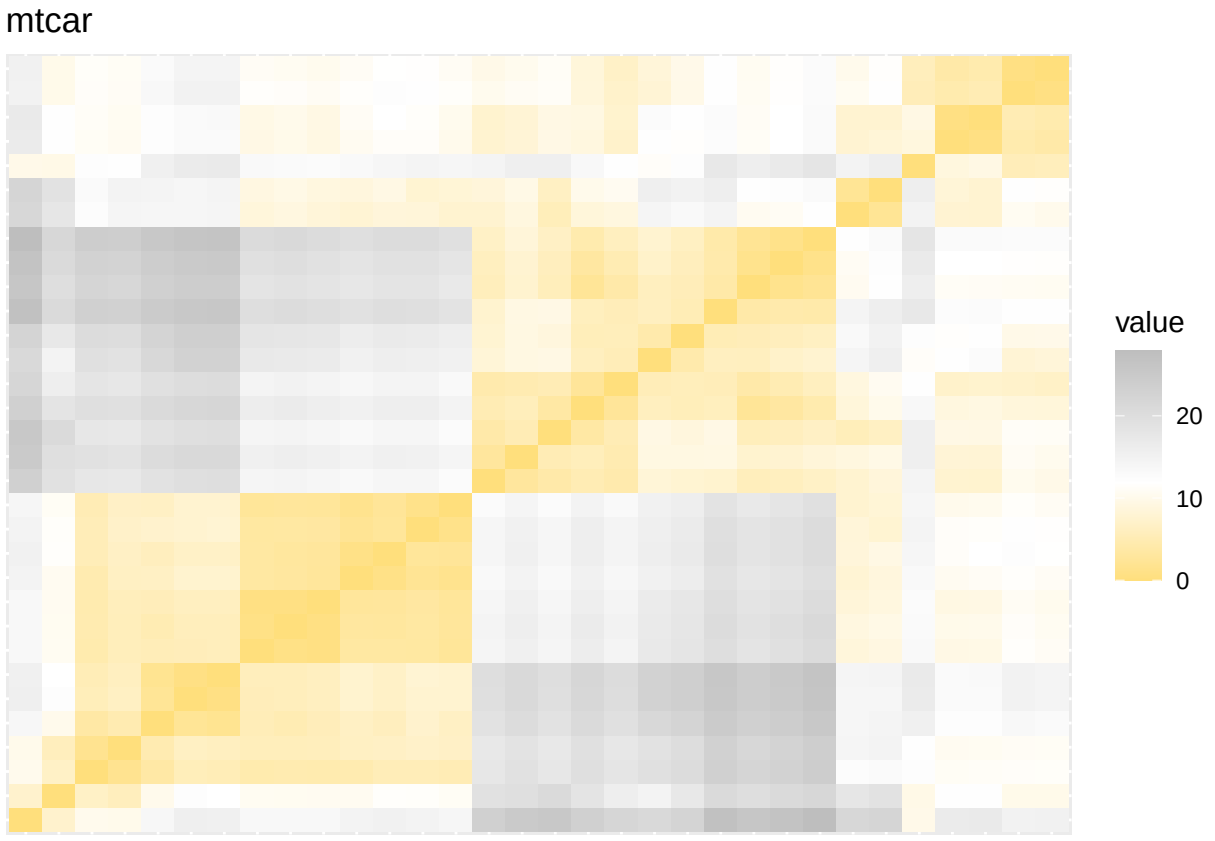
3.1.1. Hopkins Statistics

```
set.seed(25)

hopkins_mtcars <- hopkins(mtcars, m = ceiling(nrow(mtcars) / 10))
hopkins_mtcars
```

```
## [1] 0.9999998
```

3.1.2. Visual Assessment of Cluster Tendency (VAT)

```
fviz_dist(
  dist(mtcars, method = "manhattan"),
  show_labels = FALSE,
  gradient = list(low = "#f9d448", mid = "white", high = "grey")
) + labs(title = "mtcar")
```

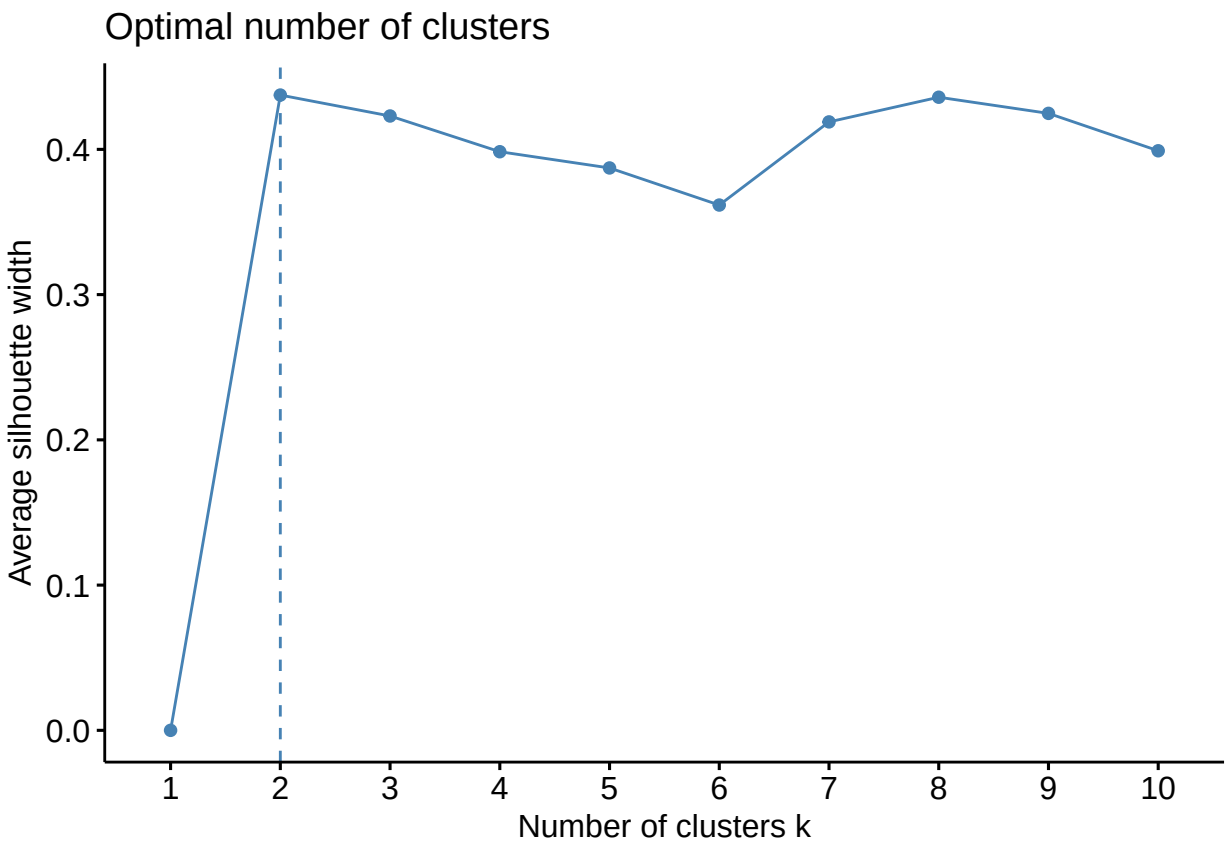


The Hopkins statistic of 0.9999998 indicates a strong clustering tendency in the dataset, suggesting that the data is well-suited for cluster analysis. The VAT plot visually reinforces this, showing two distinct blocks of similar values along the diagonal, which corresponds to well-separated clusters.

3.2. Finding the Optimal Number of Clusters

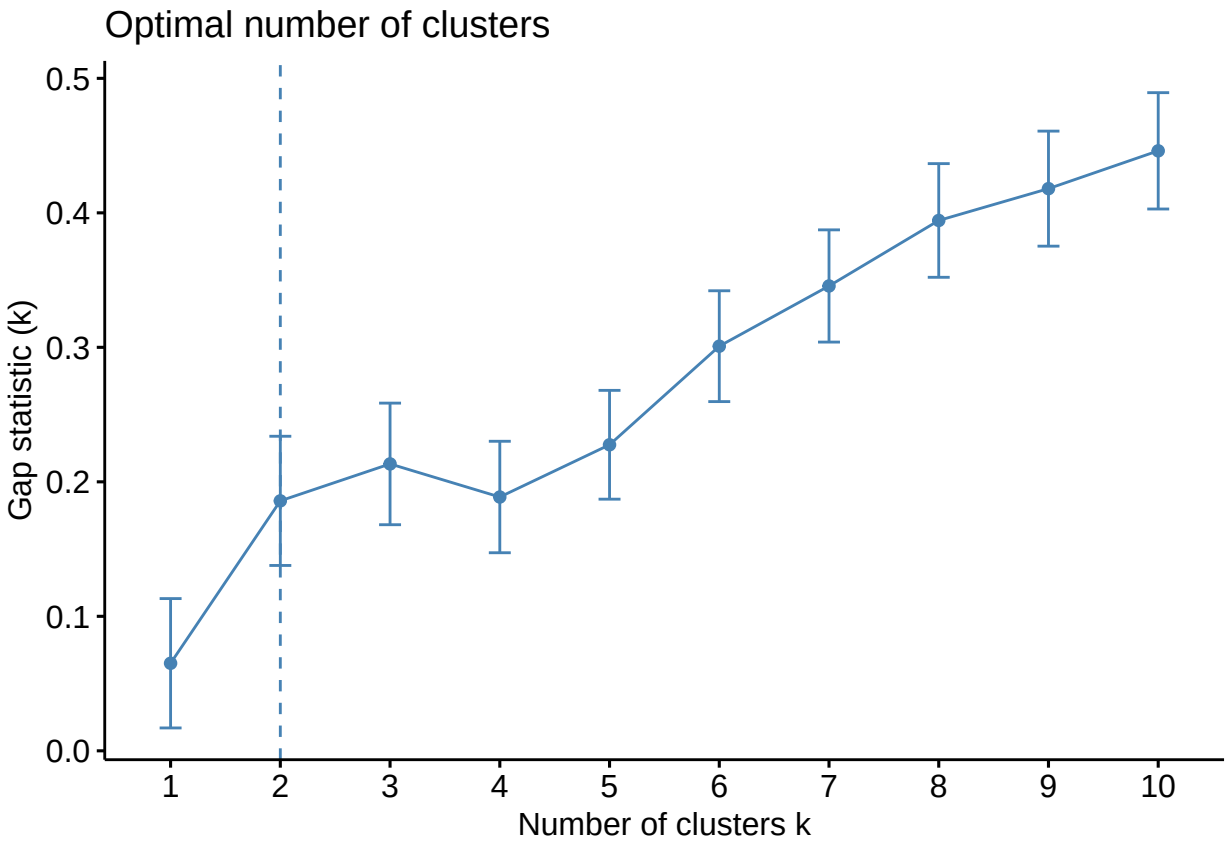
3.2.1. Silhouette Method

```
mtcars_silhouette_hierarchical <- fviz_nbclust(mtcars, hcut, method = "silhouette")
mtcars_silhouette_hierarchical
```



3.2.2. Gap Statistics

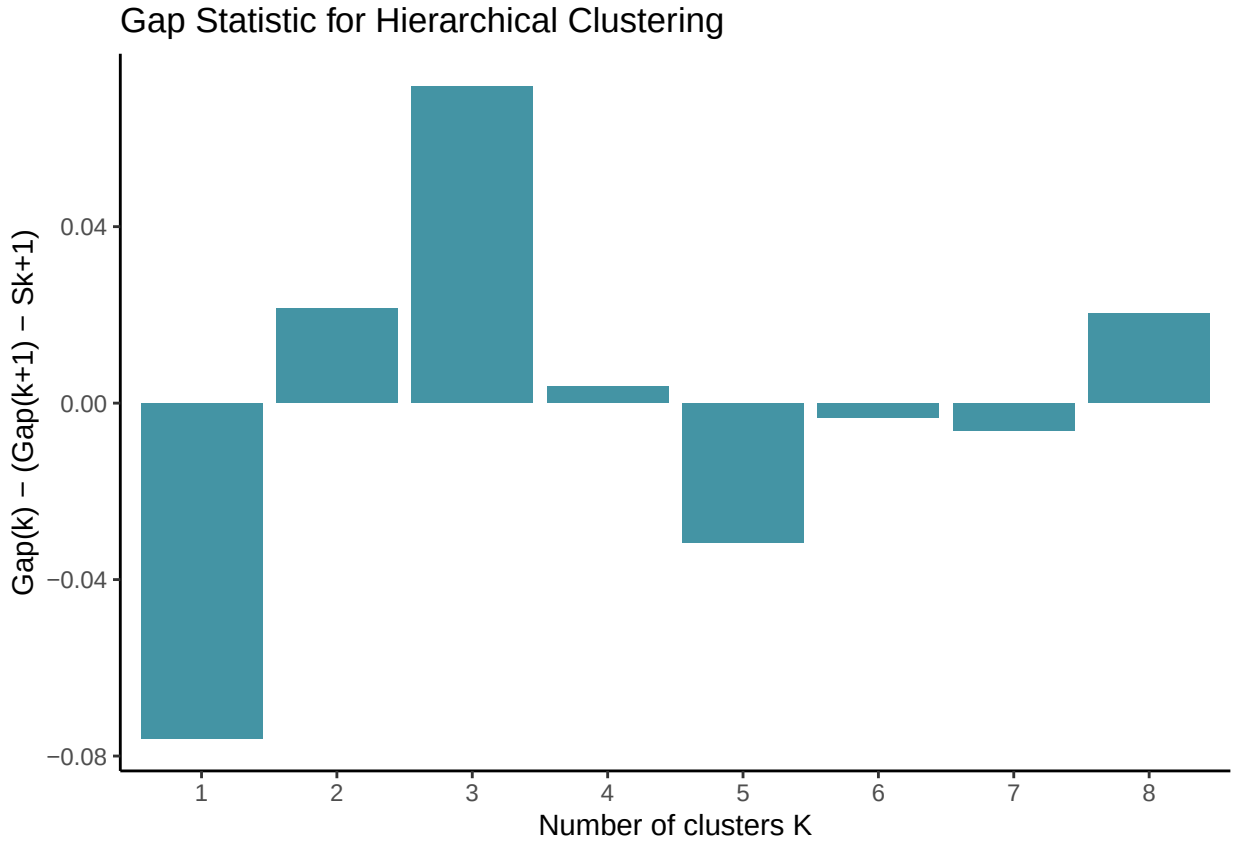
```
mtcars_gap_hierarchical <- fviz_nbclust(mtcars, hcut, nstart = 50,
                                        method = "gap_stat", nboot = 500)
mtcars_gap_hierarchical
```



3.2.2. Gap Statistics (Positive/Negative Graph)

```
get_cluster_diff <- function(gap_stat, max_k = 10) {
  gap_df <- as.data.frame(gap_stat$Tab)

  gap_diff_list <- vector()
  gap_val_list <- gap_df$gap
  s_val_list <- gap_df$SE.sim

  for (k in 1:max_k) {
    if (k < max_k - 1) {
      val <- gap_val_list[k] -
        (gap_val_list[k + 1] -
          s_val_list[k + 1])

      gap_diff_list <- append(gap_diff_list, val)
    }
  }

  return(gap_diff_list)
}

max_k <- 10
gap_stat <- clusGap(mtcars, hcut, K.max = max_k, B = 500)
```

```r
gap_diff_list <- get_cluster_diff(gap_stat, 10)
pos_neg_df <- data.frame(cluster = factor(seq_along(gap_diff_list)),
                         gap_diff = gap_diff_list)

ggplot(data = pos_neg_df, aes(x = cluster, y = gap_diff)) +
  geom_bar(stat = "identity", fill = "#4494a4") +
  xlab("Number of clusters K") +
  ylab("Gap(k) - (Gap(k+1) - Sk+1)") +
  ggtitle("Gap Statistic for Hierarchical Clustering") +
  theme_classic()
```



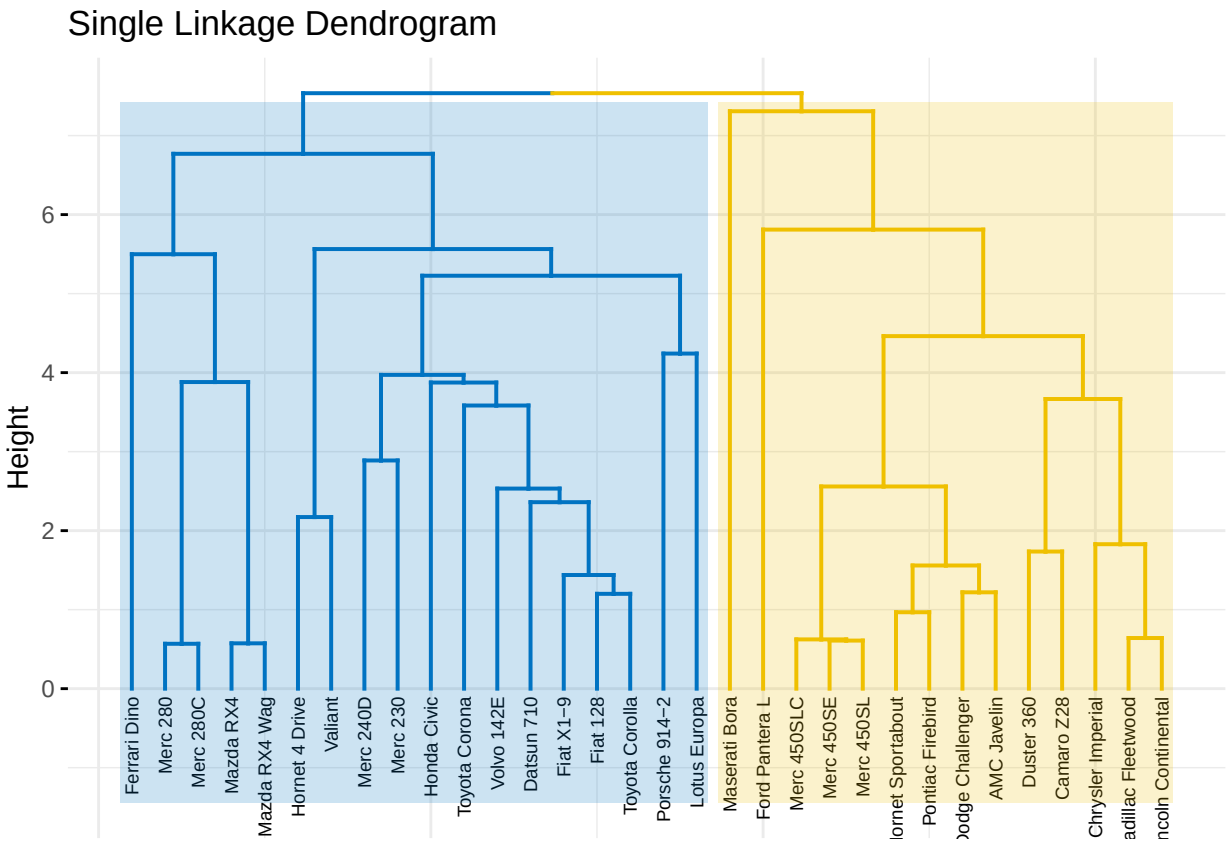
Both the Silhouette Width graph and the Gap Statistic graph indicates that the optimal number of clusters is k=2. In the case of our Silhouette Width graph, it is the point where the value of the average silhouette width is the highest and similarly for the gap statistic graph, this conclusion is drawn from observing the most significant change in the gap statistic value at this point.

4. Clustering Analysis

4.1. Agglomerative Hierarchical Clustering

4.1.1. Single Linkage Method

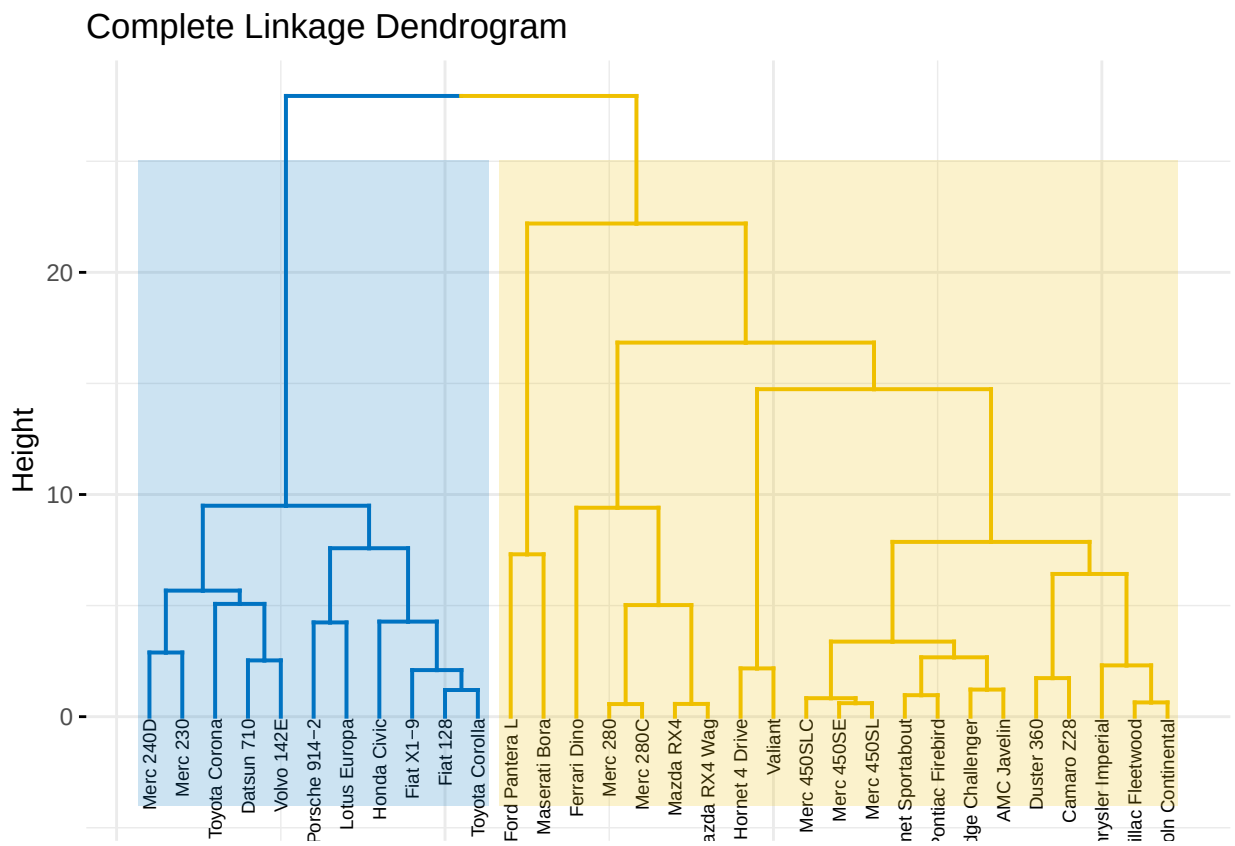
```r
res.dist <- dist(mtcars, method = "manhattan")

hc_single <- hclust(d = res.dist, method = "single")
grp_single <- cutree(hc_single, k = 2)

single_cluster <- fviz_cluster(
  list(data = mtcars, cluster = grp_single),
  palette = "jco",
  geom = "point",
  ellipse.type = "convex",
  show.clust.cent = FALSE,
  main = "Single Linkage Cluster",
  ggtheme = theme_minimal()
)

single_dendrogram <- fviz_dend(
  hc_single,
  cex = 0.5,
  k = 2,
  k_colors = "jco",
  rect = TRUE,
  rect_border = "jco",
  rect_fill = TRUE,
  label_cols = "black",
  label_cex = 0.5,
  main = "Single Linkage Dendrogram",
  ggtheme = theme_minimal()
)
single_dendrogram
```



4.1.2. Complete Linkage Method

```
hc_complete <- hclust(d = res.dist, method = "complete")
grp_complete <- cutree(hc_complete, k = 2)

complete_cluster <- fviz_cluster(
  list(data = mtcars, cluster = grp_complete),
  palette = "jco",
  geom = "point",
  ellipse.type = "convex",
  show.clust.cent = FALSE,
  main = "Complete Linkage Cluster",
  ggtheme = theme_minimal()
)

complete_dendrogram <- fviz_dend(
  hc_complete,
  cex = 0.5,
  k = 2,
  k_colors = "jco",
  rect = TRUE,
  rect_border = "jco",
  rect_fill = TRUE,
  label_cols = "black",
```

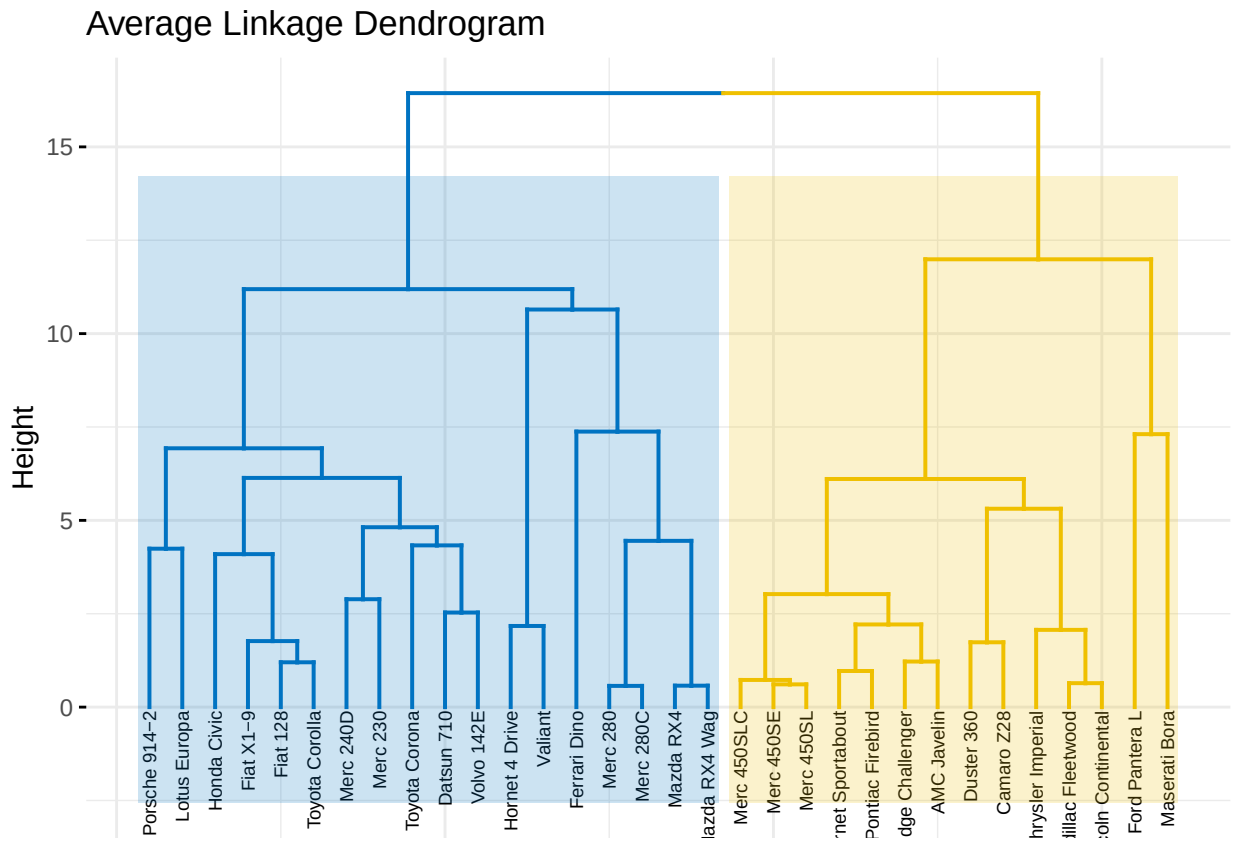
```
  label_cex = 0.5,
  main = "Complete Linkage Dendrogram",
  ggtheme = theme_minimal()
)
complete_dendrogram
```



4.1.3. Average Linkage Method

```
hc_average <- hclust(d = res.dist, method = "average")
grp_average <- cutree(hc_average, k = 2)

average_cluster <- fviz_cluster(
  list(data = mtcars, cluster = grp_average),
  palette = "jco",
  geom = "point",
  ellipse.type = "convex",
  show.clust.cent = FALSE,
  main = "Average Linkage Cluster",
  ggtheme = theme_minimal()
)

average_dendrogram <- fviz_dend(
  hc_average,
```

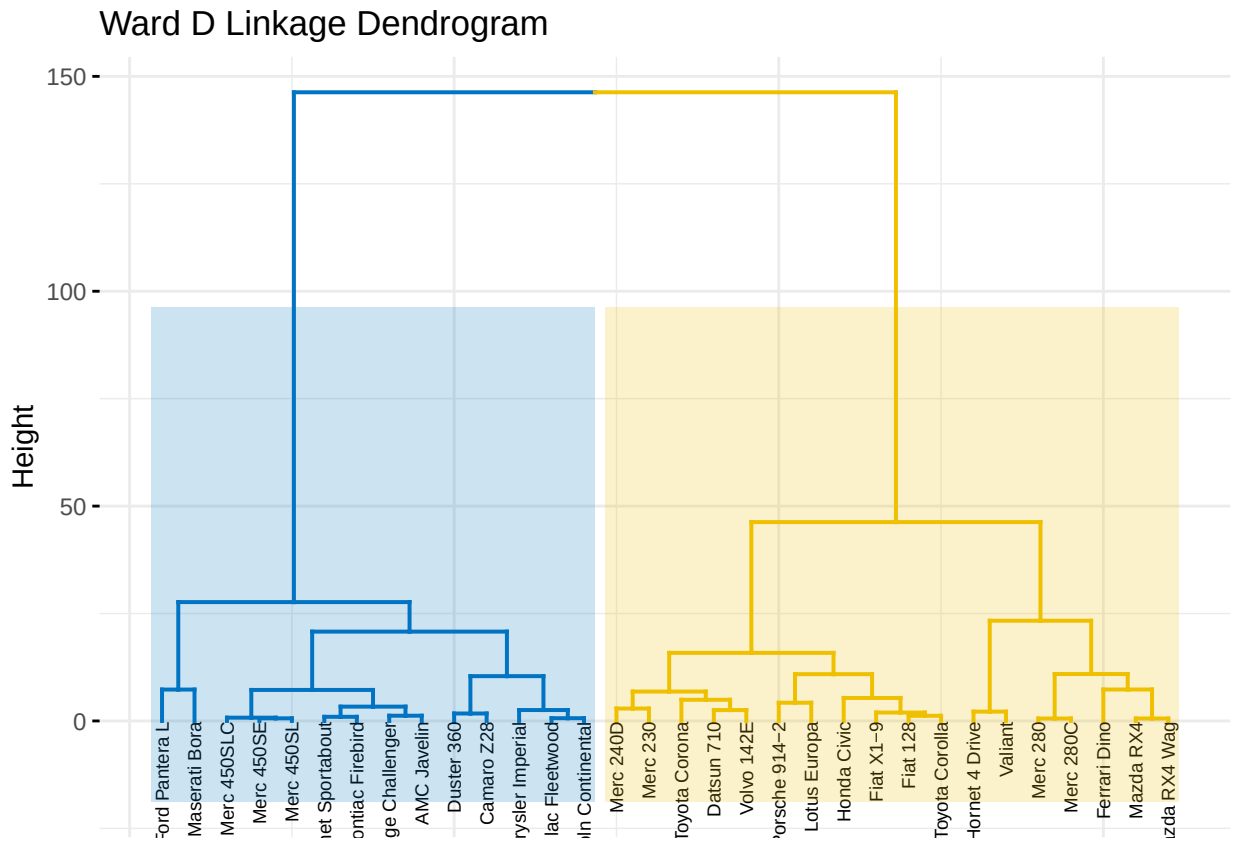
```
  cex = 0.5,
  k = 2,
  k_colors = "jco",
  rect = TRUE,
  rect_border = "jco",
  rect_fill = TRUE,
  label_cols = "black",
  label_cex = 0.5,
  main = "Average Linkage Dendrogram",
  ggtheme = theme_minimal()
)
average_dendrogram
```



4.1.4. Ward Linkage Method

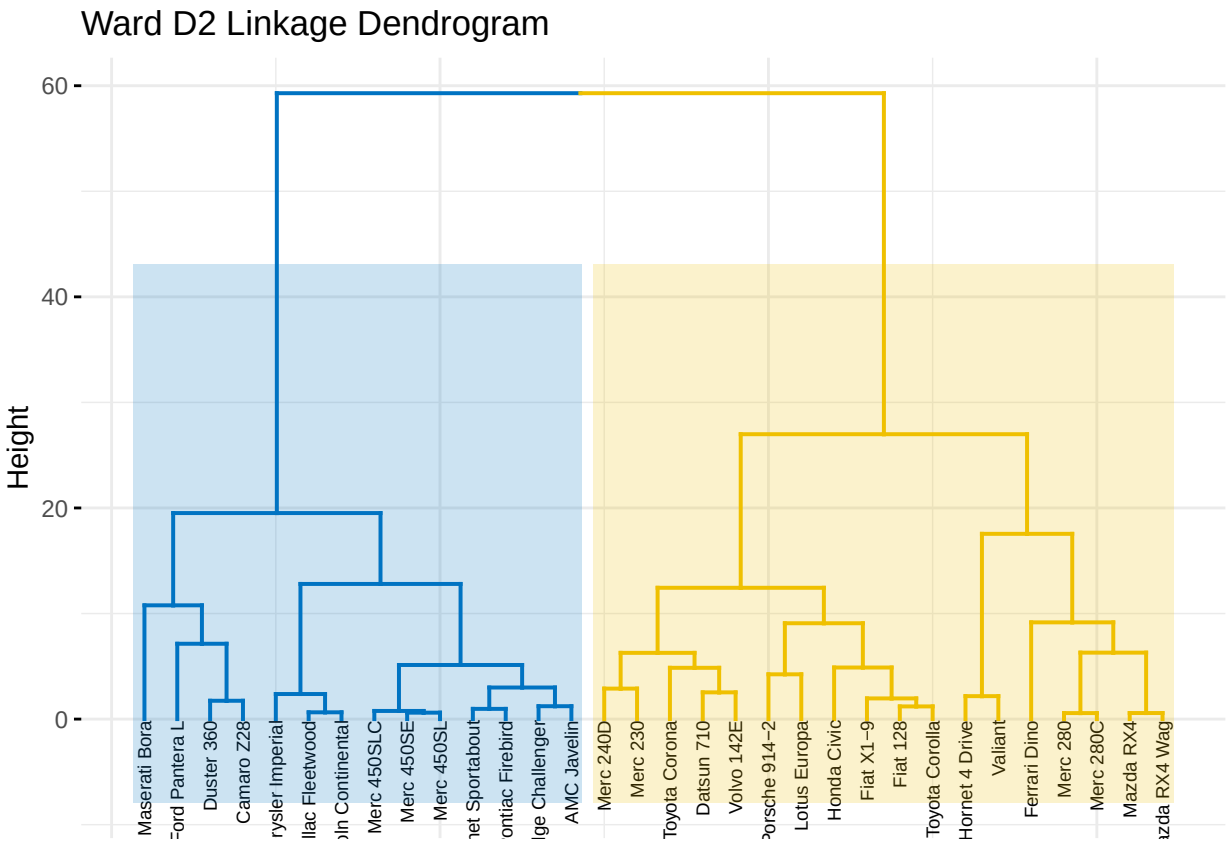
```
hc_ward_d <- hclust(d = res.dist, method = "ward.D")
grp_ward_d <- cutree(hc_ward_d, k = 2)

ward_d_cluster <- fviz_cluster(
  list(data = mtcars, cluster = grp_ward_d),
  palette = "jco",
  geom = "point",
  ellipse.type = "convex",
```

```
  show.clust.cent = FALSE,
  main = "Ward D Linkage Cluster",
  ggtheme = theme_minimal()
)

ward_d_dendrogram <- fviz_dend(
  hc_ward_d,
  cex = 0.5,
  k = 2,
  k_colors = "jco",
  rect = TRUE,
  rect_border = "jco",
  rect_fill = TRUE,
  label_cols = "black",
  label_cex = 0.5,
  main = "Ward D Linkage Dendrogram",
  ggtheme = theme_minimal()
)
ward_d_dendrogram
```



4.1.5. Ward.D2 Method

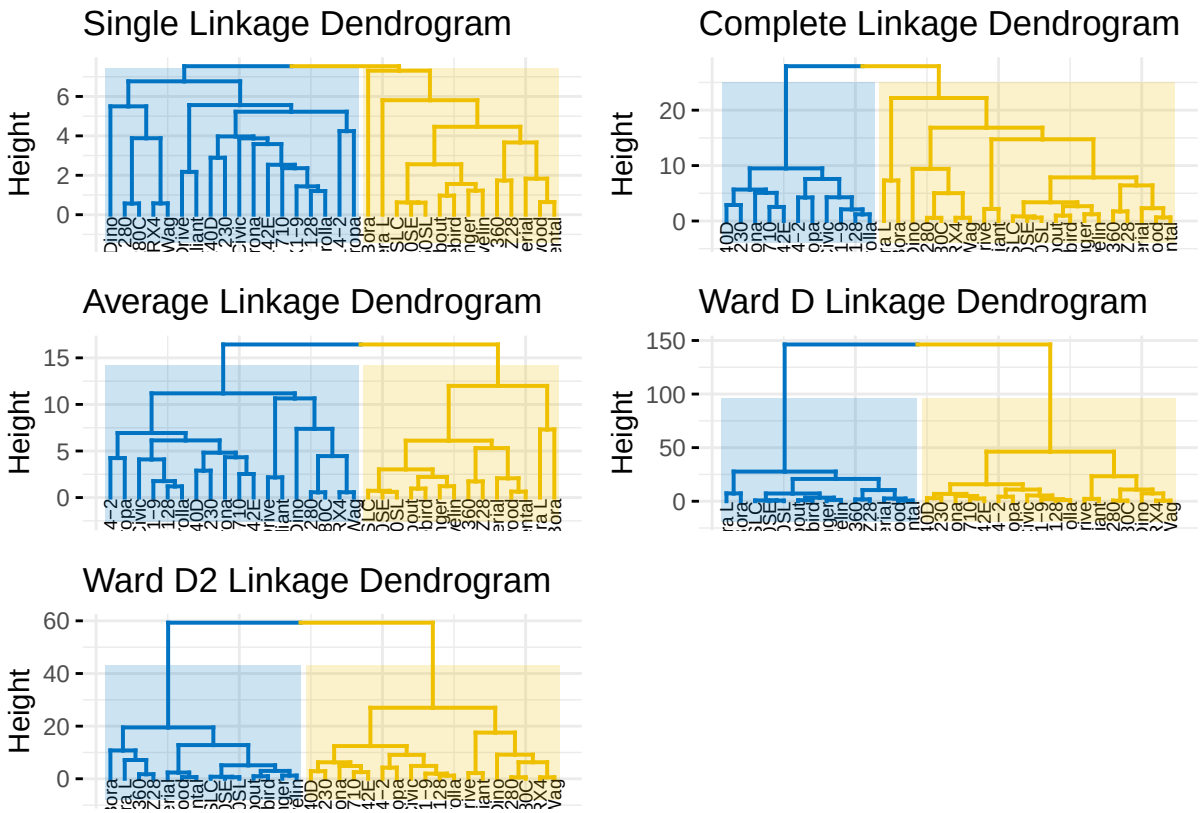
```r
hc_ward_d2 <- hclust(d = res.dist, method = "ward.D2")
grp_ward_d2 <- cutree(hc_ward_d2, k = 2)

ward_d2_cluster <- fviz_cluster(
  list(data = mtcars, cluster = grp_ward_d2),
  palette = "jco",
  geom = "point",
  ellipse.type = "convex",
  show.clust.cent = FALSE,
  main = "Ward D2 Linkage Cluster",
  ggtheme = theme_minimal()
)

ward_d2_dendrogram <- fviz_dend(
  hc_ward_d2,
  cex = 0.5,
  k = 2,
  k_colors = "jco",
  rect = TRUE,
  rect_border = "jco",
  rect_fill = TRUE,
  label_cols = "black",
  label_cex = 0.5,
  main = "Ward D2 Linkage Dendrogram",
  ggtheme = theme_minimal()
)
ward_d2_dendrogram
```



4.2 Comparing Linkage Methods

4.2.1. Comparing Dendrograms

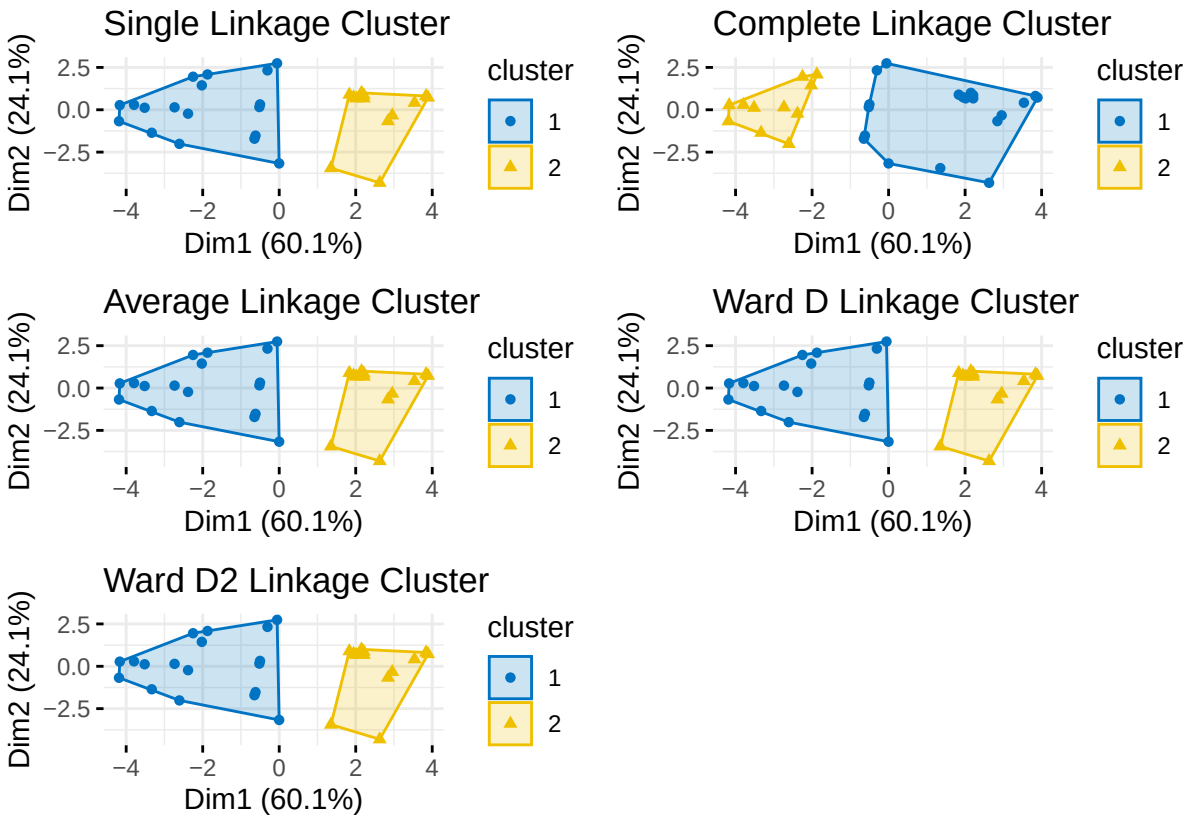
```
single_dendrogram +
  complete_dendrogram +
  average_dendrogram +
  ward_d_dendrogram +
  ward_d2_dendrogram +
  plot_layout(ncol = 2)
```



Ward D and Ward D2 linkage methods show the highest height between clusters when they are joined, indicating good separability between the clusters. The Average linkage method also shows moderate separability between clusters. In contrast, the Complete linkage method demonstrates a greater disparity in the height at which clusters are joined, suggesting less uniformity in cluster separation. The Single linkage method typically has the lowest height for merging clusters, which implies that the clusters are less well-separated compared to the other methods.

4.2.2. Comparing Clusters

```
single_cluster +
  complete_cluster +
  average_cluster +
  ward_d_cluster +
  ward_d2_cluster +
  plot_layout(ncol = 2)
```



When comparing the clusters visually, we observe that all linkage methods produce similar results except for the Complete linkage method. This suggests that Complete linkage provides a unique clustering outcome compared to the other methods. The fact that four out of the five methods yield similar results reinforces the reliability of the clustering result, indicating that the observed clustering pattern is robust and not solely dependent on the linkage method.

4.2.3. Correlation between Cophenetic Distance and the Original Distance

```
cor(res.dist, cophenetic(hc_single))
```

```
## [1] 0.7675238
```

```
cor(res.dist, cophenetic(hc_complete))
```

```
## [1] 0.7858564
```

```
cor(res.dist, cophenetic(hc_average))
```

```
## [1] 0.8010725
```

```
cor(res.dist, cophenetic(hc_ward_d))
```

```
## [1] 0.7494353
```

```r
cor(res.dist, cophenetic(hc_ward_d2))
```

```
## [1] 0.7667039
```

This suggests that the cophenetic distances are fairly consistent with the original pairwise distances in the distance matrix.

The high correlation coefficient of all methods demonstrates that all the methods effectively preserves the original distance relationships between data points. This validates the robustness of the clustering results and confirms that they are all reliable approaches.

4.2.4. Correlation Matrix

```r
dend_complete <- mtcars %>%
  dist %>%
  hclust("complete") %>%
  as.dendrogram
dend_single <- mtcars %>%
  dist %>%
  hclust("single") %>%
  as.dendrogram
dend_average <- mtcars %>%
  dist %>%
  hclust("average") %>%
  as.dendrogram
dend_ward <- mtcars %>%
  dist %>%
  hclust("ward.D") %>%
  as.dendrogram
dend_ward2 <- mtcars %>%
  dist %>%
  hclust("ward.D2") %>%
  as.dendrogram
```

```r
dend_list <- dendlist(
  "Complete" = dend_complete,
  "Single" = dend_single,
  "Average" = dend_average,
  "WardD" = dend_ward,
  "WardD2" = dend_ward2
)

cors_cophenetic <- cor.dendlist(dend_list, method = "cophenetic")

round(cors_cophenetic, 2)
```

```
##          Complete Single Average WardD WardD2
## Complete     1.00   0.84    0.79  0.59   0.65
## Single       0.84   1.00    0.92  0.60   0.68
## Average      0.79   0.92    1.00  0.80   0.86
## WardD        0.59   0.60    0.80  1.00   0.99
## WardD2       0.65   0.68    0.86  0.99   1.00
```

The cophenetic correlation measures how similarly the clustering patterns are preserved in terms of pairwise distances. In other words, this correlation indicates how well the hierarchical clustering preserves the pairwise distances between observations.

High Correlation Values:

- WardD and WardD2: The highest correlation (0.99) suggests that these two methods produce very similar dendrograms. They preserve the distance structure between clusters nearly identically.
- Single and Average: Also show high correlations (0.92), indicating that these methods provide similar hierarchical clusterings, though not as closely related as WardD and WardD2.

Moderate Correlation Values:

- Complete vs. Single (0.84): These methods are somewhat similar, though Complete is less similar to others compared to Single.
- Average vs. WardD (0.80): Average and WardD have moderate similarity, reflecting some commonality but also noticeable differences.

Lower Correlation Values:

- Complete vs. WardD (0.59): Shows a lower correlation, indicating that Complete and WardD dendrograms are quite different.
- Complete vs. WardD2 (0.65): Slightly higher than with WardD, but still relatively low compared to other pairs.

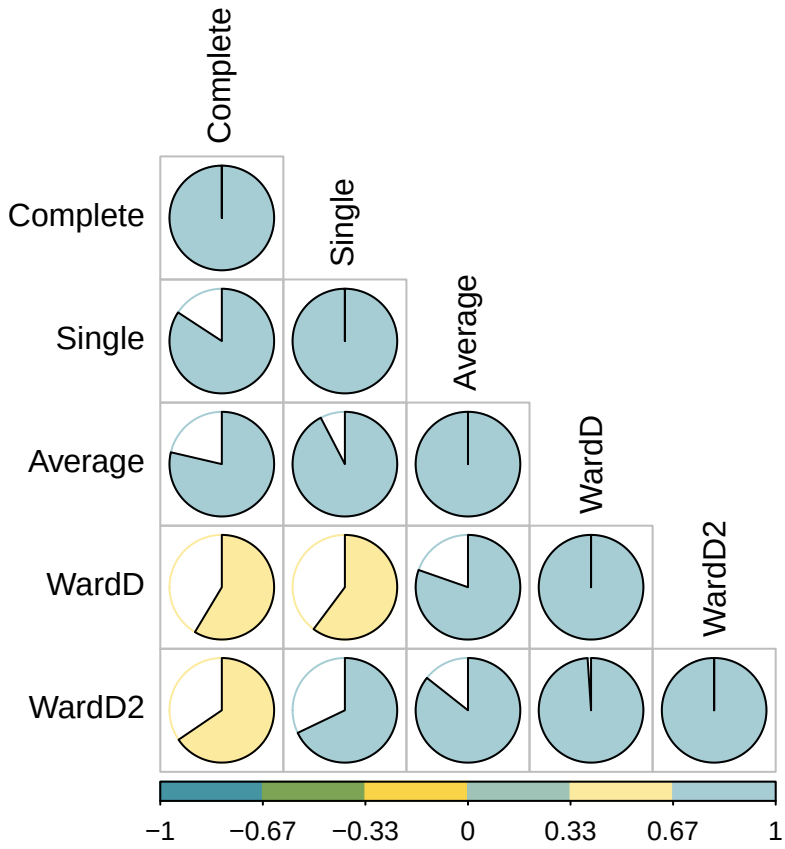
```
cors_baker <- cor.dendlist(dend_list, method = "baker")

round(cors_baker, 2)
```

```
##          Complete Single Average WardD WardD2
## Complete     1.00   0.85    0.61  0.53   0.55
## Single       0.85   1.00    0.76  0.60   0.63
## Average      0.61   0.76    1.00  0.85   0.90
## WardD        0.53   0.60    0.85  1.00   0.98
## WardD2       0.55   0.63    0.90  0.98   1.00
```

The Baker cophenetic correlation measures the similarity in the structure of dendrograms, indicating how similar the overall structures of different dendrograms are.

- **High Correlation:** WardD and WardD2 (0.98), Average and WardD (0.85), Average and WardD2 (0.90). These values indicate that the dendrogram structures are very similar.
- **Moderate Correlation:** Complete and Single (0.85), Single and Average (0.76). These values suggest that the dendrogram structures are somewhat similar but not as closely aligned as those with high correlation.

```
corrplot(cors_cophenetic, method = "pie", type = "lower", col = color_palette, tl.col = "black")
```



- **Strong Positive Correlations:** WardD and WardD2 are very similar to each other, indicating that the results from these methods are almost identical.
- **Moderate Positive Correlations:** Single Linkage and Average Linkage are more similar to each other compared to their similarity with WardD and WardD2.
- **Weak or No Correlation:** Complete Linkage has moderate similarity with Single Linkage and Average Linkage, but is less similar to WardD and WardD2.
- **Negative Correlations:** There are no cells shaded in red, indicating no negative correlations between the linkage methods.

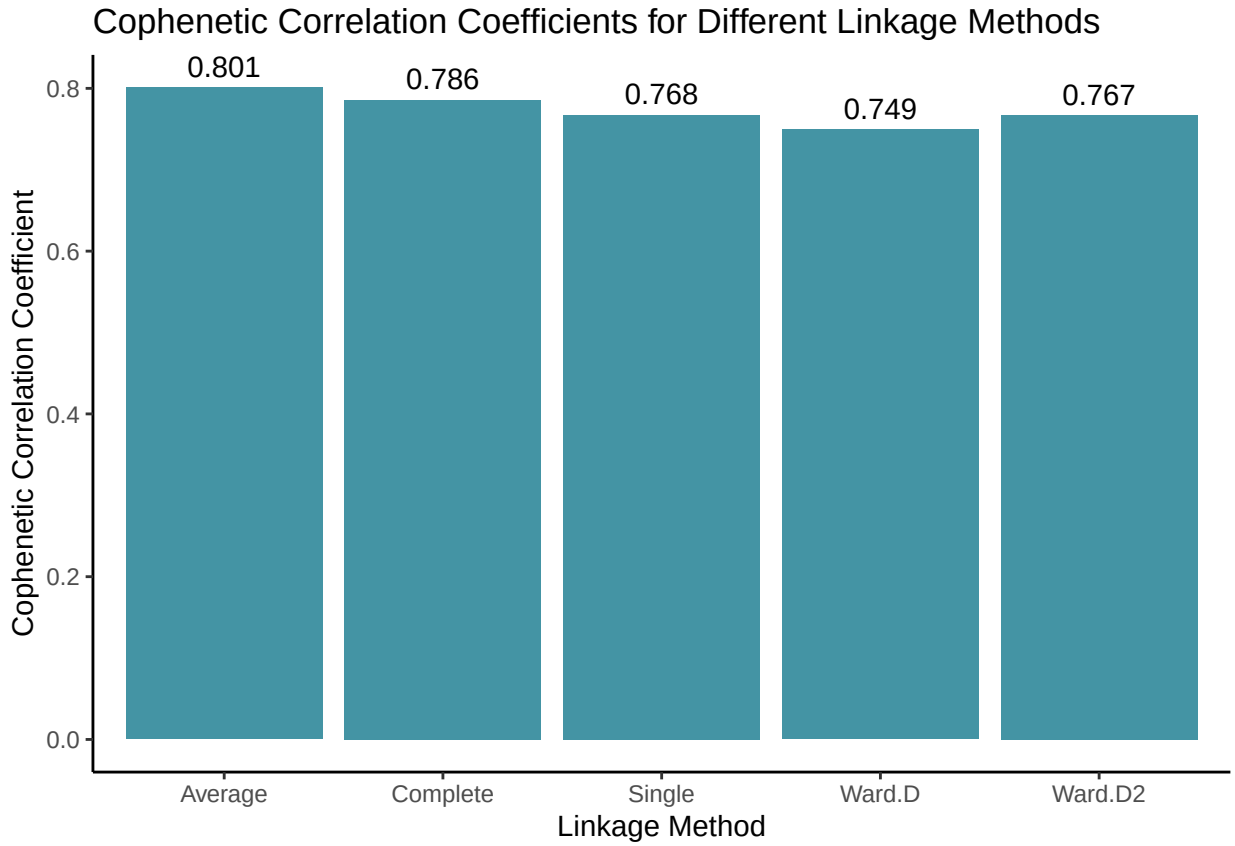
4.2.5. Finding Best Linkage Method

```
linkage_methods <- c("Single", "Complete", "Average", "Ward.D", "Ward.D2")

cophenetic_correlations <- c(
  cor(res.dist, cophenetic(hc_single)),
  cor(res.dist, cophenetic(hc_complete)),
  cor(res.dist, cophenetic(hc_average)),
  cor(res.dist, cophenetic(hc_ward_d)),
  cor(res.dist, cophenetic(hc_ward_d2))
)

cophenetic_df <- data.frame(
  linkage_method = linkage_methods,
  cophenetic_correlation = cophenetic_correlations
```

```
)

cophenetic_df
```

```
##   linkage_method cophenetic_correlation
## 1         Single              0.7675238
## 2       Complete              0.7858564
## 3        Average              0.8010725
## 4         Ward.D              0.7494353
## 5        Ward.D2              0.7667039
```

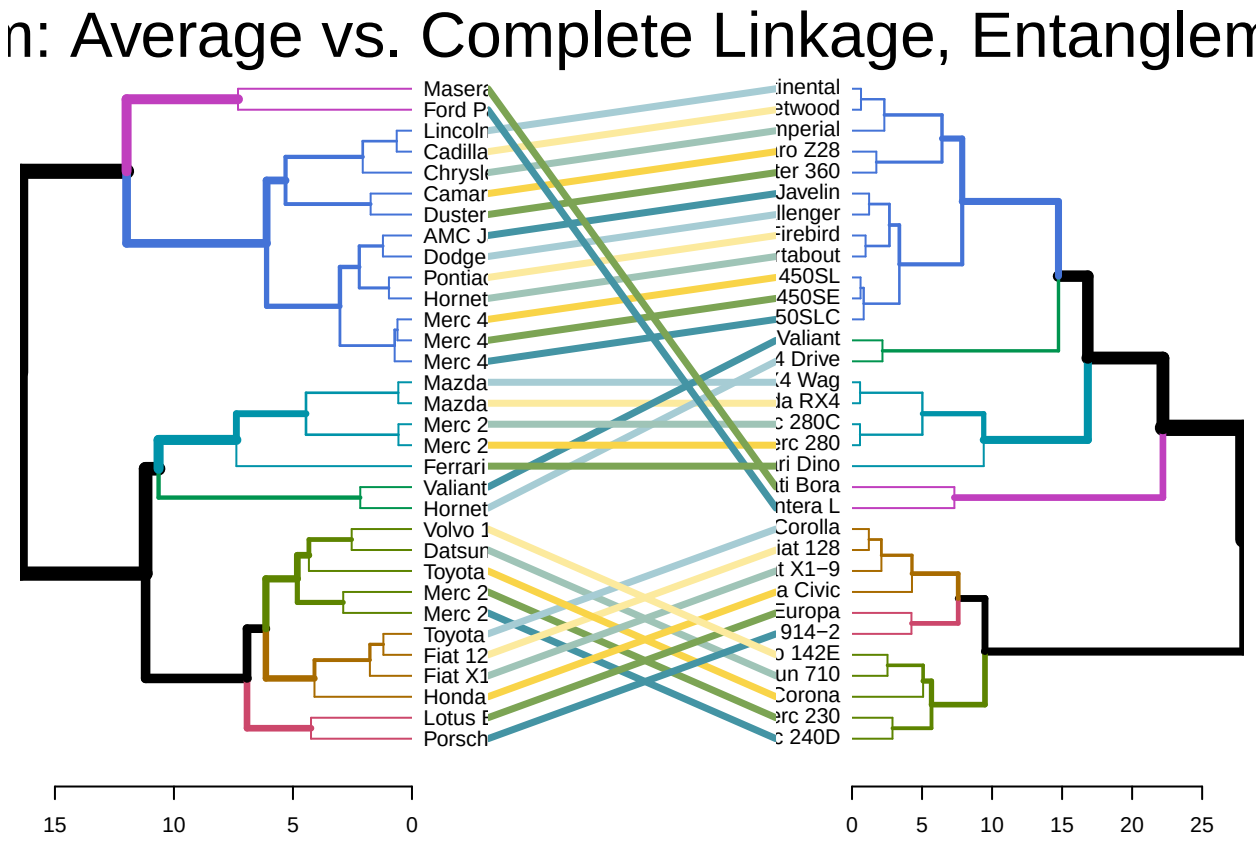
```
ggplot(cophenetic_df, aes(x = linkage_method, y = cophenetic_correlation)) +
  geom_bar(stat = "identity", fill = "#4494a4") +
  theme_classic() +
  labs(
    title = "Cophenetic Correlation Coefficients for Different Linkage Methods",
    x = "Linkage Method",
    y = "Cophenetic Correlation Coefficient"
  ) +
  geom_text(aes(label = round(cophenetic_correlation, 3)), vjust = -0.5)
```



In this case, the Average linkage method has the highest cophenetic correlation coefficient (0.8010725), followed closely by the Complete linkage method (0.7858564). These methods are the best fit for our data based on the cophenetic correlation coefficient.

4.2.6. Tanglegram of the Average Linkage Method vs. Complete Linkage Method

```
average_dendrogram <- as.dendrogram(hc_average)
complete_dendrogram <- as.dendrogram(hc_complete)

tanglegram(
  average_dendrogram,
  complete_dendrogram,
  color_lines = color_palette,
  highlight_distinct_edges = FALSE, # Turn-off dashed lines
  common_subtrees_color_lines = FALSE, # Turn-off line colors
  common_subtrees_color_branches = TRUE, # Color common branches
  main = paste("Tanglegram: Average vs. Complete Linkage, Entanglement =",
               round(entanglement(dendlist(average_dendrogram, complete_dendrogram)), 2))
)
```



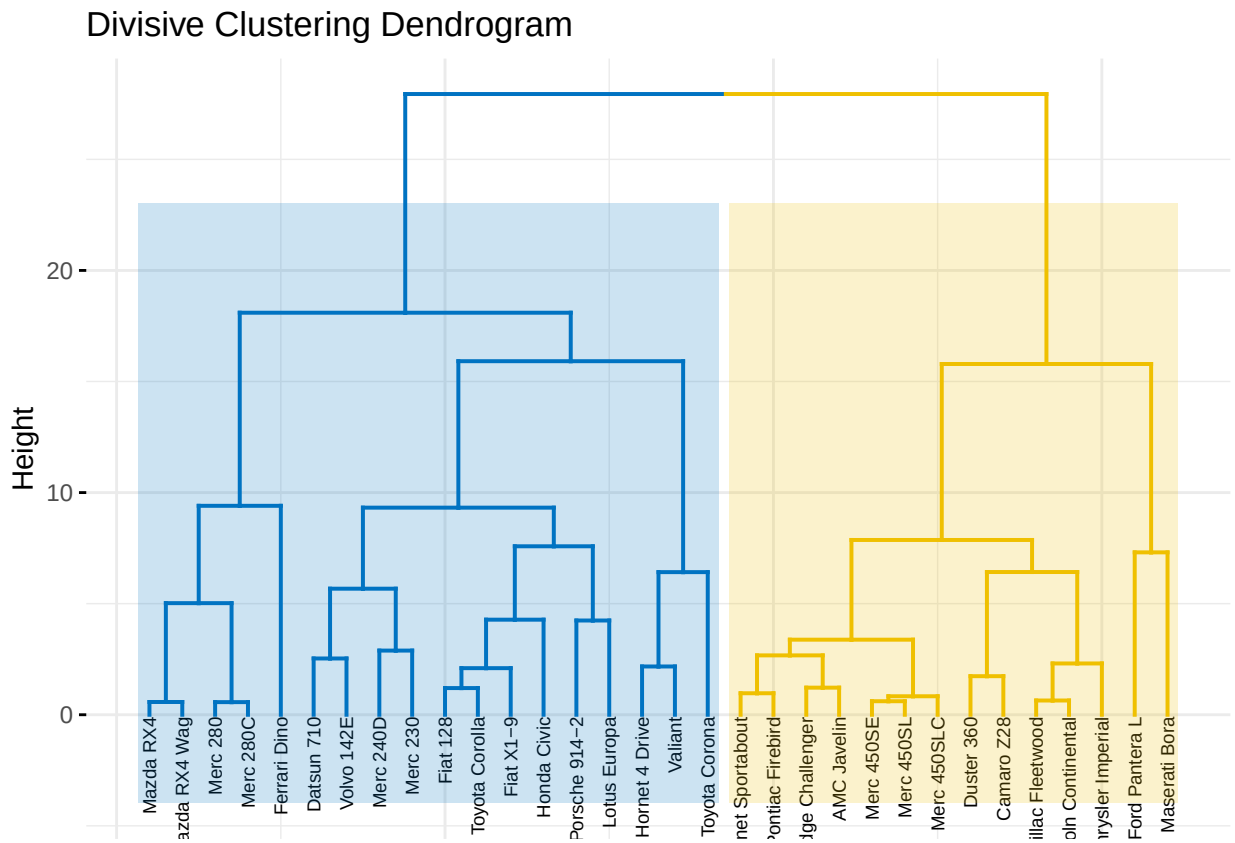
We observe that car models like the AMC Javelin, Dodge Challenger, and Pontiac Firebird are connected by very straight and untangled lines, indicating that both the Average Linkage and Complete Linkage methods cluster these models similarly. This consistency suggests that these models share strong similarities, such as engine type, body style, or performance metrics, which are robustly identified by both methods.

In contrast, the rest of the nodes exhibit significant tangling, showing that most car models are clustered differently by the two methods. The tangled lines indicate variability in how the Average Linkage and Complete Linkage methods interpret distances between these models, leading to different cluster formations. This tangling for the majority of the car models suggests that their similarities are less pronounced or more complex, making their clustering less stable across different methods.

4.3. Alternative Hierarchical Clustering Methods

4.3.1. Divisive Analysis Clustering

```
res.diana <- diana(
  x = mtcars,
  metric = "manhattan"
)

divisive_dendrogram <- fviz_dend(
  res.diana,
  cex = 0.5,
  k = 2,
  k_colors = "jco",
  rect = TRUE,
  rect_border = "jco",
  rect_fill = TRUE,
  label_cols = "black",
  label_cex = 0.5,
  main = "Divisive Clustering Dendrogram",
  ggtheme = theme_minimal()
)
divisive_dendrogram
```

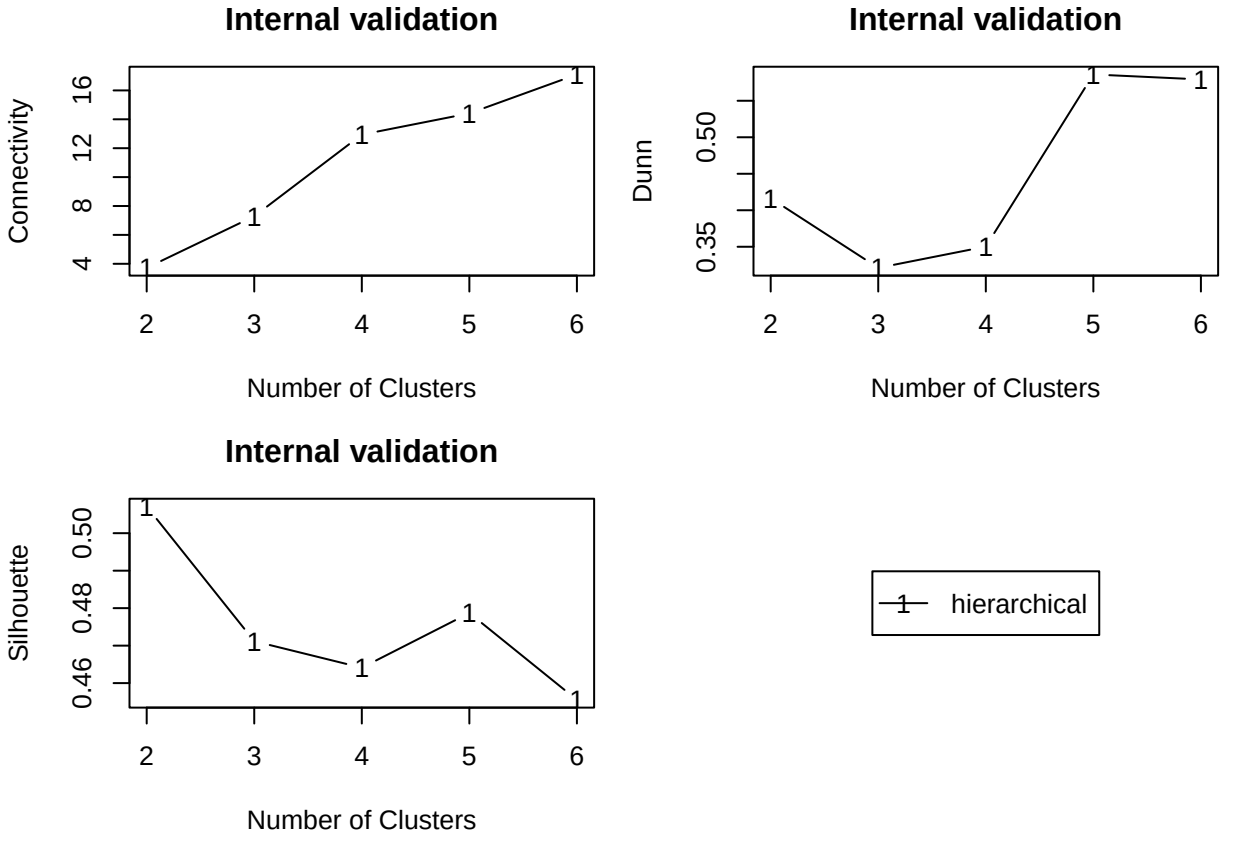


5. Cluster Validation

5.1 Internal Validation

5.1.1 Average Linkage Method

```
intern_mtcars <- clValid(mtcars, 2:6, clMethods = "hierarchical",
                         validation = "internal",
                         method = "average",
                         metric = "manhattan")

par(mfrow = c(2, 2), mar = c(4, 4, 3, 1))

plot(intern_mtcars, legend = FALSE)

plot(nClusters(intern_mtcars),
     measures(intern_mtcars, "Dunn")[, , 1], type = "n", axes = FALSE, xlab = "", ylab = "")
legend("center", clusterMethods(intern_mtcars), col = 1:9, lty = 1:9, pch = paste(1:9))
```



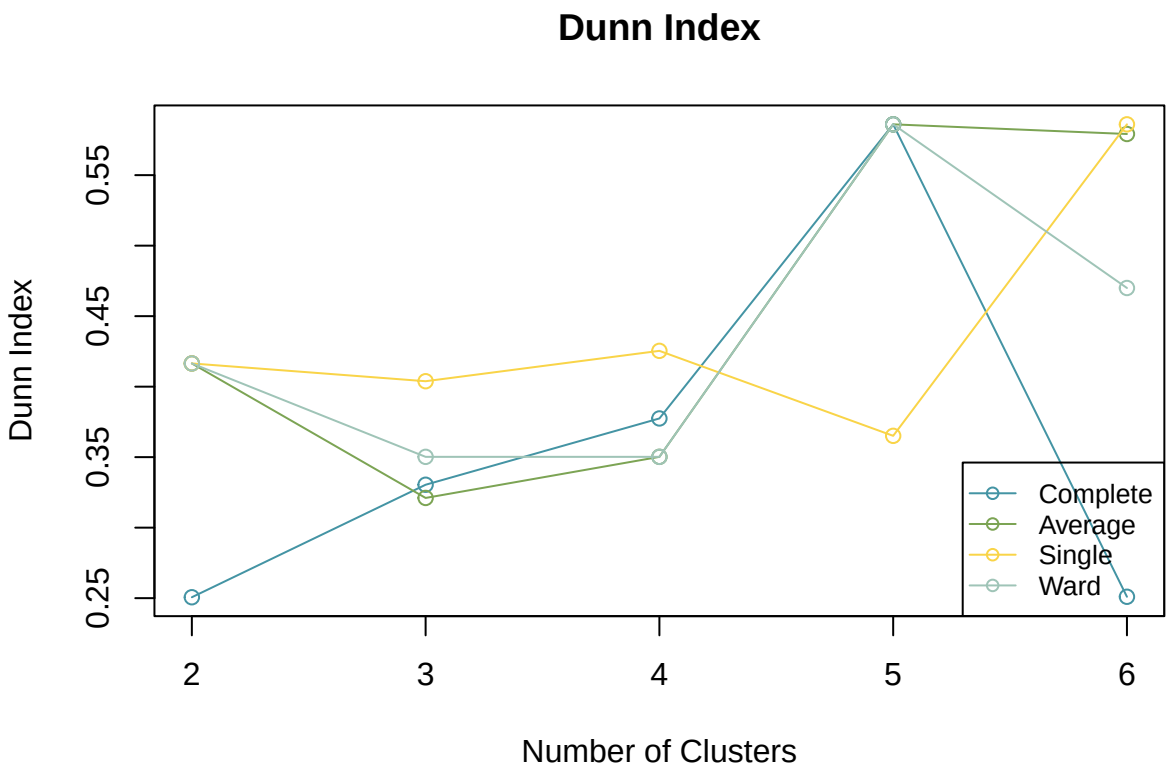
- **Connectivity:** According to connectivity, the optimal number of clusters is 2 as that is where it is lowest.
- **Dunn:** According to Dunn index, the optimal number of clusters is 5 as it is highest at k=5.
- **Silhouette:** According to Silhouette width, the optimal number of clusters is 2 as it is highest at k=2.

Overall, we can conclude that, based on these three validation methods, we should choose the number of clusters as 2.

5.1.2. Internal Validation in Comparison with the Linkages Method

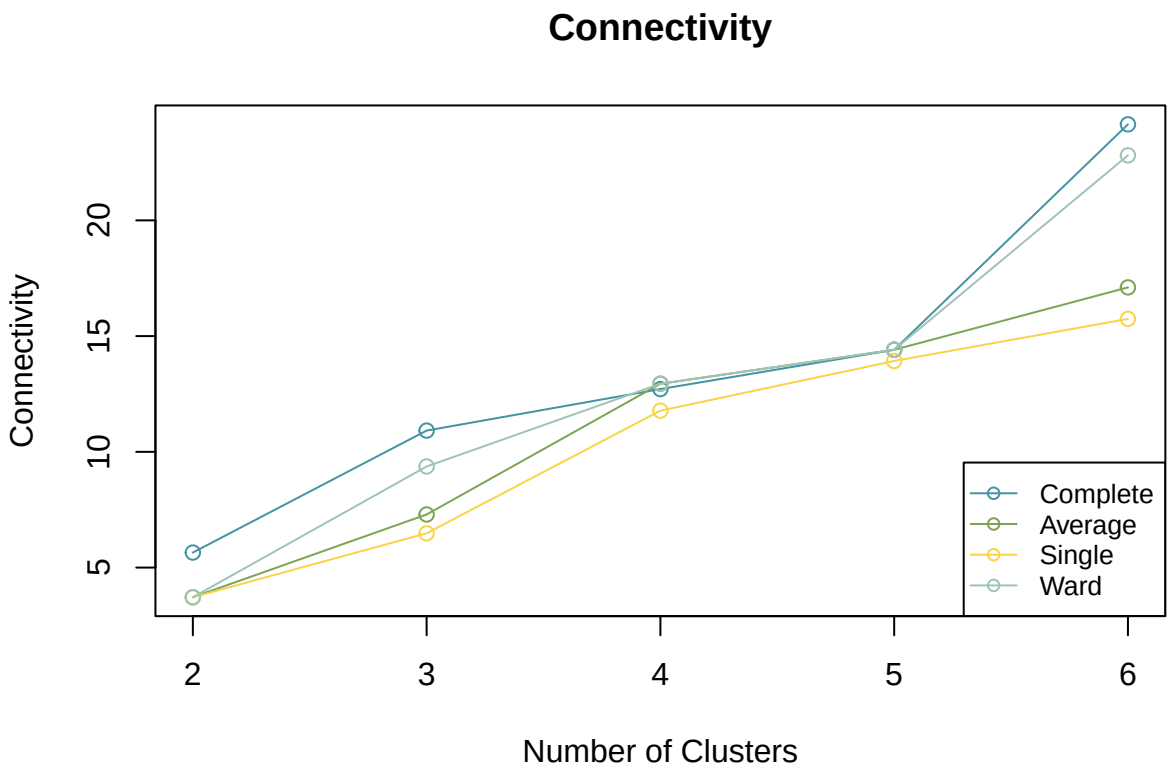
```
intern_complete <- clValid(mtcars, 2:6, clMethods = "hierarchical",
                           validation = "internal", method = "complete",
                           metric = "manhattan")
intern_average <- clValid(mtcars, 2:6, clMethods = "hierarchical",
                          validation = "internal", method = "average",
                          metric = "manhattan")
intern_single <- clValid(mtcars, 2:6, clMethods = "hierarchical",
                         validation = "internal", method = "single",
                         metric = "manhattan")
intern_ward <- clValid(mtcars, 2:6, clMethods = "hierarchical",
                       validation = "internal", method = "ward",
                       metric = "manhattan")

num_clusters <- nClusters(intern_complete)

dunn_complete <- measures(intern_complete, "Dunn")[, , 1]
dunn_average <- measures(intern_average, "Dunn")[, , 1]
dunn_single <- measures(intern_single, "Dunn")[, , 1]
dunn_ward <- measures(intern_ward, "Dunn")[, , 1]

connectivity_complete <- measures(intern_complete, "Connectivity")[, , 1]
connectivity_average <- measures(intern_average, "Connectivity")[, , 1]
connectivity_single <- measures(intern_single, "Connectivity")[, , 1]
connectivity_ward <- measures(intern_ward, "Connectivity")[, , 1]

silhouette_complete <- measures(intern_complete, "Silhouette")[, , 1]
silhouette_average <- measures(intern_average, "Silhouette")[, , 1]
silhouette_single <- measures(intern_single, "Silhouette")[, , 1]
silhouette_ward <- measures(intern_ward, "Silhouette")[, , 1]
```

```
# Plot Dunn Index
plot(num_clusters, dunn_complete, type = "o", col = "#4494a4",
     ylim = range(dunn_complete, dunn_average, dunn_single, dunn_ward),
     xlab = "Number of Clusters", ylab = "Dunn Index", main = "Dunn Index")
lines(num_clusters, dunn_average, type = "o", col = "#7ca454")
lines(num_clusters, dunn_single, type = "o", col = "#f9d448")
lines(num_clusters, dunn_ward, type = "o", col = "#9fc4b7")
legend("bottomright", legend = c("Complete", "Average", "Single", "Ward"),
       col = c("#4494a4", "#7ca454", "#f9d448", "#9fc4b7"), lty = 1, pch = 1, cex = 0.8)
```



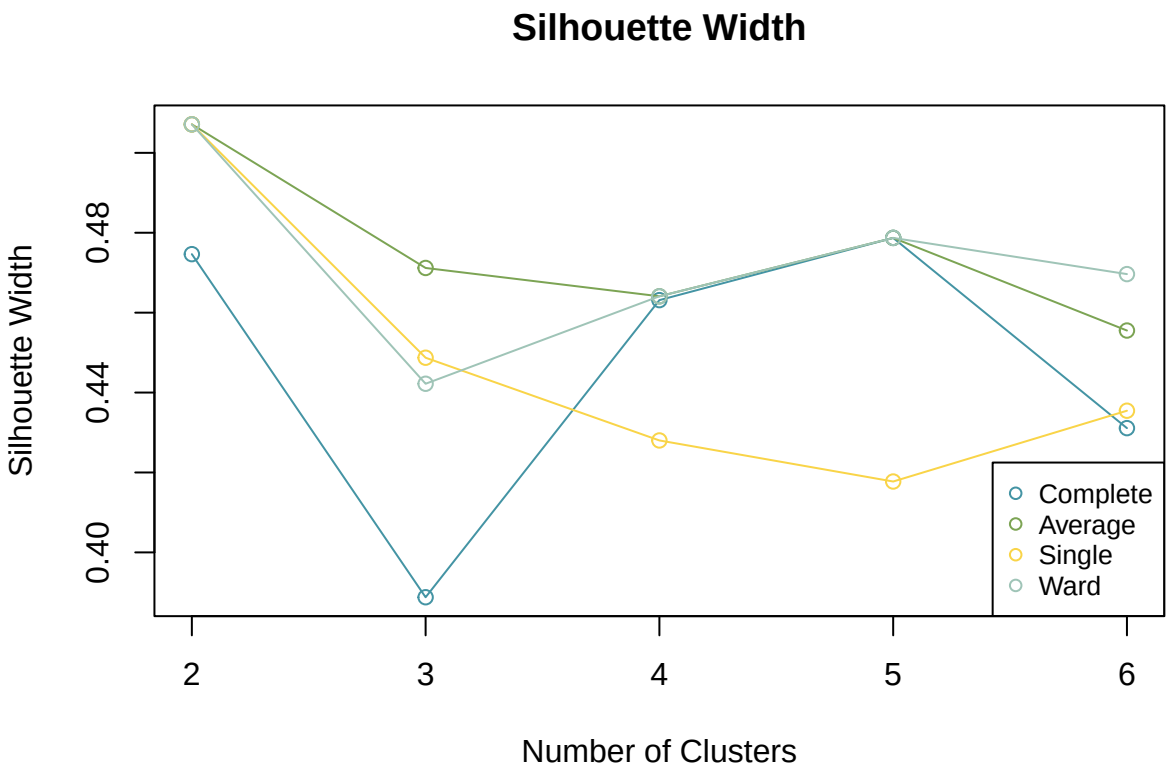
Dunn Index: All methods report the same Dunn Index value (0.5860) for different numbers of clusters (5 for Complete, Average, and Ward; 6 for Single). Since the Dunn Index values are the same across methods, it does not provide a clear distinction in terms of separation quality.

```
# Plot Connectivity
plot(num_clusters, connectivity_complete, type = "o", col = "#4494a4",
     ylim = range(connectivity_complete, connectivity_average,
                  connectivity_single, connectivity_ward),
     xlab = "Number of Clusters", ylab = "Connectivity", main = "Connectivity")
lines(num_clusters, connectivity_average, type = "o", col = "#7ca454")
lines(num_clusters, connectivity_single, type = "o", col = "#f9d448")
lines(num_clusters, connectivity_ward, type = "o", col = "#9fc4b7")
legend("bottomright", legend = c("Complete", "Average", "Single", "Ward"),
       col = c("#4494a4", "#7ca454", "#f9d448", "#9fc4b7"), lty = 1, pch = 1, cex = 0.8)
```



Connectivity: The Average, Single, and Ward methods have the same Connectivity score (3.7179) for 2 clusters, which is lower than the Complete method's score (5.6488). This suggests that the Average, Single, and Ward methods provide more compact clusters compared to the Complete method.

```
# Plot Silhouette Width
plot(num_clusters, silhouette_complete, type = "o", col = "#4494a4",
     ylim = range(silhouette_complete, silhouette_average,
                  silhouette_single, silhouette_ward),
     xlab = "Number of Clusters", ylab = "Silhouette Width", main = "Silhouette Width")
lines(num_clusters, silhouette_average, type = "o", col = "#7ca454")
lines(num_clusters, silhouette_single, type = "o", col = "#f9d448")
lines(num_clusters, silhouette_ward, type = "o", col = "#9fc4b7")
legend("bottomright", legend = c("Complete", "Average", "Single", "Ward"),
       col = c("#4494a4", "#7ca454", "#f9d448", "#9fc4b7"), lty1, pch = 1, cex = 0.8)
```



Silhouette Width: The Average, Single, and Ward methods all have a higher Silhouette Width (0.5071 for 2 clusters) compared to the Complete method (0.4787 for 5 clusters). This indicates that the Average, Single, and Ward methods provide better overall clustering in terms of separation and cohesion.

```
print("Summary of Complete Linkage Method:", summary(intern_complete))
```

```
##
## Clustering Methods:
##  hierarchical
##
## Cluster sizes:
##  2 3 4 5 6
##
## Validation Measures:
##                                  2       3       4       5       6
##
## hierarchical Connectivity   5.6488 10.9179 12.7107 14.4091 24.1480
##              Dunn           0.2506  0.3304  0.3774  0.5860  0.2510
##              Silhouette     0.4746  0.3888  0.4631  0.4787  0.4311
##
## Optimal Scores:
##
##              Score  Method       Clusters
## Connectivity 5.6488 hierarchical 2
## Dunn         0.5860 hierarchical 5
## Silhouette   0.4787 hierarchical 5
```

```
##
## [1] "Summary of Complete Linkage Method:"
```

```
print("Summary of Average Linkage Method:", summary(intern_average))
```

```
##
## Clustering Methods:
##  hierarchical
##
## Cluster sizes:
##  2 3 4 5 6
##
## Validation Measures:
##                                  2       3       4       5       6
##
## hierarchical Connectivity   3.7179  7.2925 12.9413 14.4091 17.1036
##              Dunn           0.4165  0.3211  0.3502  0.5860  0.5792
##              Silhouette     0.5071  0.4712  0.4641  0.4787  0.4555
##
## Optimal Scores:
##
##              Score  Method       Clusters
## Connectivity 3.7179 hierarchical 2
## Dunn         0.5860 hierarchical 5
## Silhouette   0.5071 hierarchical 2
##
## [1] "Summary of Average Linkage Method:"
```

```
print("Summary of Single Linkage Method:", summary(intern_single))
```

```
##
## Clustering Methods:
##  hierarchical
##
## Cluster sizes:
##  2 3 4 5 6
##
## Validation Measures:
##                                  2       3       4       5       6
##
## hierarchical Connectivity   3.7179  6.4802 11.7790 13.9246 15.7425
##              Dunn           0.4165  0.4038  0.4254  0.3651  0.5860
##              Silhouette     0.5071  0.4487  0.4280  0.4177  0.4355
##
## Optimal Scores:
##
##              Score  Method       Clusters
## Connectivity 3.7179 hierarchical 2
## Dunn         0.5860 hierarchical 6
## Silhouette   0.5071 hierarchical 2
##
## [1] "Summary of Single Linkage Method:"
```

```r
print("Summary of Ward Linkage Method:", summary(intern_ward))
```

```
##
## Clustering Methods:
##  hierarchical
##
## Cluster sizes:
##  2 3 4 5 6
##
## Validation Measures:
##                                  2       3       4       5       6
##
## hierarchical Connectivity   3.7179  9.3667 12.9413 14.4091 22.8087
##              Dunn           0.4165  0.3502  0.3502  0.5860  0.4700
##              Silhouette     0.5071  0.4422  0.4641  0.4787  0.4696
##
## Optimal Scores:
##
##              Score  Method       Clusters
## Connectivity 3.7179 hierarchical 2
## Dunn         0.5860 hierarchical 5
## Silhouette   0.5071 hierarchical 2
##
## [1] "Summary of Ward Linkage Method:"
```

Based on the three metrics, the Average Linkage method appears to be a suitable choice due to its:

- Lowest Connectivity score, indicating more compact clusters.
- Highest Silhouette Width score for 2 clusters, indicating better cluster cohesion and separation.

Thus, it can be concluded that the Average Linkage method with k=2 clusters is the optimal choice among the linkage methods evaluated for this dataset. This method balances both compactness and separation of clusters effectively, supporting the conclusion of our analysis.

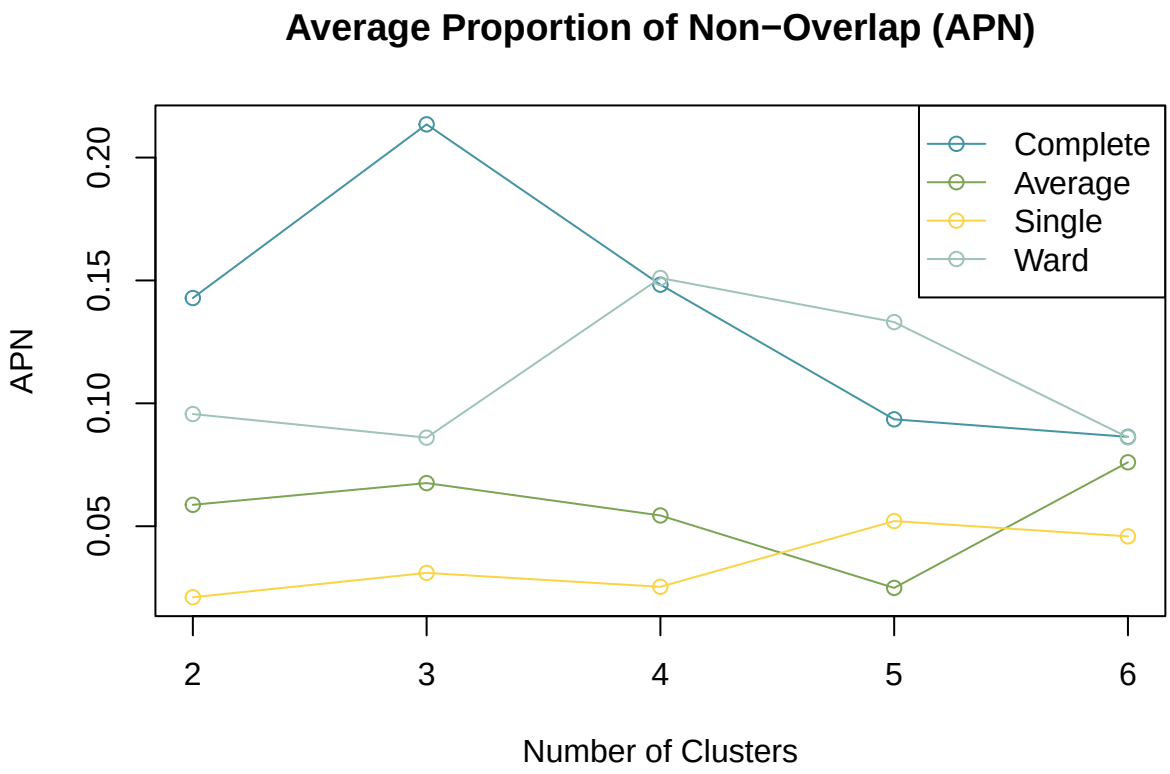
5.2 Stability Validation

```r
stab_mtcars <- clValid(mtcars, 2:6, clMethods = "hierarchical", validation = "stability")

optimal_scores_stab_mtcars <- optimalScores(stab_mtcars)
optimal_scores_stab_mtcars
```

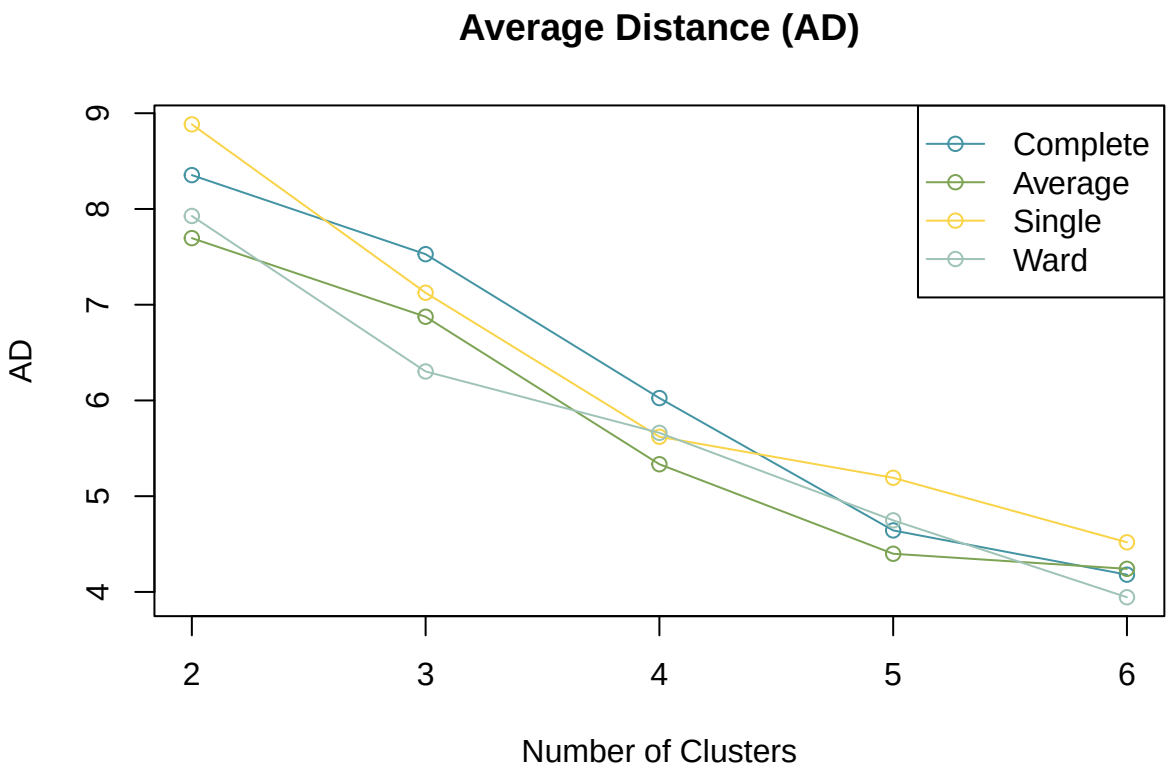
```
##          Score       Method Clusters
## APN 0.03305785 hierarchical        6
## AD  1.81739833 hierarchical        6
## ADM 0.12501894 hierarchical        6
## FOM 0.52752901 hierarchical        6
```

```r
stab_complete <- clValid(mtcars, 2:6, clMethods = "hierarchical",
                         validation = "stability", method = "complete",
                         metric = "manhattan")
stab_average <- clValid(mtcars, 2:6, clMethods = "hierarchical",
                        validation = "stability", method = "average",
                        metric = "manhattan")
stab_single <- clValid(mtcars, 2:6, clMethods = "hierarchical",
                       validation = "stability", method = "single",
                       metric = "manhattan")
stab_ward <- clValid(mtcars, 2:6, clMethods = "hierarchical",
                     validation = "stability", method = "ward",
                     metric = "manhattan")

apn_complete <- measures(stab_complete, "APN")[, , 1]
apn_average <- measures(stab_average, "APN")[, , 1]
apn_single <- measures(stab_single, "APN")[, , 1]
apn_ward <- measures(stab_ward, "APN")[, , 1]

ad_complete <- measures(stab_complete, "AD")[, , 1]
ad_average <- measures(stab_average, "AD")[, , 1]
ad_single <- measures(stab_single, "AD")[, , 1]
ad_ward <- measures(stab_ward, "AD")[, , 1]

adm_complete <- measures(stab_complete, "ADM")[, , 1]
adm_average <- measures(stab_average, "ADM")[, , 1]
adm_single <- measures(stab_single, "ADM")[, , 1]
adm_ward <- measures(stab_ward, "ADM")[, , 1]

fom_complete <- measures(stab_complete, "FOM")[, , 1]
fom_average <- measures(stab_average, "FOM")[, , 1]
fom_single <- measures(stab_single, "FOM")[, , 1]
fom_ward <- measures(stab_ward, "FOM")[, , 1]
```

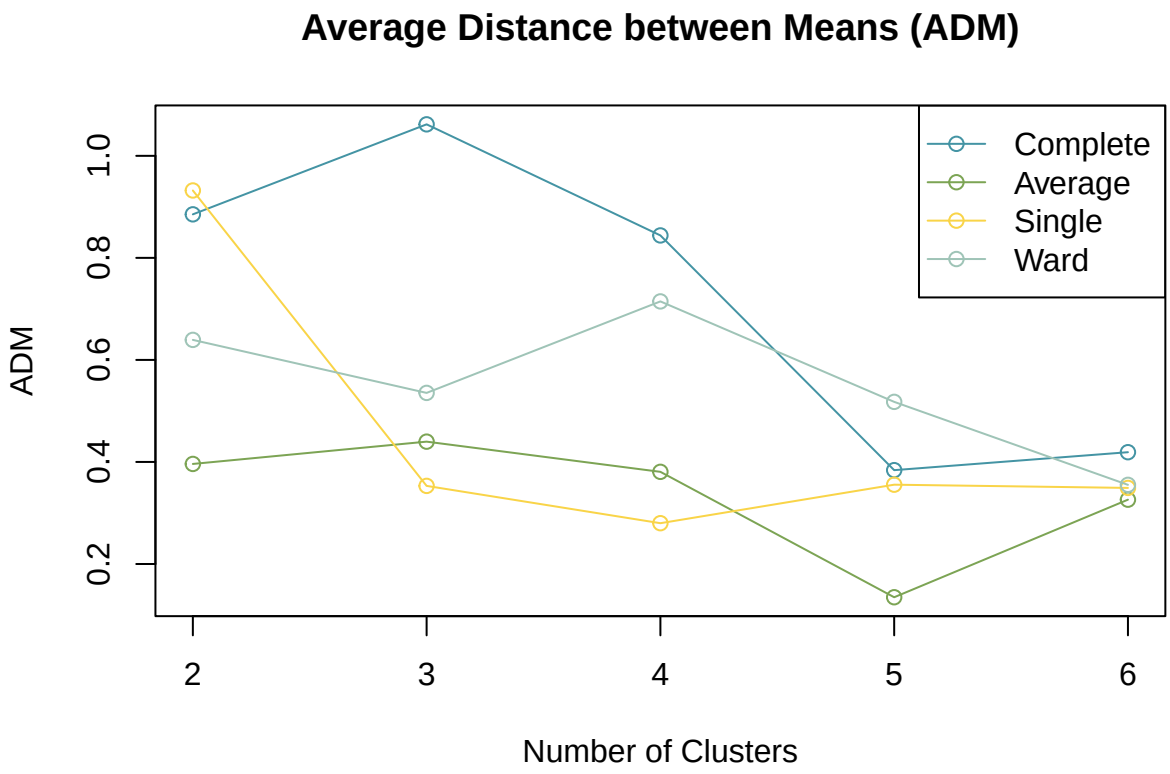
```r
# Plot APN
plot(num_clusters, apn_complete, type = "o", col = "#4494a4",
     ylim = range(apn_complete, apn_average, apn_single, apn_ward),
     xlab = "Number of Clusters", ylab = "APN",
     main = "Average Proportion of Non-Overlap (APN)")
lines(num_clusters, apn_average, type = "o", col = "#7ca454")
lines(num_clusters, apn_single, type = "o", col = "#f9d448")
lines(num_clusters, apn_ward, type = "o", col = "#9fc4b7")
legend("topright", legend = c("Complete", "Average", "Single", "Ward"),
       col = c("#4494a4", "#7ca454", "#f9d448", "#9fc4b7"), lty = 1, pch = 1)
```



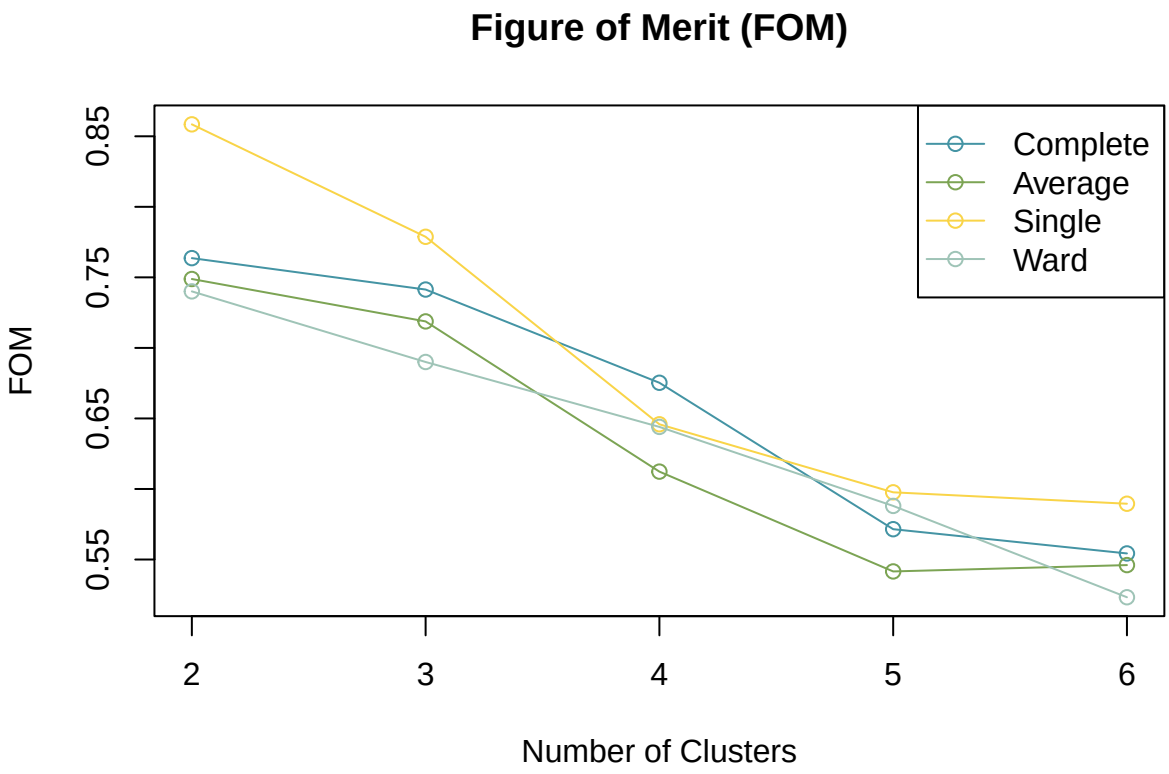

```r
# Plot AD
plot(num_clusters, ad_complete, type = "o", col = "#4494a4",
     ylim = range(ad_complete, ad_average, ad_single, ad_ward),
     xlab = "Number of Clusters", ylab = "AD", main = "Average Distance (AD)")
lines(num_clusters, ad_average, type = "o", col = "#7ca454")
lines(num_clusters, ad_single, type = "o", col = "#f9d448")
lines(num_clusters, ad_ward, type = "o", col = "#9fc4b7")
legend("topright", legend = c("Complete", "Average", "Single", "Ward"),
       col = c("#4494a4", "#7ca454", "#f9d448", "#9fc4b7"), lty = 1, pch = 1)
```




```
# Plot ADM
plot(num_clusters, adm_complete, type = "o", col = "#4494a4",
     ylim = range(adm_complete, adm_average, adm_single, adm_ward),
     xlab = "Number of Clusters", ylab = "ADM",
     main = "Average Distance between Means (ADM)")
lines(num_clusters, adm_average, type = "o", col = "#7ca454")
lines(num_clusters, adm_single, type = "o", col = "#f9d448")
lines(num_clusters, adm_ward, type = "o", col = "#9fc4b7")
legend("topright", legend = c("Complete", "Average", "Single", "Ward"),
       col = c("#4494a4", "#7ca454", "#f9d448", "#9fc4b7"), lty = 1, pch = 1)
```




```r
# Plot FOM
plot(num_clusters, fom_complete, type = "o", col = "#4494a4",
     ylim = range(fom_complete, fom_average, fom_single, fom_ward),
     xlab = "Number of Clusters", ylab = "FOM", main = "Figure of Merit (FOM)")
lines(num_clusters, fom_average, type = "o", col = "#7ca454")
lines(num_clusters, fom_single, type = "o", col = "#f9d448")
lines(num_clusters, fom_ward, type = "o", col = "#9fc4b7")
legend("topright", legend = c("Complete", "Average", "Single", "Ward"),
       col = c("#4494a4", "#7ca454", "#f9d448", "#9fc4b7"), lty = 1, pch = 1)
```



- Average Linkage method performs well across most metrics, particularly with a low APN and the lowest ADM.
- Single Linkage method has the lowest APN, indicating minimal variation in clustering results, but it falls short on other metrics like AD and FOM.
- Ward Linkage method also performs well, particularly in AD and FOM, indicating stable clustering results.

Therefore, considering all metrics, Average Linkage with k = 5 clusters is a strong candidate for optimal clustering due to its balance of compactness and stability, supported by low APN and ADM scores. However, Single Linkage with k = 2 clusters could also be considered if APN is the primary concern.

6. Conclusion and Recommendation

6.1. Conclusion

In this study, we employed agglomerative hierarchical clustering to analyze the automobile dataset, using various linkage methods to determine the optimal clustering approach. After comparing the performance of different methods, we found that the Average Linkage method provided the most meaningful and well-separated clusters. This method effectively grouped the car models into distinct categories, highlighting underlying patterns and characteristics, thereby aiding in better understanding and targeting of specific vehicle segments within the industry.

The optimal number of clusters was determined to be two, based on various validation methods such as the Silhouette Analysis and Gap Statistics.

Cluster 1, characterized by lower horsepower, higher fuel efficiency, lower weight, smaller engine displacement, and fewer cylinders, indicating more economical and smaller cars like compact and subcompact cars.

Cluster 2, characterized by higher horsepower, lower fuel efficiency, higher weight, larger engine displacement, and more cylinders, indicating more powerful and larger cars such as sports cars, muscle cars, and luxury cars.

Our interpretation and analysis of the clustering results are detailed in Research Question #2 of this project.

This clustering approach offered valuable insights by highlighting key features that differentiate distinct categories of car models, enabling a more precise understanding and targeted marketing of specific vehicle segments within the industry.

6.2. Recommendation

1. **Applying Additional Clustering Techniques:** To validate and possibly enhance our findings, we recommend applying additional clustering techniques such as K-means clustering, DBSCAN, or Gaussian Mixture Models. Comparing the results of these methods with our hierarchical clustering results can provide deeper insights and potentially reveal new patterns or clusters.

2. **Further Research and Integration:** Further investigate the impact of additional variables and external factors on clustering results. Consider integrating customer feedback and market trends to refine and validate the clustering model for more comprehensive insights.