Lab 8

Adeline Casali

2024-03-07

K-Means Clustering

# Load packages  
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)

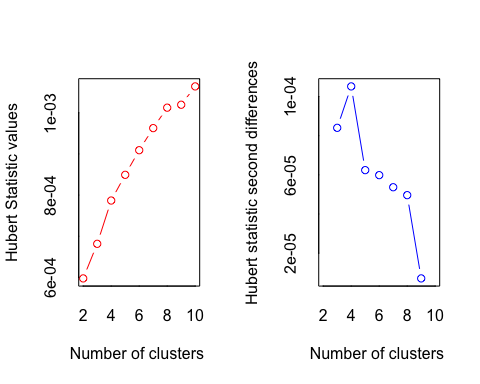
## Loading required package: ggplot2

## Loading required package: lattice

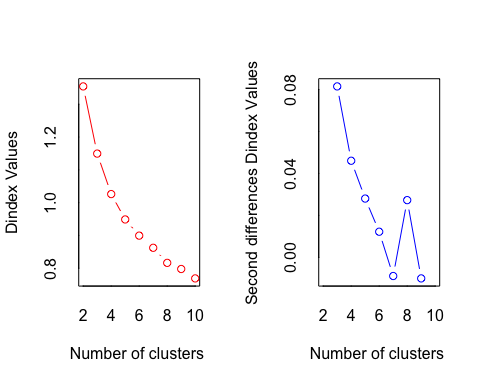
library(NbClust)  
  
# Data preprocessing  
data <- read.csv("lab\_8\_data.csv")  
training\_ind <- createDataPartition(data$lodgepole\_pine,  
 p = 0.75,  
 list = FALSE,  
 times = 1)  
training\_set <- data[training\_ind, ]  
test\_set <- data[-training\_ind, ]  
top\_20\_soil\_types <- training\_set %>%  
 group\_by(soil\_type) %>%  
 summarise(count = n()) %>%  
 arrange(desc(count)) %>%  
 select(soil\_type) %>%  
 top\_n(20)

## Selecting by soil\_type

training\_set$soil\_type <- ifelse(training\_set$soil\_type %in% top\_20\_soil\_types$soil\_type,  
 training\_set$soil\_type,  
 "other")  
training\_set$wilderness\_area <- factor(training\_set$wilderness\_area)  
training\_set$soil\_type <- factor(training\_set$soil\_type)  
onehot\_encoder <- dummyVars(~ wilderness\_area + soil\_type,  
 training\_set[, c("wilderness\_area", "soil\_type")],  
 levelsOnly = TRUE,  
 fullRank = TRUE)  
onehot\_enc\_training <- predict(onehot\_encoder,  
 training\_set[, c("wilderness\_area", "soil\_type")])  
training\_set <- cbind(training\_set, onehot\_enc\_training)  
test\_set$soil\_type <- ifelse(test\_set$soil\_type %in% top\_20\_soil\_types$soil\_type,  
 test\_set$soil\_type,  
 "other")  
test\_set$wilderness\_area <- factor(test\_set$wilderness\_area)  
test\_set$soil\_type <- factor(test\_set$soil\_type)  
onehot\_enc\_test <- predict(onehot\_encoder, test\_set[, c("wilderness\_area", "soil\_type")])  
test\_set <- cbind(test\_set, onehot\_enc\_test)  
test\_set[, -c(11:13)] <- scale(test\_set[, -c(11:13)],  
 center = apply(training\_set[, -c(11:13)], 2, mean),  
 scale = apply(training\_set[, -c(11:13)], 2, sd))  
training\_set[, -c(11:13)] <- scale(training\_set[, -c(11:13)])  
training\_features <- array(data = unlist(training\_set[, -c(11:13)]),  
 dim = c(nrow(training\_set), 33))  
training\_labels <- array(data = unlist(training\_set[, 13]),  
 dim = c(nrow(training\_set)))  
test\_features <- array(data = unlist(test\_set[, -c(11:13)]),  
 dim = c(nrow(test\_set), 33))  
test\_labels <- array(data = unlist(test\_set[, 13]),  
 dim = c(nrow(test\_set)))  
  
# Performing k-means clustering  
set.seed(123)  
nc <- NbClust(training\_features[sample(nrow(training\_features), 1000), c(4, 6, 10)],   
 min.nc = 2, max.nc = 10, method = "kmeans")



## \*\*\* : The Hubert index is a graphical method of determining the number of clusters.  
## In the plot of Hubert index, we seek a significant knee that corresponds to a   
## significant increase of the value of the measure i.e the significant peak in Hubert  
## index second differences plot.   
##



## \*\*\* : The D index is a graphical method of determining the number of clusters.   
## In the plot of D index, we seek a significant knee (the significant peak in Dindex  
## second differences plot) that corresponds to a significant increase of the value of  
## the measure.   
##   
## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
## \* Among all indices:   
## \* 4 proposed 2 as the best number of clusters   
## \* 5 proposed 3 as the best number of clusters   
## \* 11 proposed 4 as the best number of clusters   
## \* 1 proposed 5 as the best number of clusters   
## \* 1 proposed 8 as the best number of clusters   
## \* 1 proposed 9 as the best number of clusters   
##   
## \*\*\*\*\* Conclusion \*\*\*\*\*   
##   
## \* According to the majority rule, the best number of clusters is 4   
##   
##   
## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

km\_clusters <- kmeans(training\_features[, c(4, 6, 10)], centers = 4)  
cluster\_number <- data.frame(cluster\_number = km\_clusters$cluster)  
training\_features <- cbind(training\_features, cluster\_number)  
head(training\_features)

## 1 2 3 4 5 6 7  
## 1 -1.6195349 -0.5612973 -0.9639199 -0.4181820 -0.5000952 -1.0493514 0.6949758  
## 2 -0.2936284 -0.6154187 -0.2769370 -1.1342230 -0.7546865 1.2998738 0.9263003  
## 3 -0.8218351 -0.1914682 0.2726493 -0.2014854 0.5352428 -0.4902268 1.1961787  
## 4 -0.3726798 -0.2636300 -0.8265234 -0.0130536 -0.6867955 0.6035447 0.7720840  
## 5 -0.5056297 -0.7417018 0.8222356 -0.8515753 -0.4661497 0.3465277 0.9648543  
## 6 -0.7823094 -0.2906907 0.6848390 -1.1342230 -0.7886320 0.2518372 1.3503950  
## 8 9 10 11 12 13  
## 1 0.28897289 -0.2817033 2.6652191 -0.2309271 -0.8917759 -0.2549411  
## 2 -0.12643507 -0.7527207 2.9774721 -0.2309271 -0.8917759 -0.2549411  
## 3 0.39282488 -0.8050560 -0.8332273 -0.2309271 -0.8917759 -0.2549411  
## 4 0.60052886 -0.2032004 0.8007958 -0.2309271 -0.8917759 -0.2549411  
## 5 -1.32073296 -1.5639173 0.2043319 -0.2309271 -0.8917759 -0.2549411  
## 6 -0.07450908 -1.2499057 0.1929635 -0.2309271 -0.8917759 -0.2549411  
## 14 15 16 17 18 19  
## 1 -0.04463561 -0.03712769 -0.5020164 -0.09294794 -0.2327585 -0.2130992  
## 2 -0.04463561 -0.03712769 -0.5020164 -0.09294794 4.2956413 -0.2130992  
## 3 -0.04463561 -0.03712769 -0.5020164 -0.09294794 4.2956413 -0.2130992  
## 4 -0.04463561 -0.03712769 -0.5020164 -0.09294794 -0.2327585 -0.2130992  
## 5 -0.04463561 -0.03712769 1.9916622 -0.09294794 -0.2327585 -0.2130992  
## 6 -0.04463561 -0.03712769 1.9916622 -0.09294794 -0.2327585 -0.2130992  
## 20 21 22 23 24 25  
## 1 -0.3149516 -0.2948736 -0.04632411 -0.06318727 -0.01236832 -0.01749279  
## 2 -0.3149516 -0.2948736 -0.04632411 -0.06318727 -0.01236832 -0.01749279  
## 3 -0.3149516 -0.2948736 -0.04632411 -0.06318727 -0.01236832 -0.01749279  
## 4 -0.3149516 -0.2948736 -0.04632411 -0.06318727 -0.01236832 -0.01749279  
## 5 -0.3149516 -0.2948736 -0.04632411 -0.06318727 -0.01236832 -0.01749279  
## 6 -0.3149516 -0.2948736 -0.04632411 -0.06318727 -0.01236832 -0.01749279  
## 26 27 28 29 30 31 32  
## 1 -0.1604046 -0.146842 -0.1424332 -0.1271384 -0.05254272 -0.1077244 -0.01236832  
## 2 -0.1604046 -0.146842 -0.1424332 -0.1271384 -0.05254272 -0.1077244 -0.01236832  
## 3 -0.1604046 -0.146842 -0.1424332 -0.1271384 -0.05254272 -0.1077244 -0.01236832  
## 4 -0.1604046 -0.146842 -0.1424332 -0.1271384 -0.05254272 -0.1077244 -0.01236832  
## 5 -0.1604046 -0.146842 -0.1424332 -0.1271384 -0.05254272 -0.1077244 -0.01236832  
## 6 -0.1604046 -0.146842 -0.1424332 -0.1271384 -0.05254272 -0.1077244 -0.01236832  
## 33 cluster\_number  
## 1 -0.04288122 4  
## 2 -0.04288122 4  
## 3 -0.04288122 2  
## 4 -0.04288122 1  
## 5 -0.04288122 1  
## 6 -0.04288122 2

1. When using k-means clustering, it’s important to scale and center each variable to have a mean of 0 and a standard deviation of 1. If you don’t do this, variables with larger magnitudes will have a stronger influence over the distance calculations, which are the backbone of the k-means clustering algorithm.
2. Below are the cluster sizes and centers.

cat("Cluster Sizes:\n")

## Cluster Sizes:

table(km\_clusters$cluster)

##   
## 1 2 3 4   
## 1351 3157 1301 728

cat("\nCluster Centers:\n")

##   
## Cluster Centers:

print(km\_clusters$centers)

## [,1] [,2] [,3]  
## 1 -0.19620627 1.2490225 -0.16603786  
## 2 -0.50034715 -0.6086650 -0.39024626  
## 3 1.45027312 -0.3443423 -0.08881702  
## 4 -0.05787734 0.9369716 2.15916969

1. Below are the means of each variable within each cluster.

cluster\_means <- aggregate(. ~ cluster\_number, data = training\_features, FUN = mean)  
cluster\_means

## cluster\_number 1 2 3 4 5  
## 1 1 0.47871656 0.080860993 -0.23585026 -0.19620627 -0.2234682  
## 2 2 -0.40707450 -0.000246403 0.16902520 -0.50034715 -0.2411330  
## 3 3 0.46392154 -0.005045708 0.01549274 1.45027312 0.9370052  
## 4 4 0.04783819 -0.139973683 -0.32298753 -0.05787734 -0.2141229  
## 6 7 8 9 10 11  
## 1 1.2490225 0.02089322 0.21821363 0.13182659 -0.16603786 -0.230927094  
## 2 -0.6086650 -0.03907797 -0.13299778 -0.06553235 -0.39024626 0.004544435  
## 3 -0.3443423 -0.04315230 0.01728938 0.04725255 -0.08881702 0.203753018  
## 4 0.9369716 0.20780707 0.14089821 -0.04490065 2.15916969 0.044715722  
## 12 13 14 15 16 17  
## 1 -0.4254123 -0.2549411 -0.04463561 -0.03712769 0.4485717 -0.09294794  
## 2 0.1813364 0.2345813 0.00513123 -0.03712769 -0.1939598 0.09263985  
## 3 0.4326636 -0.1618375 0.05887636 0.14942398 -0.2681741 -0.07626845  
## 4 -0.7701133 -0.2549411 -0.04463561 -0.03712769 0.4879192 -0.09294794  
## 18 19 20 21 22 23  
## 1 0.15270854 -0.12233246 -0.131562595 -0.155741451 0.04973829 -0.016150297  
## 2 -0.04198334 0.01529481 -0.009877884 -0.008848471 -0.02576974 -0.007832783  
## 3 -0.13877941 0.20916343 0.304639219 0.317038658 0.03680445 0.022290951  
## 4 0.14668159 -0.21309916 -0.257431428 -0.239184024 -0.04632411 0.024102499  
## 24 25 26 27 28 29  
## 1 -0.01236832 0.06714847 0.14721176 0.089996424 -0.11592614 -0.05024130  
## 2 -0.01236832 -0.01749279 -0.07534443 -0.034473317 0.11846409 -0.07398064  
## 3 0.04977750 -0.01749279 0.03617305 -0.002485562 -0.08738168 0.24755357  
## 4 -0.01236832 -0.01749279 -0.01110144 -0.013075813 -0.14243315 -0.02834384  
## 30 31 32 33  
## 1 -0.05254272 -0.10772436 -0.01236832 -0.04288122  
## 2 0.05625416 0.08856669 -0.01236832 0.04591021  
## 3 -0.05254272 -0.04277179 -0.01236832 -0.04288122  
## 4 -0.05254272 -0.10772436 0.09869173 -0.04288122

1. Below I’ve assigned the observations in test\_features to the clusters found above.

library(clue)  
  
# Predict clusters for test\_features  
test\_cluster <- cl\_predict(km\_clusters, newdata = test\_features)  
  
# Add the predicted cluster numbers to test\_features  
test\_features <- cbind(test\_features, cluster\_number = test\_cluster)  
  
# View the output  
head(test\_features)

##   
## [1,] -1.3069228 1.5584548 -0.963919934 -1.1342230 -0.80560473 -1.139532803  
## [2,] -0.1175595 -0.9942680 0.272649297 1.5085335 0.29762426 0.223494513  
## [3,] -0.1642717 0.6654528 -0.276937028 -0.9599236 -0.68679545 1.690230430  
## [4,] -0.6385797 -0.8950455 1.509218527 0.1800890 0.97653440 -1.305724225  
## [5,] -1.4075336 1.7208187 -0.276937028 -0.6914082 -0.09274908 -0.961102479  
## [6,] 1.2766185 -0.4801153 -0.002143866 -0.6772758 -0.39825864 0.002550104  
##   
## [1,] -0.3459840 0.4966769 0.6864990 -0.37545829 -0.2309271 -0.8917759  
## [2,] 0.3479892 -1.0091770 -0.7788883 -0.05032105 -0.2309271 -0.8917759  
## [3,] -0.4230921 1.4313448 1.1313488 -0.35651090 -0.2309271 -0.8917759  
## [4,] 0.5022055 -2.3592529 -1.8517612 -0.04198420 -0.2309271 -0.8917759  
## [5,] -0.6544165 -0.1264351 0.5556609 -0.57251116 -0.2309271 1.1211864  
## [6,] 1.1190706 -0.1264351 -0.9358941 0.54462705 -0.2309271 -0.8917759  
##   
## [1,] -0.2549411 -0.04463561 -0.03712769 -0.5020164 -0.09294794 -0.2327585  
## [2,] -0.2549411 -0.04463561 -0.03712769 1.9916622 -0.09294794 -0.2327585  
## [3,] -0.2549411 -0.04463561 -0.03712769 -0.5020164 -0.09294794 4.2956413  
## [4,] -0.2549411 -0.04463561 -0.03712769 1.9916622 -0.09294794 -0.2327585  
## [5,] -0.2549411 -0.04463561 -0.03712769 -0.5020164 -0.09294794 -0.2327585  
## [6,] -0.2549411 -0.04463561 -0.03712769 -0.5020164 -0.09294794 -0.2327585  
##   
## [1,] -0.2130992 -0.3149516 -0.2948736 -0.04632411 -0.06318727 -0.01236832  
## [2,] -0.2130992 -0.3149516 -0.2948736 -0.04632411 -0.06318727 -0.01236832  
## [3,] -0.2130992 -0.3149516 -0.2948736 -0.04632411 -0.06318727 -0.01236832  
## [4,] -0.2130992 -0.3149516 -0.2948736 -0.04632411 -0.06318727 -0.01236832  
## [5,] -0.2130992 -0.3149516 -0.2948736 -0.04632411 -0.06318727 -0.01236832  
## [6,] -0.2130992 -0.3149516 -0.2948736 -0.04632411 -0.06318727 -0.01236832  
##   
## [1,] -0.01749279 -0.1604046 -0.146842 -0.1424332 -0.1271384 -0.05254272  
## [2,] -0.01749279 -0.1604046 -0.146842 -0.1424332 -0.1271384 -0.05254272  
## [3,] -0.01749279 -0.1604046 -0.146842 -0.1424332 -0.1271384 -0.05254272  
## [4,] -0.01749279 -0.1604046 -0.146842 -0.1424332 -0.1271384 -0.05254272  
## [5,] -0.01749279 -0.1604046 -0.146842 -0.1424332 -0.1271384 -0.05254272  
## [6,] -0.01749279 -0.1604046 6.808999 -0.1424332 -0.1271384 -0.05254272  
## cluster\_number  
## [1,] -0.1077244 -0.01236832 -0.04288122 2  
## [2,] -0.1077244 -0.01236832 -0.04288122 2  
## [3,] -0.1077244 -0.01236832 -0.04288122 2  
## [4,] -0.1077244 -0.01236832 -0.04288122 2  
## [5,] -0.1077244 -0.01236832 -0.04288122 2  
## [6,] -0.1077244 -0.01236832 -0.04288122 2