R Zone 6: Chapter 21 and 22

Brandon Hufstetler, Garrett Alarcon, Jinan Andrews, Anson Cheng, Nick Forrest, and Nestor Herandez

19 August 2019

Chapter 21

OPEN REQUIRED PACKAGE, READ IN THE DATA

Commentary: Note that birch is no longer supported on CRAN and must be installed through an archived version. The approval and interest were removed from the classification training data because the interest is perfectly correlated with the requested amount. Also, the original R-Zone code says to use 5000 records but is scripted to only load in 1000 values. The correction was made below.

```
#install.packages("birch")
setwd("~/IMGT680")
library(birch)
loan.test <- read.csv(file="Loans_Test.csv", header = TRUE)
loan.train <- read.csv(file="Loans_Training.csv", header = TRUE)

# Use 5,000 records for a small example
train <- as.matrix(loan.train[1:5000,-c(1,5)])
```

BIRCH CLUSTERING

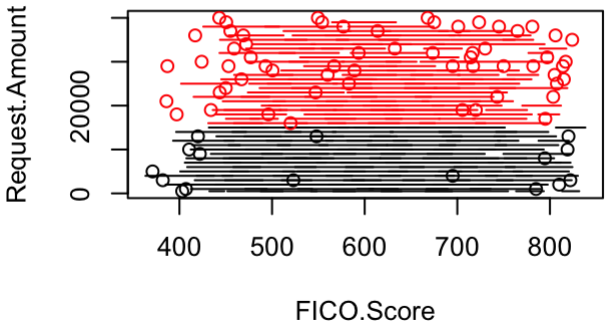
Commentary: The following snippet first creates the CF tree (radius 1000) using the birch() command, then uses kmeans to classify the birch clusters, with 2 centroids.

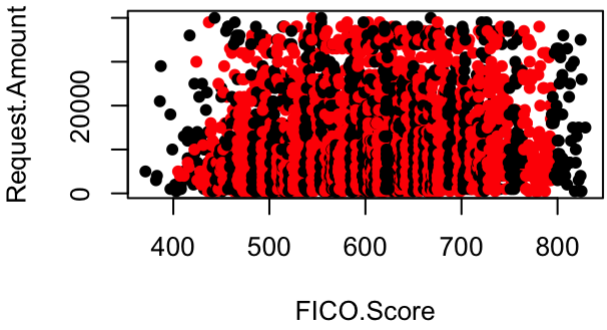
```
b1 <- birch(x = train, radius=1000) # Create the Birch tree

# Cluster the sub-clusters using kmeans
kb1 <- kmeans.birch(birchObject = b1, centers = 2, nstart = 1)
```

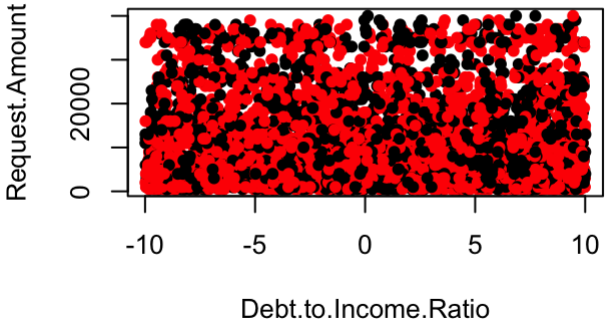
PLOT THE RESULTS

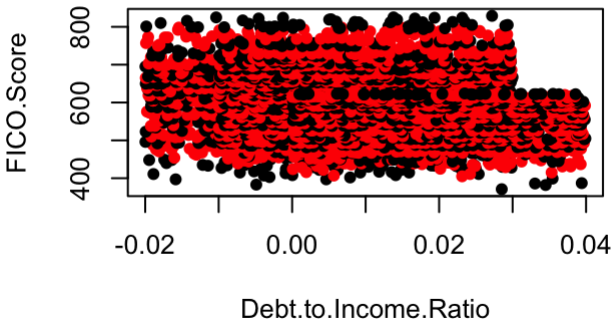
Commentary: Since only one value of k is selected (2), we cannot make a comparison to different models and BIRCH is unable to suggest a k. The CF tree clearly splits the data into two distinct categories when viewing the loan amount vs FICO score.

```
par(mfrow=c(2,2))
plot(b1[,c(2,3)], col = kb1$clust$sub)
plot(train[,c(2,3)], col = kb1$clust$sub, pch = 16)
plot(jitter(train[,c(1,3)], .1), col = kb1$clust$sub, pch = 16)
plot(jitter(train[,c(1,2)], .1), col = kb1$clust$sub, pch = 16)
```









Chapter 22

READ IN AND PREPARE THE DATA

Commentary: Iris data used here. Min-max normalization applied to normalize all variables (from chapter 2)

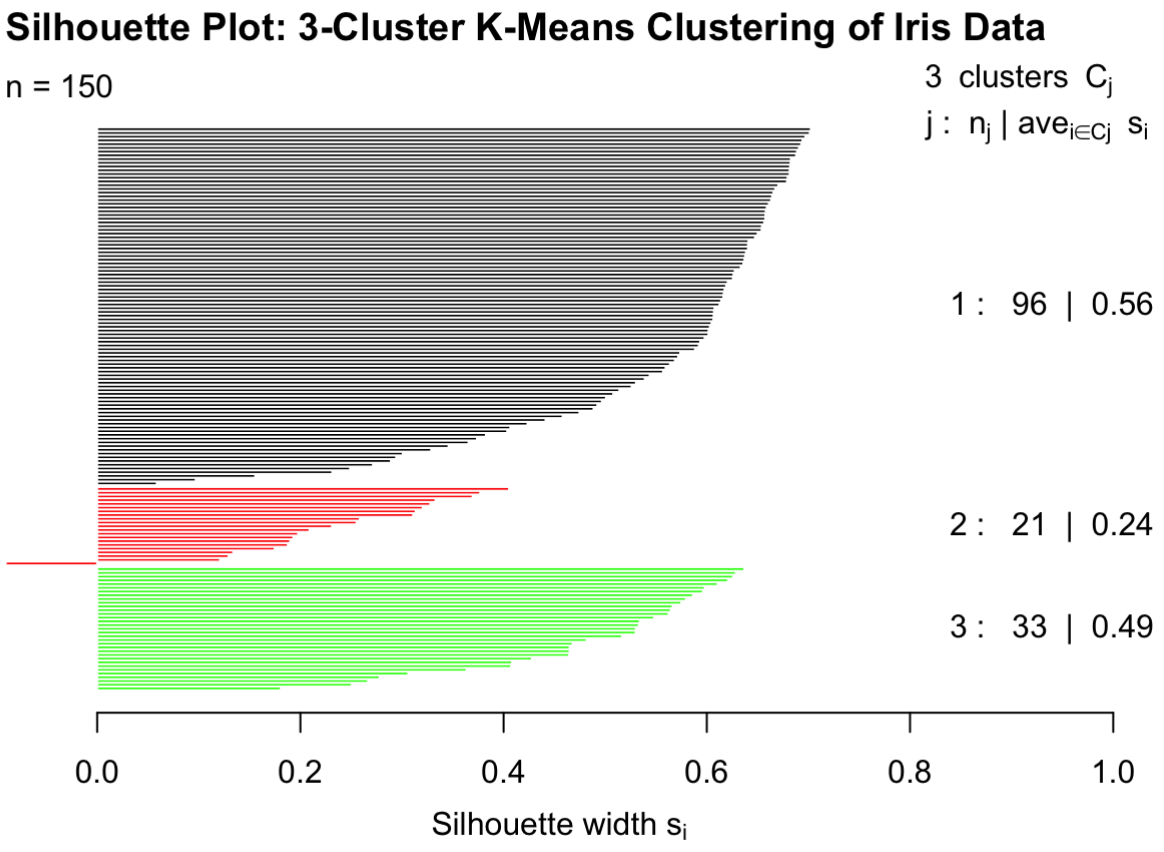
```
i.data <- iris # Iris is a built-in dataset
# Min-max normalization
i.data$SL <- (i.data$Sepal.Length - min(i.data$Sepal.Length))/
  (max(i.data$Sepal.Length) - min(i.data$Sepal.Length))
i.data$SW <- (i.data$Sepal.Width - min(i.data$Sepal.Width))/
  (max(i.data$Sepal.Width) - min(i.data$Sepal.Width))
i.data$PL <- (i.data$Petal.Length - min(i.data$Petal.Length))/
  (max(i.data$Petal.Length) - min(i.data$Petal.Length))
i.data$PW <- (i.data$Petal.Width - min(i.data$Petal.Width))/
  (max(i.data$Petal.Width) - min(i.data$Petal.Width))
```

SILHOUETTE VALUES

Commentary: Silhouette values provide a combined measure of cohesion and separation. A positive value indicates that the assignment is good, with higher values being better than lower values. A value that is close to zero is considered to be a weak assignment, as the observation could have been assigned to the next closest

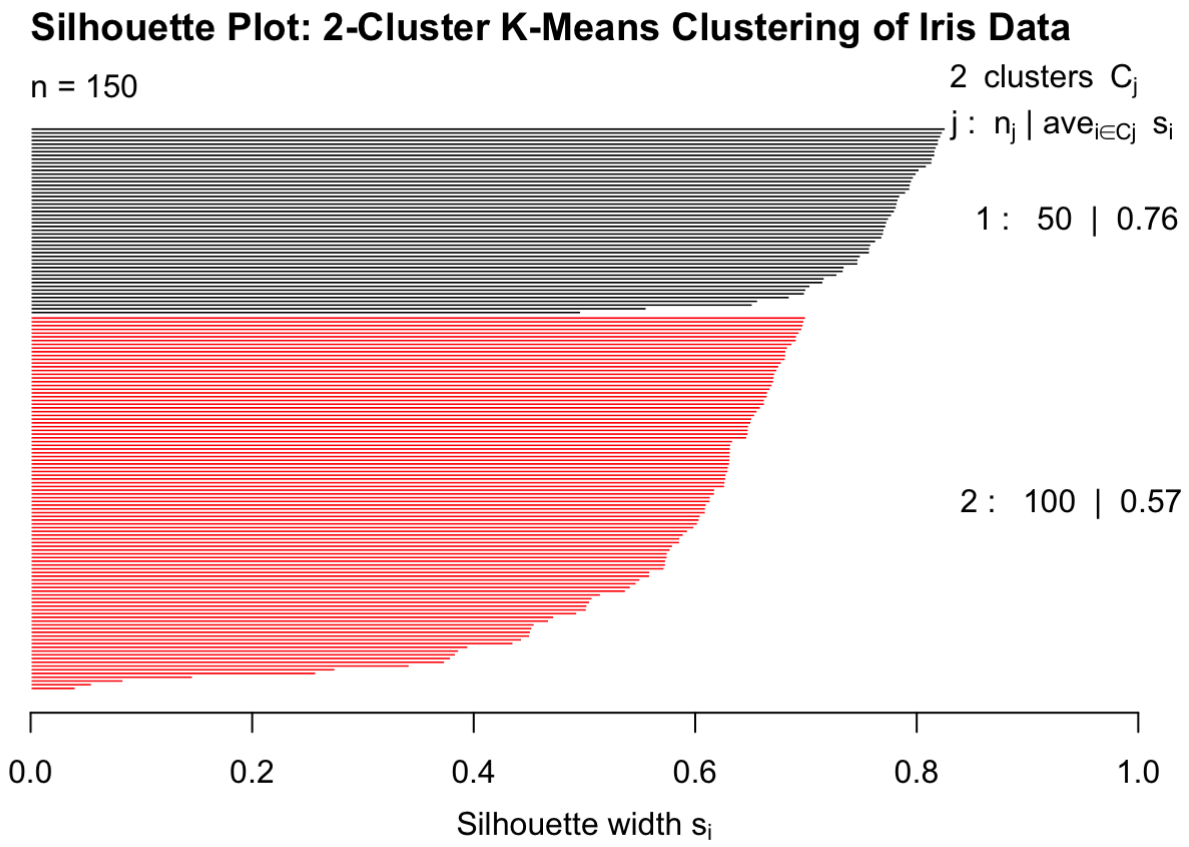
cluster with limited negative consequence. A negative silhouette value is considered to be misclassified, as assignment to the next closest cluster would have been better. A high average silhouette value (>0.5) provides good evidence of the reality of the clusters in the data. A medium average value (0.25-0.5) provides some evidence of the same and hopefully domain-specific knowledge can be brought to bear to support the reality of the clusters. A low average silhouette value (<0.25) provides scant evidence of cluster reality. In the following plots, we see that the average silhouette value when k=3 is 0.5 (not compelling) and 2 of the clusters have an average value of 0.4 (yikes!). The average silhouette value when k=2 is 0.63 (much better) and both clusters have an average value of >0.5.

```
# Requires package 'cluster'
#install.packages("cluster")
library(cluster)

# k-means (k=3)
km1 <- kmeans(i.data[,6:9], 3)
dist1 <- dist(i.data[,6:9], method = "euclidean")
sil1 <- silhouette(km1$cluster, dist1)
plot(sil1, col = c("black", "red", "green"),
     main = "Silhouette Plot: 3-Cluster K-Means Clustering of Iris Data")
```



Average silhouette width : 0.5

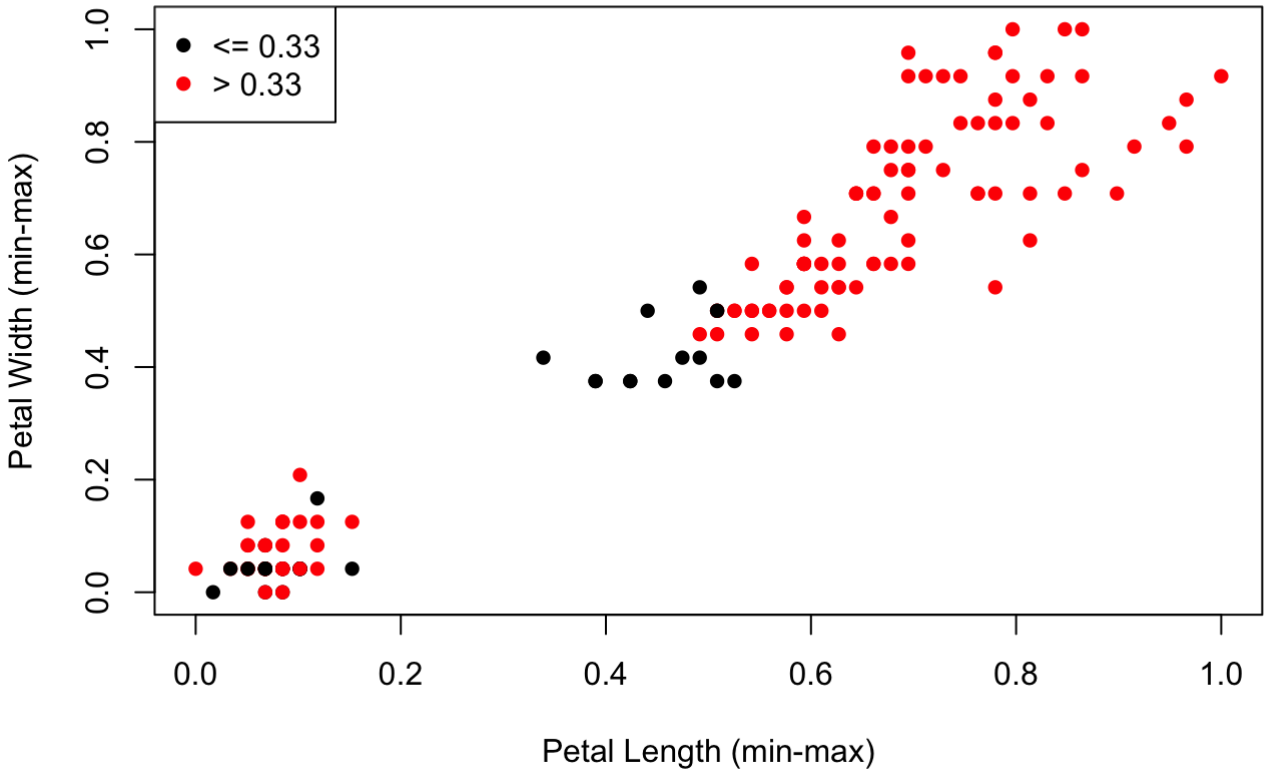
```
# k-means (k=2)
km2 <- kmeans(i.data[,6:9], 2)
dist2 <- dist(i.data[,6:9], method = "euclidean")
sil2 <- silhouette(km2$cluster, dist2)
plot(sil2, col = c("black", "red"),
     main = "Silhouette Plot: 2-Cluster K-Means Clustering of Iris Data")
```



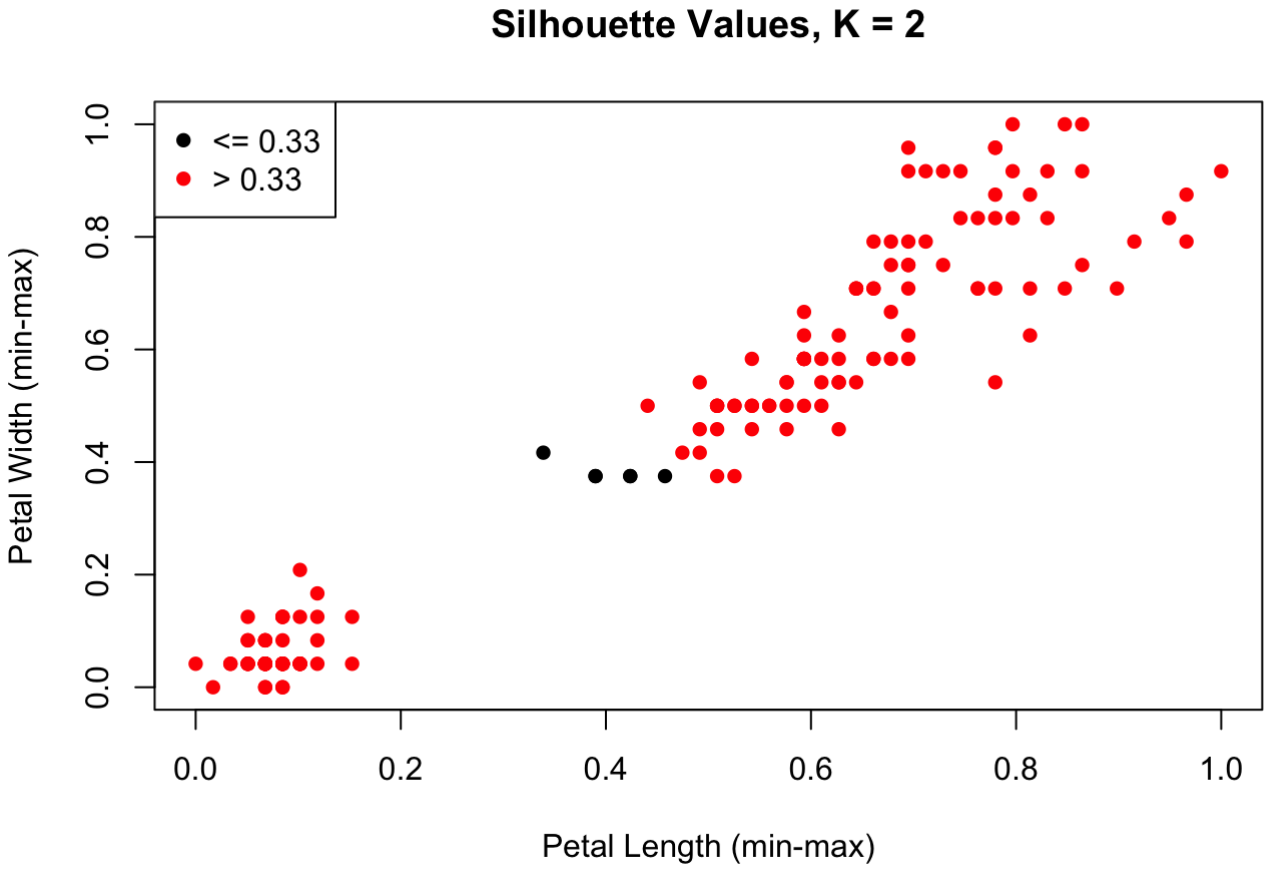
PLOT SILHOUETTE VALUES

Commentary: The plots here show which observations have an extremely low (<0.33) silhouette value, shown in black. When k=3 there are 32 such observations but when k=2 there are only 6.

```
silval1 <- ifelse(sil1[,3] <= 0.33, 0, 1)
plot(i.data$PL, i.data$PW, col = silval1+1,
     pch = 16,
     main = "Silhouette Values, K = 3",
     xlab = "Petal Length (min-max)",
     ylab = "Petal Width (min-max)")
legend("topleft", col=c(1,2), pch = 16, legend=c("<= 0.33", "> 0.33"))
```




```
silval2 <- ifelse(sil2[,3] <= 0.33, 0, 1)
plot(i.data$PL, i.data$PW, col = silval2+1,
      pch = 16,
      main = "Silhouette Values, K = 2",
      xlab = "Petal Length (min-max)",
      ylab = "Petal Width (min-max)")
legend("topleft", col=c(1,2), pch = 16,
        legend=c("<= 0.33", "> 0.33"))
```



PSEUDO-F

Commentary: The pseudo-F for k=3 is 359.85 and the pseudo-F for k=2 is 354.37. Since the p-value for k=3 goes out to the 57th decimal place while the p-value for k=2 only goes out to the 41st decimal place, the pseudo-F statistic prefers k=3. But seriously, are they really so different at that point?

```
# Requires package 'clusterSim'
#install.packages("clusterSim")
library("clusterSim")
n <- dim(i.data)[1]
psF1 <- index.G1(i.data[,6:9], cl = km1$cluster)
pf(psF1, 2, n-2)
```

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

```
psF2 <- index.G1(i.data[,6:9], cl = km2$cluster)
pf(psF2, 1, n-1)
```

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

CLUSTER VALIDATION—PREPARE THE DATA

Commentary: Here we read in the loan data and separate out the response variable (APPROVAL) in column 1. Next we take the k-means clusters for k=3 for both the testing and training data.

```
  setwd("~/IMGT680")
  loan.test <- read.csv(file="Loans_Test.csv", header = TRUE)
  loan.train <- read.csv(file="Loans_Training.csv", header = TRUE)
  test <- loan.test[,-1]
  train <- loan.train[,-1]
  kmtest <- kmeans(test, centers = 3)
  kmtrain <- kmeans(train, centers = 3)
```

CLUSTER VALIDATION—VARIABLE SUMMARIES BY CLUSTER

Commentary: Knowing the means and standard deviations allows us to perform a t-test to determine if the clusters are significantly independent. While these may not work well for large sample sizes, it is always recommended to report the difference in variable means between partitions so that the client (or someone famailiar with the data) can render judgement on whether the difference is of practical significance. The following table only lays out the mean and standard deviation for the testing and training sets' 3 clusters for the Debt-to-Income Ratio. We can see that the means are all 2-3 DIR apart.

```
  clust.sum <- matrix(0.0, ncol = 3, nrow = 4)
  colnames(clust.sum) <- c("Cluster 1", "Cluster 2", "Cluster 3")
  rownames(clust.sum) <- c("Test Data Mean", "Train Data Mean", "Test Data Std Dev", "Tr
ain Data Std Dev")
  clust.sum[1,] <- tapply(test$Debt.to.Income.Ratio, kmtest$cluster, mean)
  clust.sum[2,] <- tapply(train$Debt.to.Income.Ratio, kmtrain$cluster, mean)
  clust.sum[3,] <- tapply(test$Debt.to.Income.Ratio, kmtest$cluster, sd)
  clust.sum[4,] <- tapply(train$Debt.to.Income.Ratio, kmtrain$cluster, sd)
  clust.sum
```

```
##                    Cluster 1 Cluster 2 Cluster 3
## Test Data Mean     0.2154606 0.1876278 0.1691428
## Train Data Mean    0.2187358 0.1705706 0.1903536
## Test Data Std Dev  0.1546664 0.1310462 0.1335943
## Train Data Std Dev 0.1549292 0.1334694 0.1309812
```