Week8_SimonsenHomework

Steven Simonsen

2024-10-15

```
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
```

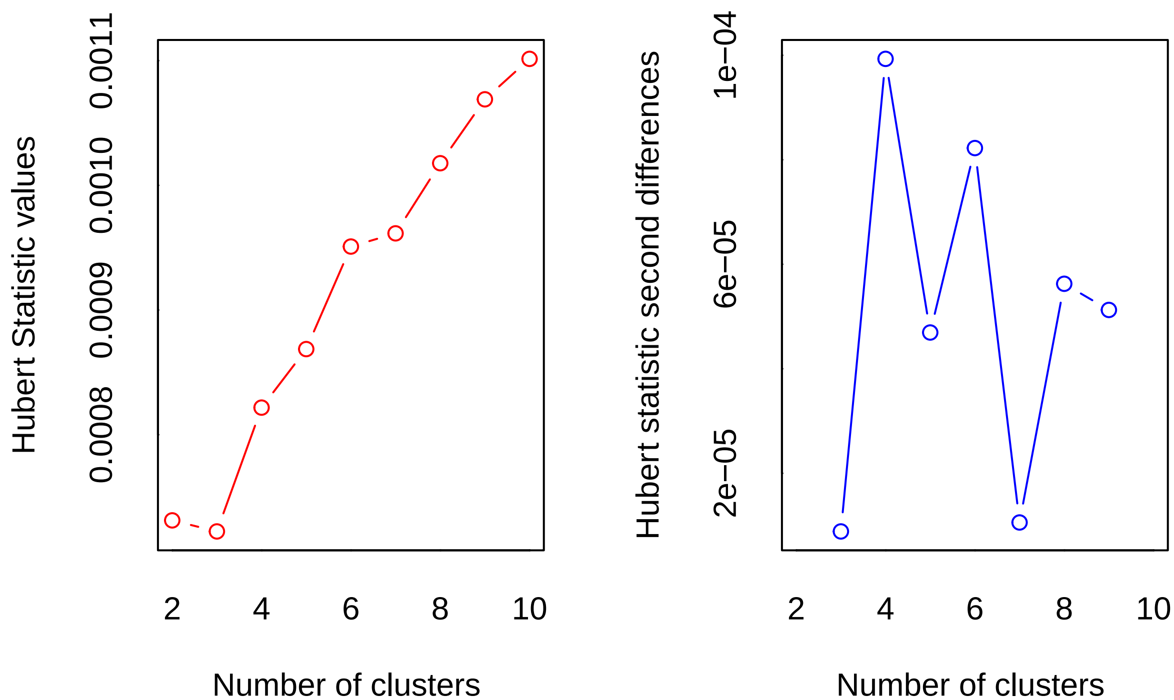
```
library(NbClust)

setwd("C:\\Users\\steve\\OneDrive\\Documents\\School\\DSE6211\\Week8")

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
```

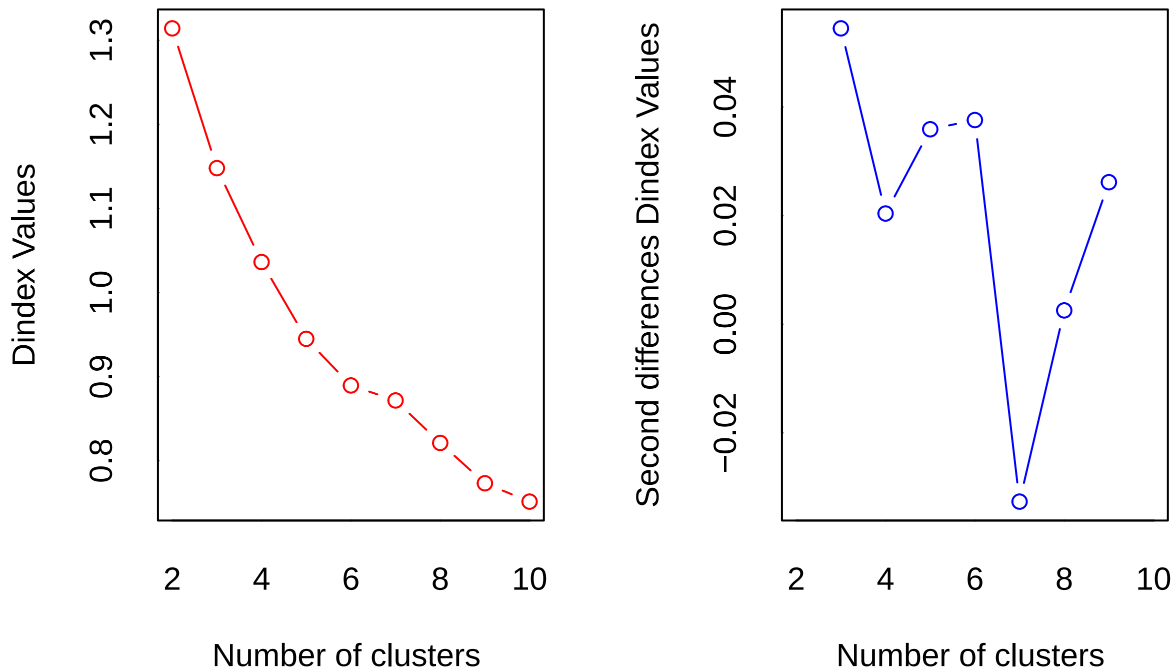
```
training_set$soil_type <- ifelse(training_set$soil_type %in%
                                   top_20_soil_types$soil_type,
                                 training_set$soil_type,
                                 "other")
```

```r
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)))
```

```r
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:
## * 6 proposed 2 as the best number of clusters
## * 5 proposed 3 as the best number of clusters
## * 7 proposed 4 as the best number of clusters
## * 4 proposed 9 as the best number of clusters
## * 1 proposed 10 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 -1.6114538 -0.5680384 -0.9792743 -0.41298603 -0.5009574 -1.0490066  0.7065639
## 2 -0.3640291 -0.2733056 -0.8437867 -0.00681951 -0.6894465  0.6082246  0.7822666
## 3 -1.2986989  1.5308164 -0.9792743 -1.13086174 -0.8093941 -1.1394245 -0.3154219
## 4 -0.1087924 -0.9967407  0.2401137  1.51866638  0.3044052  0.2271777  0.3659020
## 5 -0.1555259  0.6466180 -0.3018365 -0.95611568 -0.6894465  1.6977605 -0.3911246
## 6 -1.0542468 -0.8270460 -0.0308614 -0.27130004 -0.4838220 -0.7557942  0.7444152
##            8          9          10         11         12         13
## 1  0.3047499 -0.2846374  2.69809322 -0.2253618 -0.8939894 -0.2596715
## 2  0.6094299 -0.2064688  0.81638142 -0.2253618 -0.8939894 -0.2596715
## 3  0.5078699  0.6794428 -0.37077984 -0.2253618 -0.8939894 -0.2596715
## 4 -0.9647500 -0.7797057 -0.04262766 -0.2253618 -0.8939894 -0.2596715
## 5  1.4219098  1.1223986 -0.35165676 -0.2253618 -0.8939894 -0.2596715
## 6 -0.6092900 -0.8578743 -1.32999391 -0.2253618 -0.8939894 -0.2596715
##             14          15         16          17         18         19
## 1 -0.04795369 -0.03273853 -0.5001052 -0.08953925 -0.2349412 -0.2154418
## 2 -0.04795369 -0.03273853 -0.5001052 -0.08953925 -0.2349412 -0.2154418
## 3 -0.04795369 -0.03273853 -0.5001052 -0.08953925 -0.2349412 -0.2154418
## 4 -0.04795369 -0.03273853  1.9992736 -0.08953925 -0.2349412 -0.2154418
## 5 -0.04795369 -0.03273853 -0.5001052 -0.08953925  4.2557331 -0.2154418
## 6 -0.04795369 -0.03273853 -0.5001052 -0.08953925 -0.2349412 -0.2154418
##          20         21          22          23          24          25
## 1 -0.315538 -0.2933388 -0.05398665 -0.06195546 -0.01749279 -0.01236832
## 2 -0.315538 -0.2933388 -0.05398665 -0.06195546 -0.01749279 -0.01236832
## 3 -0.315538 -0.2933388 -0.05398665 -0.06195546 -0.01749279 -0.01236832
## 4 -0.315538 -0.2933388 -0.05398665 -0.06195546 -0.01749279 -0.01236832
## 5 -0.315538 -0.2933388 -0.05398665 -0.06195546 -0.01749279 -0.01236832
## 6 -0.315538 -0.2933388 -0.05398665 -0.06195546 -0.01749279 -0.01236832
##           26         27         28         29          30         31
## 1 -0.1604046 -0.1563453 -0.1390423 -0.1265158 -0.05105843 -0.1055234
## 2 -0.1604046 -0.1563453 -0.1390423 -0.1265158 -0.05105843 -0.1055234
## 3 -0.1604046 -0.1563453 -0.1390423 -0.1265158 -0.05105843 -0.1055234
## 4 -0.1604046 -0.1563453 -0.1390423 -0.1265158 -0.05105843 -0.1055234
## 5 -0.1604046 -0.1563453 -0.1390423 -0.1265158 -0.05105843 -0.1055234
## 6 -0.1604046 -0.1563453 -0.1390423 -0.1265158 -0.05105843 -0.1055234
##            32          33 cluster_number
## 1 -0.01749279 -0.04288122              4
## 2 -0.01749279 -0.04288122              3
## 3 -0.01749279 -0.04288122              2
## 4 -0.01749279 -0.04288122              1
## 5 -0.01749279 -0.04288122              3
## 6 -0.01749279 -0.04288122              2
```

Exercises

1) Why is it good practice to center and scale before applying k-means clustering? It is important to center using a mean of 0 and scale the data, typically to a standard deviation of 1, to avoid variables with larger magnitudes having more influence on the clusters, since they will dominate the distance calculation.

2) Print the cluster sizes and centers to the R console. Include a screenshot of the output.

```r
km_clusters$size
```

```
## [1] 1268 3220 1326  723
```

```r
km_clusters$centers
```

```
##          [,1]       [,2]        [,3]
## 1  1.47956864 -0.2889577 -0.02882224
## 2 -0.48366329 -0.6139885 -0.39437417
## 3 -0.21139514  1.2421734 -0.19289360
## 4 -0.05309446  0.9630976  2.16073076
```

3) Use the aggregate() function to calculate the mean of each variable within each cluster. Include a screenshot of the output.

```r
aggregate(training_features[,-c(11:13)], by=list(training_features$cluster_number), mean)
```

```
##   Group.1           1            2            3           4          5
## 1       1  0.47307130 -0.017354926 -0.007460947  1.47956864  0.9190489
## 2       2 -0.39836362  0.009880297  0.176956314 -0.48366329 -0.2275790
## 3       3  0.48921083  0.084855563 -0.227553364 -0.21139514 -0.2189215
## 4       4  0.04727926 -0.169193621 -0.357680621 -0.05309446 -0.1967630
##            6           7          8           9          10           14
## 1 -0.2889577 -0.02019353  0.0167296  0.03247584 -0.02882224  0.083896263
## 2 -0.6139885 -0.05340142 -0.1246407 -0.05228389 -0.39437417 -0.002522781
## 3  1.2421734  0.02127219  0.2164019  0.13139752 -0.19289360 -0.047953688
## 4  0.9630976  0.23423379  0.1288811 -0.06508762  2.16073076 -0.047953688
##            15         16          17          18          19           20
## 1  0.13604048 -0.2418886 -0.07178512 -0.12161188  0.18670121  0.294479346
## 2 -0.03273853 -0.1927281  0.08524518 -0.06061378  0.03190053  0.004752669
## 3 -0.03273853  0.4442303 -0.08953925  0.16806803 -0.13853121 -0.144741782
## 4 -0.03273853  0.4678424 -0.08953925  0.17499587 -0.21544183 -0.272165701
##             21          22            23          24          25          26
## 1  0.278870672  0.03390412  0.0147012294  0.07268887 -0.01236832  0.04128903
## 2  0.009016867 -0.02514465 -0.0066134568 -0.01749279 -0.01236832 -0.07502304
## 3 -0.153751874  0.05807519 -0.0008690585 -0.01749279  0.04860582  0.15301146
## 4 -0.247257728 -0.05398665  0.0052649287 -0.01749279 -0.01236832 -0.01891219
##             27          28           29          30          31          32
## 1 -0.006508853 -0.07545402  0.266091643 -0.05105843 -0.05263340 -0.01749279
## 2 -0.040372072  0.10680851 -0.081630670  0.05259652  0.08787472 -0.01749279
## 3  0.105515815 -0.11140281 -0.059906465 -0.05105843 -0.10552340 -0.01749279
## 4 -0.002299684 -0.13904232  0.006753151 -0.05105843 -0.10552340  0.14066813
##            33 cluster_number
## 1 -0.04288122              1
## 2  0.04417298              2
## 3 -0.04288122              3
## 4 -0.04288122              4
```