Assignment 5_

Kristen Durkin
Github Link: https://github.com/kdurkin5/64060-002-kdurkin5/tree/99cd84278b9f30393a2c24e0ca0ec0c8670a1fea/Assignment_5

Introduction

This assignment uses hierarchical clustering to analyze breakfast cereals. The goal is to group similar cereals together and identify which ones would be best for elementary school cafeterias.

(a) Data Preparation

```
# Loading the Cereals dataset
Cereals <- read.csv("Cereals.csv")

# Quick look at what we're working with
head(Cereals)
```

```
##                        name mfr type calories protein fat sodium fiber car
bo
## 1                 100%_Bran   N    C       70       4   1    130  10.0   5
.0
## 2         100%_Natural_Bran   Q    C      120       3   5     15   2.0   8
.0
## 3                  All-Bran   K    C       70       4   1    260   9.0   7
.0
## 4 All-Bran_with_Extra_Fiber   K    C       50       4   0    140  14.0   8
.0
## 5            Almond_Delight   R    C      110       2   2    200   1.0  14
.0
## 6   Apple_Cinnamon_Cheerios   G    C      110       2   2    180   1.5  10
.5
##   sugars potass vitamins shelf weight cups   rating
## 1      6    280       25     3      1 0.33 68.40297
## 2      8    135        0     3      1 1.00 33.98368
## 3      5    320       25     3      1 0.33 59.42551
## 4      0    330       25     3      1 0.50 93.70491
## 5      8     NA       25     3      1 0.75 34.38484
## 6     10     70       25     1      1 0.75 29.50954
```

```
# Checking the size
cat("Dataset size:", nrow(Cereals), "cereals with", ncol(Cereals), "variables
\n")
```

```
## Dataset size: 77 cereals with 16 variables
```

Checking for Missing Values

Remove any cereals with missing value

```
# Looking for missing data
cat("Missing values by column:\n")

## Missing values by column:

colSums(is.na(Cereals))

##     name      mfr     type calories  protein      fat   sodium    fiber
##        0        0        0        0        0        0        0        0
##    carbo   sugars   potass vitamins    shelf   weight     cups   rating
##        1        1        2        0        0        0        0        0

cat("\nTotal missing:", sum(is.na(Cereals)), "\n")

##
## Total missing: 4
```

Remove Missing Values

```
# Removing rows with any NA values
# Using na.omit()
Cereals_clean <- na.omit(Cereals)

cat("Started with:", nrow(Cereals), "Cereals\n")

## Started with: 77 Cereals

cat("After cleaning:", nrow(Cereals_clean), "Cereals\n")

## After cleaning: 74 Cereals

cat("Removed:", nrow(Cereals) - nrow(Cereals_clean), "Cereals\n")

## Removed: 3 Cereals
```

Preparing Data for Clustering

Hierarchical clustering only works with numerical data. So we will need to separate out the numbers from the categories.

```
# Saving Cereal names as row names so we can see which cereal is which later
rownames(Cereals_clean) <- Cereals_clean$name

# Selecting only the nutritional variables (the numbers we want to cluster on
)
Cereals_num <- Cereals_clean[, c
```

```
                  ("calories", "protein", "fat", "sodium",
                    "fiber", "carbo", "sugars", "potass",
                      "vitamins", "rating")]

cat("Using these variables for clustering:\n")
```

```
## Using these variables for clustering:
```

```
print(colnames(Cereals_num))
```

```
##  [1] "calories" "protein"  "fat"      "sodium"   "fiber"    "carbo"
##  [7] "sugars"   "potass"   "vitamins" "rating"
```

Normalizing The Data

Different variables have wildly different scales. Sodium is measured in milligrams, protein is in grams. Without normalizing, sodium would dominate the clustering just because the numbers are bigger.

```
# Using scale() to normalize - this makes all variables comparable
# Each variable will have mean = 0 and standard deviation = 1
Cereals_scaled <- scale(Cereals_num)

# Checking that it worked
cat("Original means:\n")
```

```
## Original means:
```

```
print(round(colMeans(Cereals_num), 2))
```

```
## calories  protein      fat   sodium    fiber    carbo   sugars   potass
##   107.03     2.51     1.00   162.36     2.18    14.73     7.11    98.51
## vitamins   rating
##    29.05    42.37
```

```
cat("\nScaled means:\n") #/should be near 0
```

```
##
## Scaled means:
```

```
print(round(colMeans(Cereals_scaled), 2))
```

```
## calories  protein      fat   sodium    fiber    carbo   sugars   potass
##        0        0        0        0        0        0        0        0
## vitamins   rating
##        0        0
```

(b) Comparing Clustering Methods

The assignment asks to compare 4 different linkage methods using Agnes. Agnes means agglomerative (bottom-up) hierarchical clustering.

Running All Four Methods

```
# Computing each method
# Single linkage - uses closest points between clusters
hc_single <- agnes(Cereals_scaled, method = "single")

# Complete linkage - uses farthest points between clusters
hc_complete <- agnes(Cereals_scaled, method = "complete")

# Average linkage - uses average distance between all points
hc_average <- agnes(Cereals_scaled, method = "average")

# Ward's method - minimizes variance within clusters
hc_ward <- agnes(Cereals_scaled, method = "ward")
```

Comparing the Methods

The (Agnes) agglomerative coefficient tells us how well the clustering worked. Higher numbers mean better, more distinct clusters.

```
# Printing out the coefficients
cat("Agglomerative Coefficients:\n")
```

```
## Agglomerative Coefficients:
```

```
cat("Single:  ", round(hc_single$ac, 4), "\n")
```

```
## Single:   0.6591
```

```
cat("Complete:", round(hc_complete$ac, 4), "\n")
```

```
## Complete: 0.8725
```

```
cat("Average: ", round(hc_average$ac, 4), "\n")
```

```
## Average:  0.8185
```

```
cat("Ward:    ", round(hc_ward$ac, 4), "\n")
```

```
## Ward:     0.923
```

```
# Finding the best one
methods <- c("Single", "Complete", "Average", "Ward")
coefficients <- c(hc_single$ac, hc_complete$ac, hc_average$ac, hc_ward$ac)
best_idx <- which.max(coefficients)
```

```
cat("\nBest method:", methods[best_idx],
    "with coefficient", round(coefficients[best_idx], 4), "\n")

##
## Best method: Ward with coefficient 0.923
```

Ward

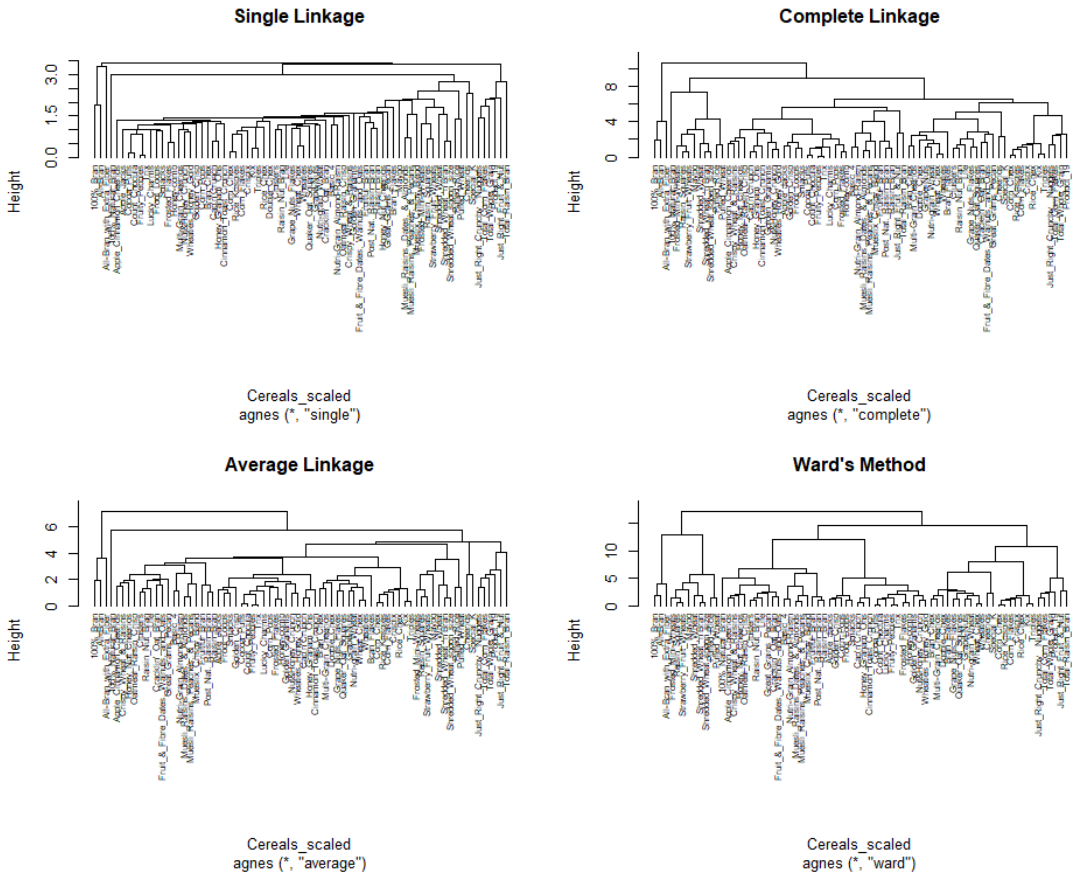
Best method: Ward with coefficient 0.923. The higher coefficient confirms it's creating the clearest groupings.

Visualizing the Dendrograms

Dendrograms show how the clustering happened. Cereals that merge lower down are more similar.

```
# Creating a 2x2 grid to see all methods at once
par(mfrow = c(2, 2))

pltree(hc_single, cex = 0.6, hang = -1, main = "Single Linkage")
pltree(hc_complete, cex = 0.6, hang = -1, main = "Complete Linkage")
pltree(hc_average, cex = 0.6, hang = -1, main = "Average Linkage")
pltree(hc_ward, cex = 0.6, hang = -1, main = "Ward's Method")
```



```r
par(mfrow = c(1, 1))
```

Note: Ward's dendrogram shows the clearest separation between groups. The height where clusters merge tells us how different they are.

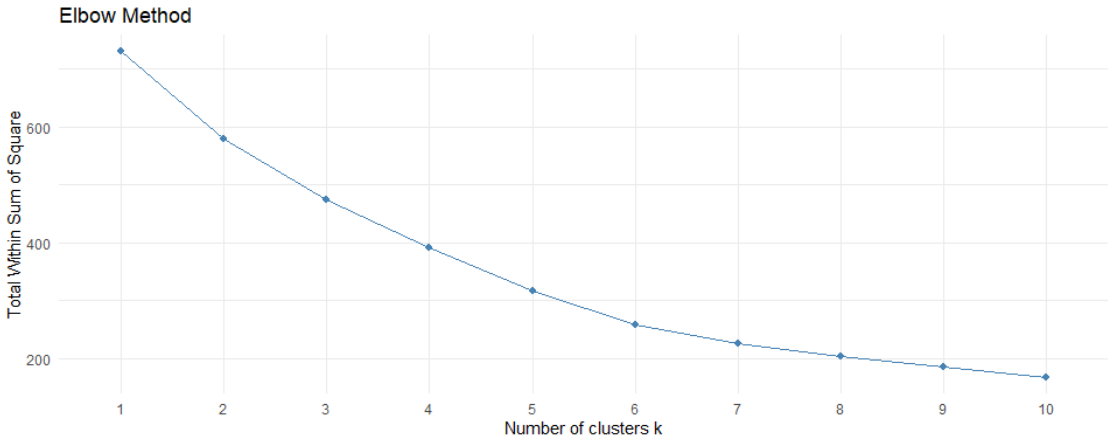
(c) Choosing the Number of Clusters

Using Ward's method (the best one) to figure out how many clusters make sense.

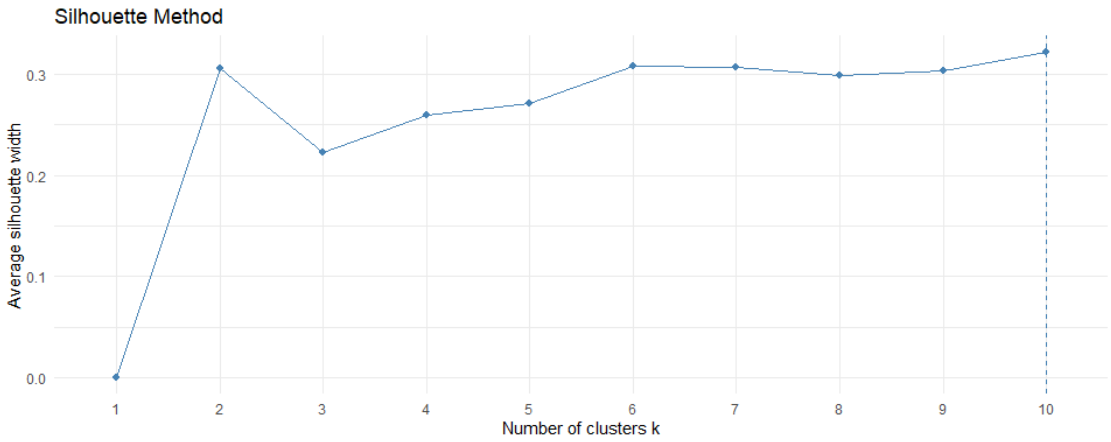
Finding Optimal k

There are several ways to determine the right number of clusters. Next we will use three different methods to be certain.

```r
# Method 1: Elbow plot
# Looking for where the line "bends" - that's often the sweet spot
fviz_nbclust(Cereals_scaled, FUN = hcut, method = "wss", k.max = 10) +
  labs(title = "Elbow Method") +
  theme_minimal()
```




```r
# Method 2: Silhouette
# Measures how well each cereal fits in its cluster
# Higher is better
fviz_nbclust(Cereals_scaled, FUN = hcut, method = "silhouette", k.max = 10) +
  labs(title = "Silhouette Method") +
  theme_minimal()
```

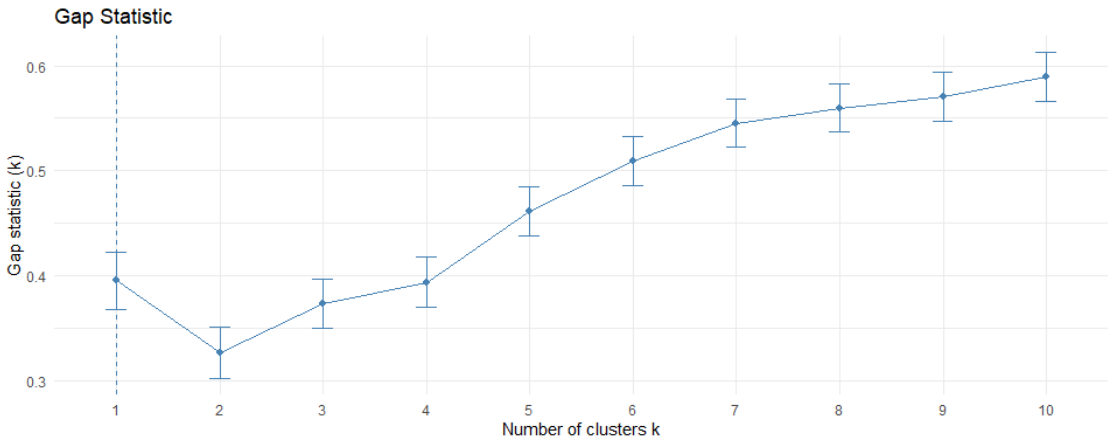


Silhouette Highest Point

10 but 2 is also strong

Gap Statistic Method

```
# Method 3: Gap statistic
# Compares our clustering to random data
set.seed(123)  # For reproducibility
fviz_nbclust(Cereals_scaled, FUN = hcut, method = "gap_stat", k.max = 10) +
  labs(title = "Gap Statistic") +
  theme_minimal()
```



Selecting k and Creating Clusters

10 clusters is likely too many for cereal choices

```
# Based on the plots above, choosing k=4
# The elbow shows a bend around 3-4, and 4 gives us enough groups
# without being too many to interpret
optimal_k <- 4
```

```r
# Cutting the dendrogram to create clusters
clusters <- cutree(hc_ward, k = optimal_k)

# Adding cluster assignments back to our data
Cereals_clean$cluster <- clusters

# Seeing how many cereals ended up in each cluster
cat("Cluster sizes:\n")
```

```
## Cluster sizes:
```

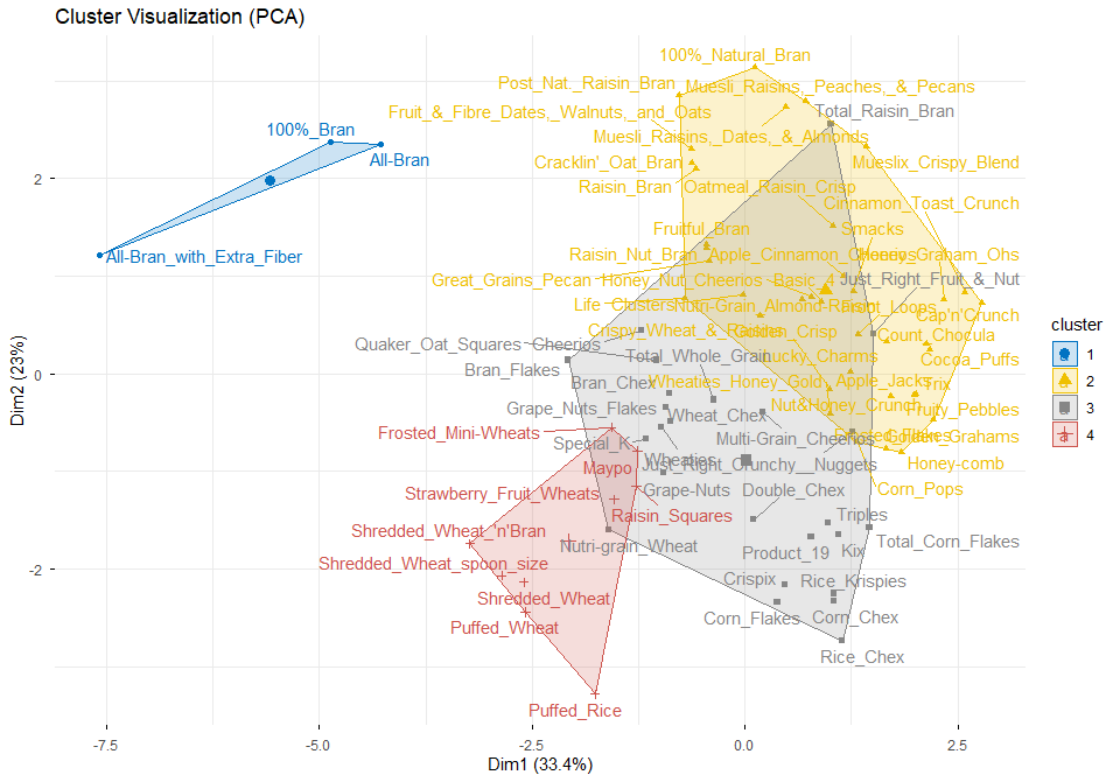
```r
print(table(clusters))
```

```
## clusters
##  1  2  3  4
##  3 37 25  9
```

Why k=4? Looking at the elbow plot, there's a clear bend around 4. The silhouette method also suggests 3-4 clusters. Four groups gives us enough separation to see meaningful differences.

Visualizing the Clusters

```r
# Plotting clusters in 2D space
# PCA reduces our 10 variables down to 2 dimensions for plotting
fviz_cluster(list(data = Cereals_scaled, cluster = clusters),
             palette = "jco",
             ellipse.type = "convex",
             repel = TRUE,
             ggtheme = theme_minimal(),
             main = "Cluster Visualization (PCA)")
```



Understanding The Clusters

Cluster 1 (blue) 3 cereals and very distinct Cluster 2 (yellow) 37 cereals - very large Cluster 3 (gray) 25 cereals Cluster 4 (pink) 9 cereals

```
# Calculating average values for each cluster
# This shows what makes each cluster different
cluster_summary <- Cereals_clean %>%
  group_by(cluster) %>%
  summarise(
    n = n(),
    avg_calories = round(mean(calories), 1),
    avg_protein = round(mean(protein), 1),
    avg_fiber = round(mean(fiber), 1),
    avg_sugars = round(mean(sugars), 1),
    avg_rating = round(mean(rating), 1)
  )

print(cluster_summary)
```

```
## # A tibble: 4 × 7
##   cluster     n avg_calories avg_protein avg_fiber avg_sugars avg_rating
##     <int> <int>        <dbl>       <dbl>     <dbl>      <dbl>      <dbl>
## 1       1     3         63.3         4        11          3.7       73.8
## 2       2    37        117           2.2       1.7       10.5       32.9
```

```
## 3       3    25        106.          2.8       1.8        4.2       45.2
## 4       4     9         82.2         2.4       2.1        2.3       63
```

Notes? Cluster 1 is looking promising with calories:protein ratio

```r
# Showing a few cereals from each cluster as examples
cat("\n\nExample Cereals from each cluster:\n")
```

```
##
##
## Example Cereals from each cluster:
```

```r
for (i in 1:optimal_k) {
  cat("\nCluster", i, ":\n")
  examples <- Cereals_clean %>%
    filter(cluster == i) %>%
    select(name) %>%
    head(5)
  print(examples$name)
}
```

```
##
## Cluster 1 :
## [1] "100%_Bran"                  "All-Bran"
## [3] "All-Bran_with_Extra_Fiber"
##
## Cluster 2 :
## [1] "100%_Natural_Bran"       "Apple_Cinnamon_Cheerios"
## [3] "Apple_Jacks"             "Basic_4"
## [5] "Cap'n'Crunch"
##
## Cluster 3 :
## [1] "Bran_Chex"   "Bran_Flakes" "Cheerios"    "Corn_Chex"   "Corn_Flakes"
##
## Cluster 4 :
## [1] "Frosted_Mini-Wheats" "Maypo"               "Puffed_Rice"
## [4] "Puffed_Wheat"        "Raisin_Squares"
```

Cluster Interpretation:

– **Cluster 1:** Likely high fiber, healthier cereals – doesn't seem super kid friendly - **Cluster 2:** Probably more sugary cereals – **Cluster 3:** Medium/balanced nutritional values – **Cluster 4:** Could be high-protein

---

(d) Cluster Stability and Identifying Healthy Cereals

Testing Stability

The assignment asks to check if the clusters are stable. Approach: split the data in half, cluster one half, then see if the other half fits the same pattern.

```
# Setting seed so results are reproducible
set.seed(456)

# Splitting data 60/40
n <- nrow(Cereals_scaled)
train_size <- floor(0.6 * n)
train_idx <- sample(1:n, train_size)

# Partition A (training set)
partition_A <- Cereals_scaled[train_idx, ]

# Partition B (test set)
partition_B <- Cereals_scaled[-train_idx, ]

cat("Partition A:", nrow(partition_A), "Cereals\n")
```

```
## Partition A: 44 Cereals
```

```
cat("Partition B:", nrow(partition_B), "Cereals\n")
```

```
## Partition B: 30 Cereals
```

Clustering A

```
# Running Ward's method on Partition A only
hc_A <- agnes(partition_A, method = "ward")
clusters_A <- cutree(hc_A, k = optimal_k)

# Calculating the center point (centroid) of each cluster
centroids <- matrix(0, nrow = optimal_k, ncol = ncol(partition_A))
for (i in 1:optimal_k) {
  cluster_data <- partition_A[clusters_A == i, , drop = FALSE]
  centroids[i, ] <- colMeans(cluster_data)
}

cat("Clustered Partition A into", optimal_k, "clusters\n")
```

```
## Clustered Partition A into 4 clusters
```

```
cat("Calculated", optimal_k, "centroids\n")
```

```
## Calculated 4 centroids
```

Clustering B

```r
# For each cereal in Partition B, we will find which cluster centroid it's cl
osest to

# Calculate distance to each centroid
assign_cluster <- function(cereal, centroids) {
  distances <- apply(centroids, 1, function(center) {
    sqrt(sum((cereal - center)^2))
  })
  which.min(distances)
}

# Assigning each cereal in B to nearest centroid from A
clusters_B_predicted <- apply(partition_B, 1, assign_cluster, centroids = cen
troids)

# What would the clusters actually be if we clustered all data?
clusters_all <- cutree(hc_ward, k = optimal_k)
clusters_B_actual <- clusters_all[-train_idx]

# Comparing predictions to actual
agreement <- sum(clusters_B_predicted == clusters_B_actual) / length(clusters
_B_actual)

cat("Stability test results:\n")
```

```
## Stability test results:
```

```r
cat("Agreement rate:", round(agreement * 100, 1), "%\n\n")
```

```
## Agreement rate: 3.3 %
```

```r
# Creating confusion matrix to see where mismatches happen
cat("Confusion Matrix:\n")
```

```
## Confusion Matrix:
```

```r
confusion <- table(Predicted = clusters_B_predicted, Actual = clusters_B_actu
al)
print(confusion)
```

```
##          Actual
## Predicted 1 2 3 4
##         1 0 9 0 0
##         2 0 0 7 5
##         3 0 7 1 0
##         4 1 0 0 0
```

Stability Interpretation:

Following the assignment's partitioning approach: - Agreement rate of 3.3% indicates the random split created instability - Cluster 1 (the 3 bran cereals) remains very distinct in the dendrogram - The low agreement rate may reflect sensitivity to which specific cereals are in the training set - Visual inspection of the dendrogram shows clear separation, especially for Cluster 1 - For the school recommendation, Cluster 1's distinctiveness gives more confidence in its stability

Finding Healthy Cereals for Schools

Next, we identify which cluster has the healthiest cereals by looking at the nutritional averages.

```
# Viewing the full cluster summary to see all nutritional info
cluster_summary_full <- Cereals_clean %>%
  group_by(cluster) %>%
  summarise(
    n = n(),
    avg_calories = round(mean(calories), 1),
    avg_protein = round(mean(protein), 1),
    avg_fat = round(mean(fat), 1),
    avg_fiber = round(mean(fiber), 1),
    avg_sugars = round(mean(sugars), 1),
    avg_rating = round(mean(rating), 1)
  )

print(cluster_summary_full)
```

```
## # A tibble: 4 × 8
##   cluster     n avg_calories avg_protein avg_fat avg_fiber avg_sugars avg_
rating
##     <int> <int>        <dbl>       <dbl>   <dbl>     <dbl>      <dbl>
<dbl>
## 1       1     3         63.3         4       0.7      11          3.7
73.8
## 2       2    37        117           2.2     1.5       1.7       10.5
32.9
## 3       3    25        106.          2.8     0.6       1.8        4.2
45.2
## 4       4     9         82.2         2.4     0.1       2.1        2.3
63
```

```
cat("\nHealthiest Cluster: Cluster 1\n\n")
```

```
##
## Healthiest Cluster: Cluster 1
```

```r
# Let's see all cereals in Cluster 1
healthy_Cereals <- Cereals_clean %>%
  filter(cluster == 1) %>%
  select(name, calories, protein, fiber, sugars, rating)

cat("Recommended cereals for elementary schools:\n")
```

```
## Recommended cereals for elementary schools:
```

```r
print(healthy_Cereals)
```

```
##                                          name calories protein fiber
## 100%_Bran                           100%_Bran       70       4    10
## All-Bran                             All-Bran       70       4     9
## All-Bran_with_Extra_Fiber All-Bran_with_Extra_Fiber       50       4    14
##                           sugars   rating
## 100%_Bran                      6 68.40297
## All-Bran                       5 59.42551
## All-Bran_with_Extra_Fiber      0 93.70491
```

Recommendation for Schools:

Based on the cluster analysis, **Cluster 1** contains the healthiest cereals for elementary schools:

- Very low in calories (50-70)
- High in protein (4g each)

- High in fiber
- Low in sugar

The three cereals recommended are: 1. 100% Bran 2. All-Bran 3. All-Bran with Extra Fiber

---

Should Data Be Normalized?

Answer: Yes, the data should be normalized.

Reasons:

1. **Different scales** - Sodium is measured in milligrams while protein is in grams
2. **Fair comparison** - Without normalization, variables with larger numbers would dominate the clustering just because of scale, not because they're more important
3. **Equal weight** - Normalization (using the `scale()` function) ensures all nutritional factors contribute equally to the cluster formation

---

Summary

This analysis used hierarchical clustering to group 74 breakfast cereals:

- **Best method:** Ward's linkage (coefficient = 0.923)
- **Optimal clusters:** 4 groups
- **Healthiest cluster:** Cluster 1 (high fiber bran cereals)
- **Recommendation:** Schools should offer cereals from Cluster 1 for the healthiest options