Assignment-1-573

Chi

2019-02-18

Q1

a)

```
library(gclus)
```

```
## Loading required package: cluster
```

```
data(bank)
head(bank,3)
```

```
##   Status Length  Left Right Bottom Top Diagonal
## 1      0  214.8 131.0 131.1    9.0 9.7    141.0
## 2      0  214.6 129.7 129.7    8.1 9.5    141.7
## 3      0  214.8 129.7 129.7    8.7 9.6    142.2
```

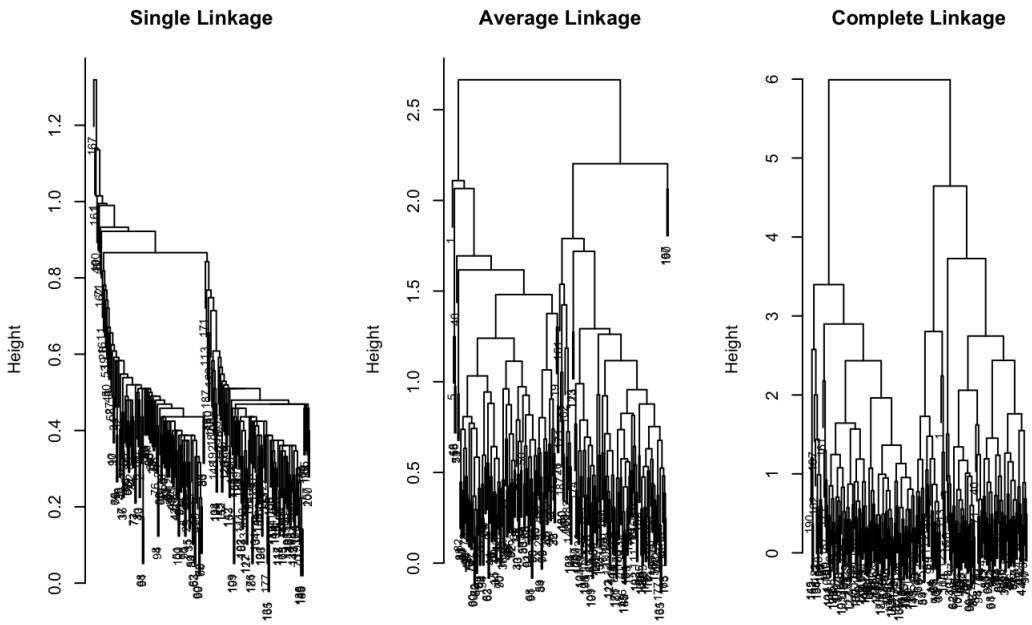
```
df <- subset(bank, select = -c(Bottom,Status))
head(df,3)
```

```
##   Length  Left Right Top Diagonal
## 1  214.8 131.0 131.1 9.7    141.0
## 2  214.6 129.7 129.7 9.5    141.7
## 3  214.8 129.7 129.7 9.6    142.2
```

The euclidean distance measure is appropriate since all the infromation on the bank note is measured on the same scale, which in this case is (mm)。

b)

```
euclid <- dist(df,method="euclidean")
single <- hclust(euclid, method ="single")
average <- hclust(euclid, method ="average")
complete <- hclust(euclid, method ="complete")

#dendrograms
par(mfrow =c(1,3))
plot(single , main= "Single Linkage", xlab="", sub ="",cex =.7)
plot(average , main = "Average Linkage", xlab="", sub ="",cex =.7)
plot(complete ,main = "Complete Linkage", xlab="", sub ="",cex =.7)
```



c)

I would choose ‘average linkage’ with the largest jump at 2 clusters.

It may seem that complete linkage will work alright but after looking at the misclassfication the complete linkage doesn’t perform well either for 2 or 3 clusters.

In addtion, single linkage has the chaining phenomenon where clusters may be forced together due to single elements being close to each other, even though many of the elements in each cluster may be very distant to each other (wiki).

d)

```
#cut at largest jump 2
table(cutree(average,2), bank$Status)
```

```
##
##        0   1
##   1   99   0
##   2    1 100
```

misclassification rate is (1/200) = 0.005 (.5 percent), which is very low.

This concludes Average linkage is doing well by clustering the data into two clusters.

e)

```
set.seed(632)
dfscaled <- kmeans(scale(df), 2)
table(dfscaled$cluster,bank$Status)
```

```
##
##       0  1
##   1 19 97
##   2 81  3
```

Missclassiciation rate is (22/200) = 0.11 (11 percent).

f)

```
set.seed(632)
dfunscaled <- kmeans(df,2)
table(dfunscaled$cluster,bank$Status)
```

```
##
##        0   1
##   1    1 100
##   2   99   0
```

```
#variances of each explantory variable
var(df$Length)
```

```
## [1] 0.141793
```

```
var(df$Left)
```

```
## [1] 0.1303394
```

```
var(df$Right)
```

```
## [1] 0.1632741
```

```
var(df$Top)
```

```
## [1] 0.6447234
```

```
var(df$Diagonal)
```

```
## [1] 1.327716
```

The reason here is that the measurements are already in the same units (mm). In addtion, k-means tend to produce some what a round cluster. In this case, unequal variances suggest that more weight is placed on variables with smaller variance, which might be the reason why it is clustering better.

g)

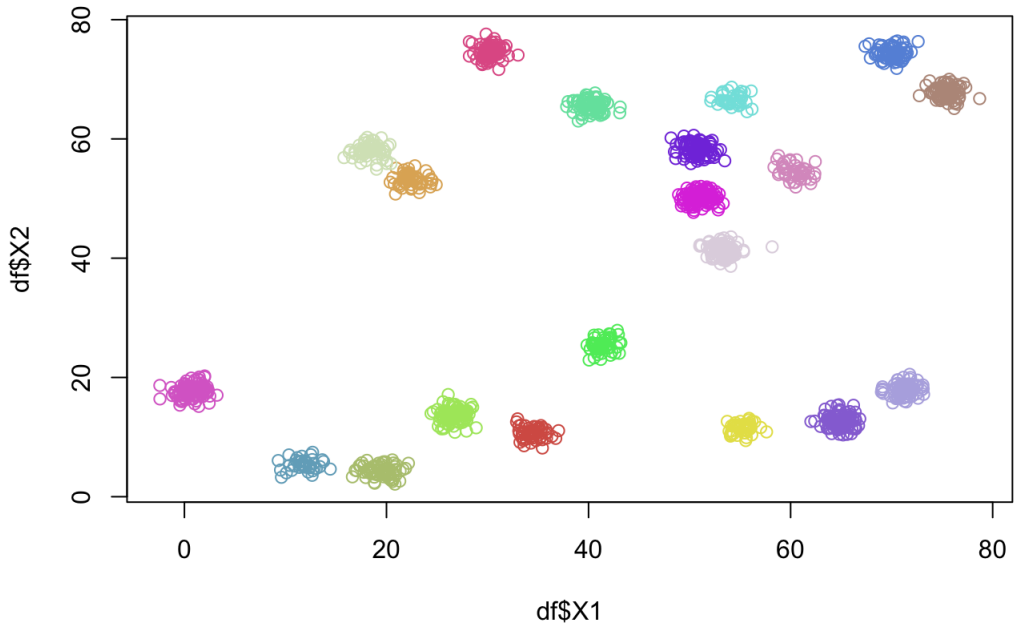
Overall, what does the (generally) strong performance of unsupervised methods signify for this data set?

The generally strong performance of 2 cluster hierarchical clustering and kmeans suggest that there are 2 groups in the data. This is true since we know that there are only couterfeit and not counterfeit in the status column.

Q2

a)

```
lots <- load("lots.Rdata")
df <- data.frame(clusts,datmat)
#install.packages("randomcoloR")
library(randomcoloR)
palette <- distinctColorPalette(20)
plot(df$X1, df$X2, col=palette[clusts])
```



b)

```
library(mclust)
```

```
## Package 'mclust' version 5.4.2
## Type 'citation("mclust")' for citing this R package in publications.
```

```
set.seed(461)
table(df$clusts)
```

```
##
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
## 44 87 95 45 86 77 67 98 57 40 56 90 86 75 98 97 64 56 40 60
```

```
kmeans1 <- kmeans(datmat, 20)
adjustedRandIndex(kmeans1$cluster,df$clusts)
```

```
## [1] 0.8317588
```

Adjusted Rand Index: 0.8317588

c)

```
library(mclust)
set.seed(41)
kmeans2 <- kmeans(datmat, 20)
adjustedRandIndex(kmeans2$cluster,df$clusts)
```

```
## [1] 0.6747311
```

Adjusted Rand Index: 0.6747311

d)

```
library(mclust)
set.seed(461)
kmeans3 <- kmeans(datmat, 20, nstart = 1000)
adjustedRandIndex(kmeans3$cluster,df$clusts)
```

```
## [1] 0.9438239
```

Adjusted Rand Index: 0.9438239

e)

```
library(mclust)
set.seed(41)
kmeans4 <- kmeans(datmat, 20, nstart = 1000)
adjustedRandIndex(kmeans4$cluster,df$clusts)
```

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

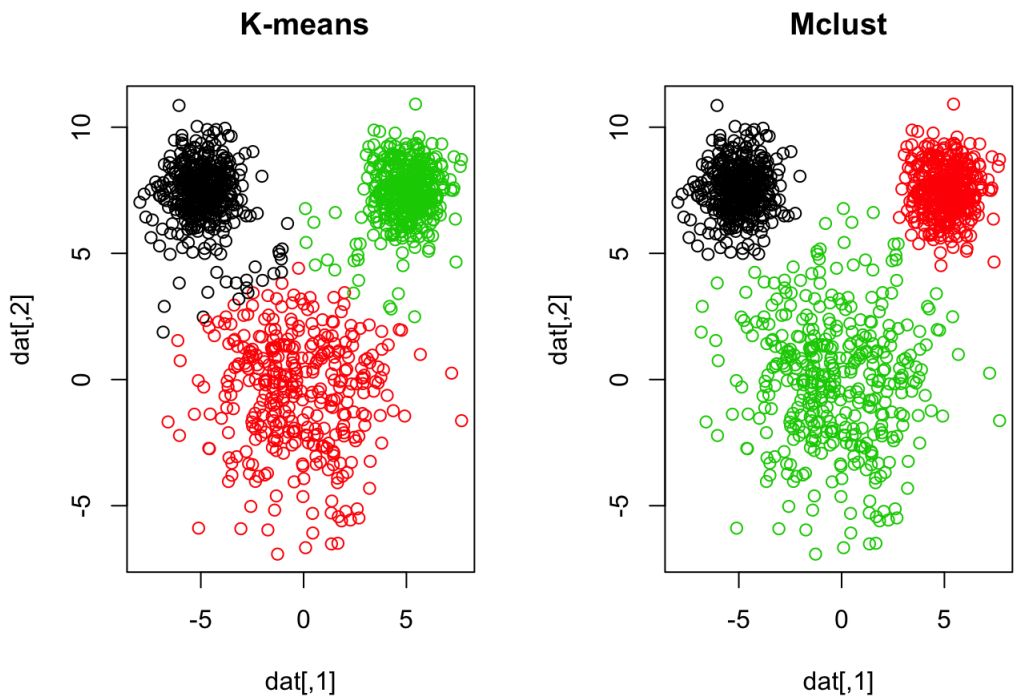
Adjusted Rand Index: 1

f)

The adjusted rand index increases in accuracy when the nstart is increased up to 1000. Since without specifying nstart kmeans start with only 1 configurations, which gives a lower adjusted rand index. This also tells us k-means works with trial and error so running it a few different times may give different results.

Q3

```r
my_k_means <- function(x, k){
  #1 start with k centroids
  centrs <- x[sample(1:nrow(x), k),]
  #start loop
  changing <- TRUE
  while(changing){
    #2a) calculate distances between all x and centrs
    dists <- matrix(NA, nrow(x), k)
    #could write as a double loop, or could utilize other built in functions probably
    for(i in 1:nrow(x)){
      for(j in 1:k){
        dists[i, j] <- sqrt(sum((x[i,] - centrs[j,])^2))
      }
    }
    #2b) assign group memberships (you probably want to provide some overview of apply functions)
    membs <- apply(dists, 1, which.min)

    #3) calculate new group centroids
    oldcentrs <- centrs #save for convergence check!
    for(j in 1:k){
      centrs[j,] <- colMeans(x[membs==j, ])
    }

    #4) check for convergence
    if(all(centrs==oldcentrs)){
      changing <- FALSE
    }
  }
  #output memberships
  membs
}

#install.packages("mvtnorm")
library(mclust)

library(mvtnorm)
set.seed(35151)
le <- rmvnorm(400, mean = c(-5,7.5))
re <- rmvnorm(400, mean = c(5,7.5))
hd <- rmvnorm(400, mean = c(0,0), sigma=7*diag(2) )
dat <- rbind(le, re, hd)
par(mfrow=c(1,2))

mickres <- my_k_means(scale(dat), 3)

plot(dat, col=mickres, main =" K-means")

clust1 <- Mclust(scale(dat))
plot(dat,col=clust1$classification,main ="Mclust")
```



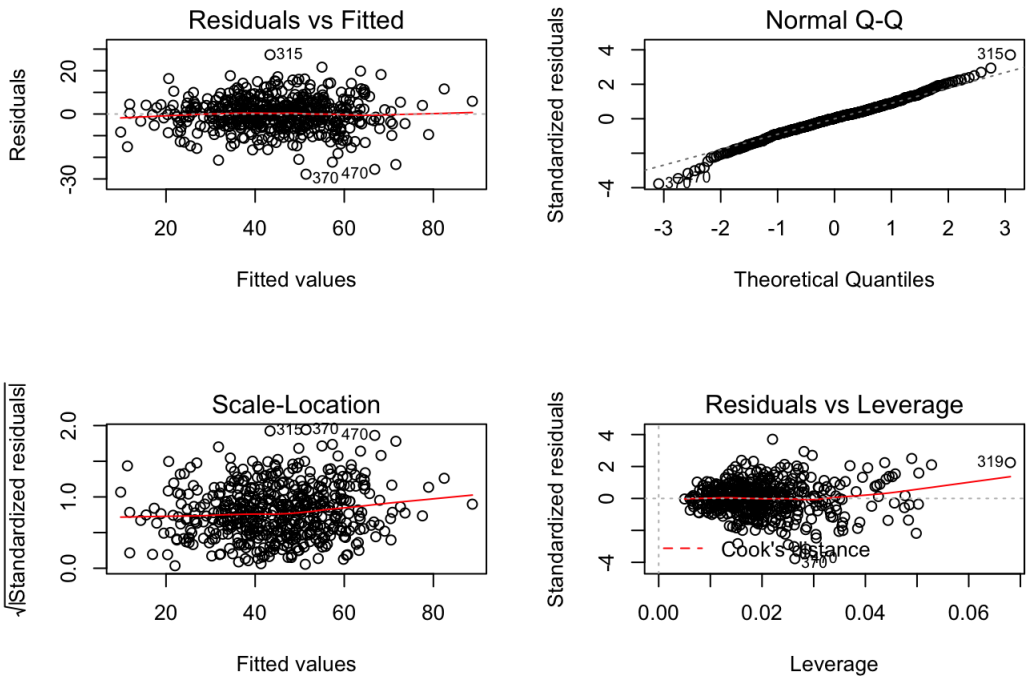
Yes, Mclust results seem more sensible than k-means. The reason here as mentioned in class is that k-means is a highly restricted mixture model. Unlike Mclust there is no covariance in k-means assuming independence among the variables. This is why we can see that in k-means the clustering around the ears of the mouse is not clustered well, but Mclust is clustering the ears well.

Q4

```
sim<- load("asim.Rdata")
df <- data.frame(asim)
mod1 <- lm(y~., data = df)

#check some assumptions first
summary(mod1) #so adjusted R squared = .735 for the full model
```

```
##
## Call:
## lm(formula = y ~ ., data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -27.8122  -4.4968   0.1784   4.4572  27.3810
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -37.2355     5.2878  -7.042 6.45e-12 ***
## V2            2.3483     0.1704  13.785  < 2e-16 ***
## V3            1.5074     0.1369  11.009  < 2e-16 ***
## V4           -0.5731     0.1406  -4.076 5.34e-05 ***
## V5           -2.2136     0.1080 -20.491  < 2e-16 ***
## V6            1.8948     0.1603  11.822  < 2e-16 ***
## V7            1.6712     0.1334  12.526  < 2e-16 ***
## V8           -1.7447     0.2503  -6.971 1.02e-11 ***
## V9           -0.3703     0.1528  -2.423   0.0158 *
## V10           3.2300     0.1573  20.530  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.473 on 490 degrees of freedom
## Multiple R-squared:   0.74,  Adjusted R-squared:  0.7352
## F-statistic: 154.9 on 9 and 490 DF,  p-value: < 2.2e-16
```

```
par(mfrow=c(2,2))
plot(mod1)
```




```
#so first of all the residuals vs fitted indicates strong linearity since we dont see any sort of pattern mu
st be a linear function

#seems very linear but it also seems like there are outliers (point 315)
```

I tried out adding interaction terms and removing high leverage points and outliers.

It did turn out to have an R^2 over 99 but this will happen if we just keep adding more predictors i.e. interactions terms.

In addition, this is overfitting so lets try some unsuperivised methods.

The code below show that I used kmeans to try unsupervised clustering. I chose two clusters since when we plot out the correlation pairs plot using R. The variable V7 seems to be clearly seperated into two groups. This indicated that I could probably try out some clustering methods such as kmeans or hiearchical clustering.

The results show that there is some underlying classification with this dataset, and indeed it was two clusters. Therefore I seperated the dataset into two by the cluster labels, and ran a linear model on top of the two datasets. This gave us two models with over 99 in R^2 and adjusted R^2.

```
x <- subset(df, select = -c(y))
kmeansasim <- kmeans(scale(x), 2)

withclusts <- cbind(df,kmeansasim$cluster)

clust1 <- subset(withclusts, kmeansasim$cluster =="1")
clust2 <- subset(withclusts, kmeansasim$cluster =="2")

df1 <-subset(clust1, select = -c(11))
df2 <-subset(clust2, select = -c(11))

mod1 <- lm(y~., data =df1)
summary(mod1)
```

```
##
## Call:
## lm(formula = y ~ ., data = df1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.34713 -0.58880  0.04983  0.60447  2.98516
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.82157    1.28843   7.623 3.41e-12 ***
## V2          -2.61228    0.06519 -40.069  < 2e-16 ***
## V3           2.08549    0.03614  57.701  < 2e-16 ***
## V4           1.95682    0.02959  66.128  < 2e-16 ***
## V5          -2.11834    0.02889 -73.328  < 2e-16 ***
## V6           1.21793    0.03221  37.815  < 2e-16 ***
## V7           1.90442    0.03538  53.830  < 2e-16 ***
## V8          -2.69549    0.06807 -39.597  < 2e-16 ***
## V9           2.75288    0.04912  56.041  < 2e-16 ***
## V10          2.13157    0.03775  56.462  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9582 on 140 degrees of freedom
## Multiple R-squared:  0.9962, Adjusted R-squared:  0.996
## F-statistic:  4077 on 9 and 140 DF,  p-value: < 2.2e-16
```

```
mod2 <- lm(y~., data =df2)
summary(mod2)
```

```
##
## Call:
## lm(formula = y ~ ., data = df2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.99101 -0.59469  0.03455  0.60554  2.38121
##
## Coefficients:
##             Estimate Std. Error  t value Pr(>|t|)
## (Intercept)  2.25572    1.18910    1.897   0.0587 .
## V2           2.50714    0.02936   85.379  < 2e-16 ***
## V3           0.18201    0.02255    8.070 1.23e-14 ***
## V4          -2.31952    0.02747  -84.443  < 2e-16 ***
## V5          -2.09769    0.01762 -119.083  < 2e-16 ***
## V6           1.64942    0.02860   57.667  < 2e-16 ***
## V7           1.00835    0.03564   28.294  < 2e-16 ***
## V8          -1.96402    0.03863  -50.835  < 2e-16 ***
## V9          -2.19966    0.03172  -69.348  < 2e-16 ***
## V10          2.36347    0.02641   89.506  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9851 on 340 degrees of freedom
## Multiple R-squared:  0.9952, Adjusted R-squared:  0.9951
## F-statistic:  7908 on 9 and 340 DF,  p-value: < 2.2e-16
```

Q5

on picture