Assignment5

2025-06-22

library(tidyverse)

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.4 ✔ readr 2.1.5  
## ✔ forcats 1.0.0 ✔ stringr 1.5.1  
## ✔ ggplot2 3.5.2 ✔ tibble 3.2.1  
## ✔ lubridate 1.9.4 ✔ tidyr 1.3.1  
## ✔ purrr 1.0.4   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(cluster)  
library(factoextra)

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

# Step 1: Load and Clean Data  
  
cereal\_data <- read.csv("Cereals.csv")  
cereal\_clean <- na.omit(cereal\_data)  
  
  
# Step 2: Normalize Numeric Data  
  
cereal\_num <- cereal\_clean %>%  
 select(where(is.numeric))  
  
cereal\_scaled <- scale(cereal\_num)  
  
  
# Step 3: Apply AGNES Clustering with Ward Linkage   
  
agnes\_ward <- agnes(cereal\_scaled, method = "ward")  
  
# Cut tree into k clusters   
k <- 4  
clusters\_all <- cutree(agnes\_ward, k = k)  
  
  
# Step 4: Partition the Data into Two Halves  
  
set.seed(123)  
n <- nrow(cereal\_scaled)  
idx <- sample(1:n, n / 2)  
  
partA <- cereal\_scaled[idx, ]  
partB <- cereal\_scaled[-idx, ]  
  
  
# Step 5: Cluster Partition A and Get Centroids  
  
agnes\_partA <- agnes(partA, method = "ward")  
clusters\_A <- cutree(agnes\_partA, k = k)  
  
# Get centroids for each cluster in partA  
centroids\_A <- aggregate(partA, by = list(cluster = clusters\_A), FUN = mean)  
centroid\_matrix <- as.matrix(centroids\_A[, -1])   
  
  
# Step 6: Assign Partition B to Nearest Centroid from A  
  
assign\_to\_centroid <- function(row, centroids) {  
 distances <- apply(centroids, 1, function(c) sqrt(sum((row - c)^2)))  
 return(which.min(distances))  
}  
  
# Assign cluster to each row in partB  
cluster\_B\_assigned <- apply(partB, 1, assign\_to\_centroid, centroids = centroid\_matrix)  
  
# Also cluster partB directly for comparison  
agnes\_partB <- agnes(partB, method = "ward")  
clusters\_B\_true <- cutree(agnes\_partB, k = k)  
  
  
# Step 7: Evaluate Stability  
  
# Calculate % agreement between assigned and true clusters   
agreement <- sum(cluster\_B\_assigned == clusters\_B\_true) / length(clusters\_B\_true)  
print(paste("Agreement between assigned and true B clusters:", round(agreement \* 100, 2), "%"))

## [1] "Agreement between assigned and true B clusters: 27.03 %"

# Step 8: Identify Healthy Cereals  
  
# Healthy cereals could be identified from clusters\_all  
  
cereal\_with\_cluster <- cereal\_clean %>%  
 mutate(cluster = clusters\_all)  
  
# Get summary stats by cluster to identify healthy ones  
healthy\_summary <- cereal\_with\_cluster %>%  
 group\_by(cluster) %>%  
 summarise(across(where(is.numeric), mean, na.rm = TRUE))

## Warning: There was 1 warning in `summarise()`.  
## ℹ In argument: `across(where(is.numeric), mean, na.rm = TRUE)`.  
## ℹ In group 1: `cluster = 1`.  
## Caused by warning:  
## ! The `...` argument of `across()` is deprecated as of dplyr 1.1.0.  
## Supply arguments directly to `.fns` through an anonymous function instead.  
##   
## # Previously  
## across(a:b, mean, na.rm = TRUE)  
##   
## # Now  
## across(a:b, \(x) mean(x, na.rm = TRUE))

print(healthy\_summary)

## # A tibble: 4 × 14  
## cluster calories protein fat sodium fiber carbo sugars potass vitamins  
## <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 1 63.3 4 0.667 177. 11 6.67 3.67 310 25   
## 2 2 124 3.15 1.95 155 3.1 14.0 9.35 152. 31.2  
## 3 3 111. 1.52 1 172. 0.571 12.6 11.3 46.0 25   
## 4 4 97.3 2.63 0.4 159. 1.8 17.5 3.03 78.8 30.8  
## # ℹ 4 more variables: shelf <dbl>, weight <dbl>, cups <dbl>, rating <dbl>