lab08.R

mirrien

2022-04-10

```r
# Question 1

# Helper function to calculate L1 distances

manhattan <- function(p1,p2) {
  distance <- matrix(NA, nrow=dim(p1)[1], ncol=dim(p2)[1])
  for(i in 1:nrow(p2)) {
    distance[,i] = rowSums(abs(t(t(p1)-p2[i,])))
  }
  return(distance)
}


# Another function for calculating L1 distances

manhattan2<-function(x,K){
  distance = matrix(NA, nrow= nrow(x), ncol = nrow(K))
  for(j in 1:nrow(K)) {
    for(i in 1:nrow(x)) {
      distance[i,j]<-dist(rbind(x[i,],K[j,]), method = "manhattan")
    }
  }
  return(distance)
}


kmedian <- function(x,K,iters) {
  # convert df to matrix
  x = as.matrix(x)

  # randomly sample some centers, set a seed 100
  set.seed(100)
  K <- x[sample(nrow(x), K),]

  # empty lists to store outputs
  assignments <- vector(iters, mode = "list")
  locations <- vector(iters, mode = "list")

  for(i in 1:iters) {
    # call manhattan distance helper function
    dists = manhattan(x,K)
    # find minimum distance
    clusters <- apply(dists,1,which.min)
    # tapply median()
    centers <- apply(x,2,tapply,clusters,median)

    # store outputs
    assignments[[i]] <- clusters
    locations[[i]] <- centers
  }

  # return outputs in list
  return(list(locations=locations[[1]], assignments = assignments[[1]]))
}

# Test:

# df = read.csv("parkinsons.data",row.names = 1)
# kmedian(df,3,10)



#################################################
# Question 2

kmedian2 <- function(x,K,iters) {
  # convert df to matrix
  x = as.matrix(x)

  # randomly sample some centers, set a seed 100
  set.seed(100)
  K <- x[sample(nrow(x), K),]

  # empty lists to store outputs
  assignments <- vector(iters, mode = "list")
  locations <- vector(iters, mode = "list")

  # initialize cluster assignments using ((i-1)%K)+1

  for (i in 1:nrow(x)) {
    assignments[i] = ((i-1)%%K)+1
  }

  for(i in 1:iters) {
    # call manhattan distance helper function
    dists = manhattan(x,K)
    # find minimum distance
    clusters <- apply(dists,1,which.min)
    # tapply median()
    centers <- apply(x,2,tapply,clusters,median)

    # store outputs
    assignments[[i]] <- clusters
    locations[[i]] <- centers
  }
```

```r
  # return outputs in list
  return(list(locations=locations[[1]], assignments = assignments[[1]]))
}

df = read.csv("parkinsons.data",row.names = 1)
result = kmedian(df,3,1000)
print(result$locations)
```

```
##   MDVP.Fo.Hz. MDVP.Fhi.Hz. MDVP.Flo.Hz. MDVP.Jitter... MDVP.Jitter.Abs.
## 1     120.168     140.2120      97.5350       0.005275            4e-05
## 2     225.534     242.5295     202.2575       0.002735            1e-05
## 3     180.198     216.3020     109.3790       0.004600            3e-05
##   MDVP.RAP MDVP.PPQ Jitter.DDP MDVP.Shimmer MDVP.Shimmer.dB. Shimmer.APQ3
## 1 0.002685 0.003060   0.008065     0.024450           0.2305     0.013655
## 2 0.001545 0.001515   0.004635     0.016255           0.1430     0.008640
## 3 0.002370 0.002540   0.007100     0.025510           0.2550     0.014100
##   Shimmer.APQ5 MDVP.APQ Shimmer.DDA      NHR     HNR status      RPDE      DFA
## 1     0.014125 0.019525    0.040965 0.012295 22.1520      1 0.5393685 0.727867
## 2     0.009970 0.011410    0.025925 0.004760 24.8555      0 0.4294560 0.678466
## 3     0.015800 0.019090    0.042310 0.018020 20.3660      1 0.4699280 0.715121
##     spread1   spread2       D2      PPE
## 1 -5.514255 0.2318010 2.246647 0.217401
## 2 -7.162888 0.1706355 2.272559 0.095626
## 3 -5.845099 0.2180370 2.608749 0.186489
```

```r
#################################################
# Question 3

# Helper function to calculate Euclidean distance

euclid <- function(x,K){
  distance = matrix(NA, nrow= nrow(x), ncol = nrow(K))
  for(j in 1:nrow(K)) {
    for(i in 1:nrow(x)) {
      distance[i,j]<-dist(rbind(x[i,],K[j,]), method = "euclidean")
    }
  }
  return(distance)
}

mykmeans <- function(x,K,iters) {
  # convert df to matrix
  x = as.matrix(x)

  # randomly sample some centers, set a seed 100
  set.seed(100)
  K <- x[sample(nrow(x), K),]

  # empty lists to store outputs
  assignments <- vector(iters, mode = "list")
  locations <- vector(iters, mode = "list")

  for(i in 1:iters) {
    # call euclidean distance helper function
    dists = euclid(x,K)
    # find minimum distance
    clusters <- apply(dists,1,which.min)
    # tapply mean()
    centers <- apply(x,2,tapply,clusters,mean)

    # store outputs
    assignments[[i]] <- clusters
    locations[[i]] <- centers
  }

  # return outputs in list
  return(list(locations=locations[[1]], assignments = assignments[[1]]))
}

# df = read.csv("parkinsons.data",row.names = 1)
result2 = mykmeans(df,3,10)
set.seed(123)
result3 = kmeans(df,3,10)


# Compare cluster assignments
v1 = result2$assignments
v2 = c()
for (i in seq_along(result3$cluster)) {
  v2[i] = result3$cluster[[i]]
}

compare1 = data.frame(mykmeans = v1, kmeans = v2)
compare2 = data.frame(cluster = c(1,2,3),
                      count_mykmeans = c(sum(v1 == 1), sum(v1==2),sum(v1==3)),
                      count_kmeans = c(sum(v2==1),sum(v2==2),sum(v2==3))
)
# compare1
compare2
```

```
##   cluster count_mykmeans count_kmeans
## 1       1            109          121
## 2       2             24           63
## 3       3             62           11
```

```
# The comparisons show that cluster assignments may vary but the distribution of
# the sums of data points in each of the three clusters
# (i.e., the counts of cluster assignments: 109, 24, 62 versus 121, 63, 11)
# remains similar. It might be due to the different initial centroids of
# the two methods that were randomly generated. This is manifested in the
# differences between result2$locations and result3$centers.

compare3 = list(mykmeans = result2$locations,
                kmeans = result3$centers)
compare3
```

```
## $mykmeans
##   MDVP.Fo.Hz. MDVP.Fhi.Hz. MDVP.Flo.Hz. MDVP.Jitter... MDVP.Jitter.Abs.
## 1    126.7810     147.9497      95.9625    0.006437248     5.220183e-05
## 2    222.9895     250.3461     208.3466    0.004764167     2.175000e-05
## 3    175.8662     262.9134     116.5012    0.006403065     3.806452e-05
##      MDVP.RAP    MDVP.PPQ  Jitter.DDP MDVP.Shimmer MDVP.Shimmer.dB.
## 1 0.003384679 0.003536147 0.010154771   0.03048220        0.2844312
## 2 0.002736667 0.002793333 0.008210833   0.02052958        0.2035833
## 3 0.003389355 0.003541290 0.010168710   0.03190339        0.3088710
##   Shimmer.APQ3 Shimmer.APQ5   MDVP.APQ Shimmer.DDA        NHR      HNR
## 1   0.01611431   0.01808248 0.02444110  0.04834312 0.02370156 21.78672
## 2   0.01081833   0.01269833 0.01614708  0.03245542 0.01479917 24.66696
## 3   0.01674855   0.01952435 0.02652065  0.05024565 0.03075048 20.98397
##      status      RPDE       DFA   spread1   spread2       D2       PPE
## 1 0.8532110 0.5298757 0.7322895 -5.418370 0.2324567 2.297751 0.2256944
## 2 0.2083333 0.4163468 0.6901854 -6.830502 0.1597985 2.204304 0.1225370
## 3 0.7903226 0.4752525 0.7039565 -5.708436 0.2418802 2.598355 0.2054193
##
## $kmeans
##   MDVP.Fo.Hz. MDVP.Fhi.Hz. MDVP.Flo.Hz. MDVP.Jitter... MDVP.Jitter.Abs.
## 1    129.5000     150.6280     99.52977    0.006151488     4.933884e-05
## 2    202.0870     231.5896    153.07944    0.005914603     3.098413e-05
## 3    152.1458     510.8475     90.56327    0.008730909     5.909091e-05
##      MDVP.RAP    MDVP.PPQ  Jitter.DDP MDVP.Shimmer MDVP.Shimmer.dB.
## 1 0.003229669 0.003383802 0.009689504   0.03058983        0.2853554
## 2 0.003231111 0.003361587 0.009694762   0.02802937        0.2714603
## 3 0.004581818 0.004620000 0.013744545   0.02964182        0.3099091
##   Shimmer.APQ3 Shimmer.APQ5   MDVP.APQ Shimmer.DDA        NHR      HNR
## 1   0.01621000   0.01820438 0.02444893  0.04863008 0.02246413 21.86427
## 2   0.01468730   0.01749444 0.02348810  0.04406190 0.02481698 21.88935
## 3   0.01525455   0.01648909 0.02343818  0.04576545 0.05123182 22.10536
##      status      RPDE       DFA   spread1   spread2       D2       PPE
## 1 0.8677686 0.5191681 0.7318962 -5.508738 0.2286055 2.307389 0.2184998
## 2 0.5396825 0.4605148 0.6977117 -6.069726 0.2160535 2.499325 0.1807017
## 3 0.7272727 0.4893328 0.6830941 -5.409763 0.2633536 2.527684 0.2231709
```

```
# The other reason might be due to the different names created
# for the 3 clusters. Because the names (i.e., 1, 2, and 3) are just an
# indicator of three different groups rather than something meaningful. There
# is no real measurements differetiating the three groups. If we plot out the
# distribution of the clustering, we would find the distributions of two methods
# are similar.

##################################################
# Question 4

# Three benefits:
# Because the method doesn't require the data to be labeled,
# it is frequently utilized in a variety of real-world problem statements.
# K-means clustering is easy to implement.
# The approach can handle massive amounts of data.


# Three drawbacks:
```

```r
# The value of K must be manually selected.
# Outliers would have an adverse impact on the clustering.
# Clusters cannot overlap: one point can only belong to one cluster at a time,
# leading to certain points placed in incorrect clusters.

#################################################
# Question 5


suppressWarnings(suppressMessages(library(ggpubr)))
suppressWarnings(suppressMessages(library(factoextra)))

# Create a test data frame
set.seed(100)
sample_df = data.frame(V1 = rnorm(50,0,10), V2 = rnorm(50,0,10))
# head(sample_df)

# Use mykmeans()

result01 = mykmeans(scale(sample_df),3,1000)
result01$assignments
```

```
##  [1] 1 1 2 2 1 2 1 2 2 1 1 2 1 2 2 1 2 2 1 3 1 1 1 2 1 1 1 2 1 1 2 3 1 1 2 1 2 2
## [39] 2 3 1 3 1 1 1 3 1 3 1 1
```

```r
# Plot

plot1 = fviz_cluster(list(data = sample_df, cluster = result01$assignments),
             data = sample_df,
             palette = c("#2E9FDF", "#00AFBB", "#E7B800"),
             geom = "point",
             ellipse.type = "convex",
             ggtheme = theme_bw()
)


# Use kmeans()

set.seed(222)
result02 = kmeans(scale(sample_df),3,1000)
result02$cluster
```

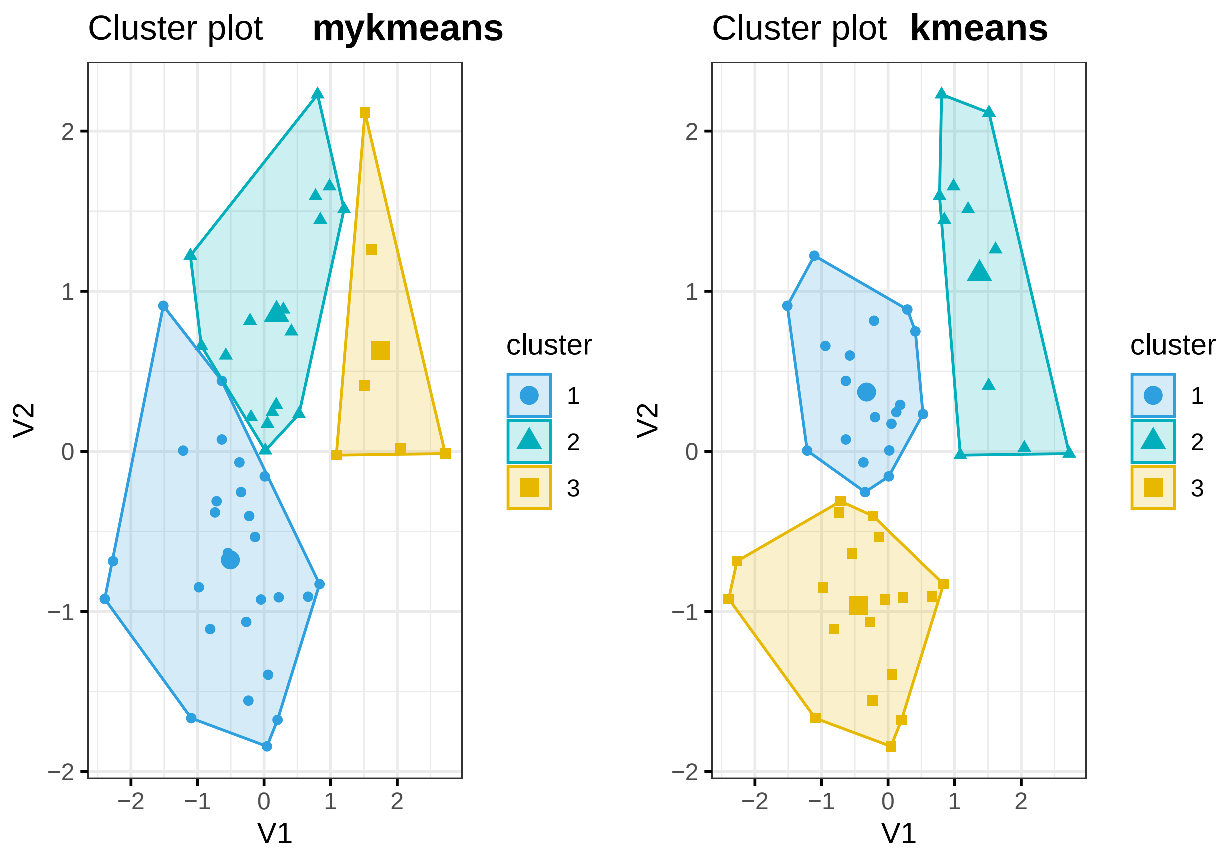
```
##  [1] 3 3 1 2 3 1 3 2 1 3 1 1 1 2 1 3 1 1 1 2 1 3 3 2 3 1 3 1 1 3 1 2 3 3 1 1 1 1
## [39] 2 2 3 2 3 3 3 2 3 2 3 3
```

```r
# Plot

plot2 = fviz_cluster(result02, data = sample_df,
             palette = c("#2E9FDF", "#00AFBB", "#E7B800"),
             geom = "point",
             ellipse.type = "convex",
             ggtheme = theme_bw()
)

# Combine two plots to visualize comparison
figure <- ggarrange(plot1, plot2, labels = c("mykmeans", "kmeans"),
                    label.x = 0.35, label.y = 1,
                    ncol=2,nrow=1)

# Output the combined plot
figure
```


Cluster plot
mykmeans
Cluster plot
kmeans
V1
V2
cluster