Statistical Learning HW 9 - Unsupervised Learning

Christian Conroy

April 29, 2019

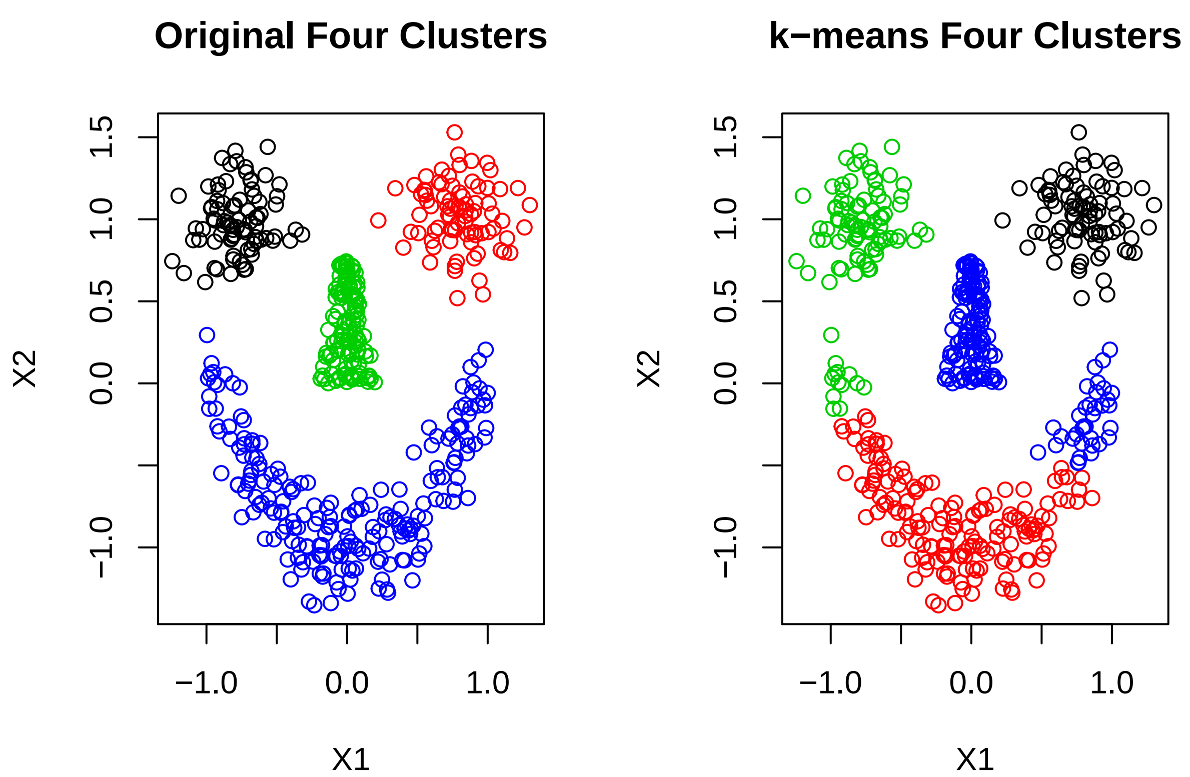
3 points # Extra 67 (3 points)

Make 500 smiley data points with sd1 = sd2 = 0.2.

```
set.seed(1)
smiley <- mlbench.smiley(n=500, sd1 = 0.2, sd2 = 0.2)
```

(a) Demonstrate with a colored plot that k-means with four clusters is incapable of recovering the four original clusters exactly. Do another run of k-means and use a confusion matrix to show that the four original clusters are not recovered exactly.

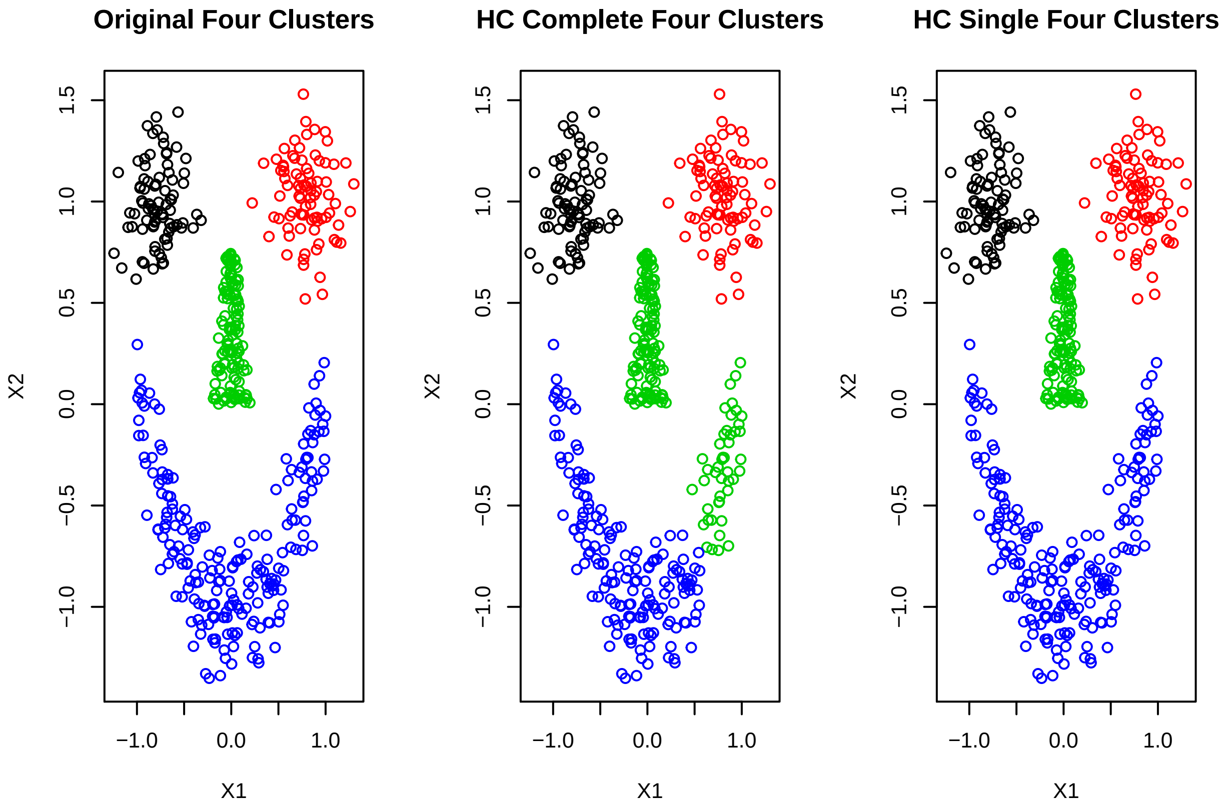
```
# Run K Means
set.seed(1)
km.out <- kmeans(smiley$x,4,nstart=15)

# Set plots on same page
par(mfrow = c(1,2))
# Plot the original clusters
plot(smiley$x[,1],smiley$x[,2], col = smiley$classes, main = "Original Four Clusters", xlab = "X1", ylab
# Plot New Clusters
plot(smiley$x[,1],smiley$x[,2], col = km.out$cluster, main = "k-means Four Clusters", xlab = "X1", ylab
```



(b) Try to use hierarchical clustering with a suitable choice of linkage to recover the four clusters. Explain your choice of linkage. Use a confusion matrix to show whether this attempt is successful.

```
# Complete Linkage
clust.complete <- hclust(dist(smiley$x),method="complete")
clust.complete.cut <- cutree(clust.complete,4)

# Complete Linkage
clust.single <- hclust(dist(smiley$x),method="single")
clust.single.cut <- cutree(clust.single,4)
```

```
# Set plots on same page
par(mfrow = c(1,3))
# Plot the original clusters
plot(smiley$x[,1],smiley$x[,2], col = smiley$classes, main = "Original Four Clusters", xlab = "X1", ylal
# Plot HC with Complete Linkage
plot(smiley$x[,1],smiley$x[,2], col = clust.complete.cut, main = "HC Complete Four Clusters", xlab = "X
# Plot HC with Single Linkage
plot(smiley$x[,1],smiley$x[,2], col = clust.single.cut, main = "HC Single Four Clusters", xlab = "X1", 
```




```
# For single linkage
confusionMatrix(smiley$classes, clust.single.cut)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2  3  4
##          1 83  0  0  0
```

```
##          2   0  83   0   0
##          3   0   0 125   0
##          4   0   0   0 209
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.993, 1)
##     No Information Rate : 0.418
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4
## Sensitivity             1.000    1.000     1.00    1.000
## Specificity             1.000    1.000     1.00    1.000
## Pos Pred Value          1.000    1.000     1.00    1.000
## Neg Pred Value          1.000    1.000     1.00    1.000
## Prevalence              0.166    0.166     0.25    0.418
## Detection Rate          0.166    0.166     0.25    0.418
## Detection Prevalence    0.166    0.166     0.25    0.418
## Balanced Accuracy       1.000    1.000     1.00    1.000
```

```r
# For complete linkage
confusionMatrix(smiley$classes, clust.complete.cut)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2   3   4
##          1  83   0   0   0
##          2   0  83   0   0
##          3   0   0 125   0
##          4   0   0  45 164
##
## Overall Statistics
##
##                Accuracy : 0.91
##                  95% CI : (0.881, 0.934)
##     No Information Rate : 0.34
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.875
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4
## Sensitivity             1.000    1.000    0.735    1.000
## Specificity             1.000    1.000    1.000    0.866
## Pos Pred Value          1.000    1.000    1.000    0.785
## Neg Pred Value          1.000    1.000    0.880    1.000
```

```
## Prevalence              0.166    0.166    0.340    0.328
## Detection Rate          0.166    0.166    0.250    0.328
## Detection Prevalence    0.166    0.166    0.250    0.418
## Balanced Accuracy       1.000    1.000    0.868    0.933
```

Hierachical Clustering with single linkage appears to best replicate the original clusters. Because single linkage tends to yield trailing clusters as opposed to complete linkage which yields more balanced attractive, clusters, single linkage here is better able to capture the smile part in the scatter plot.

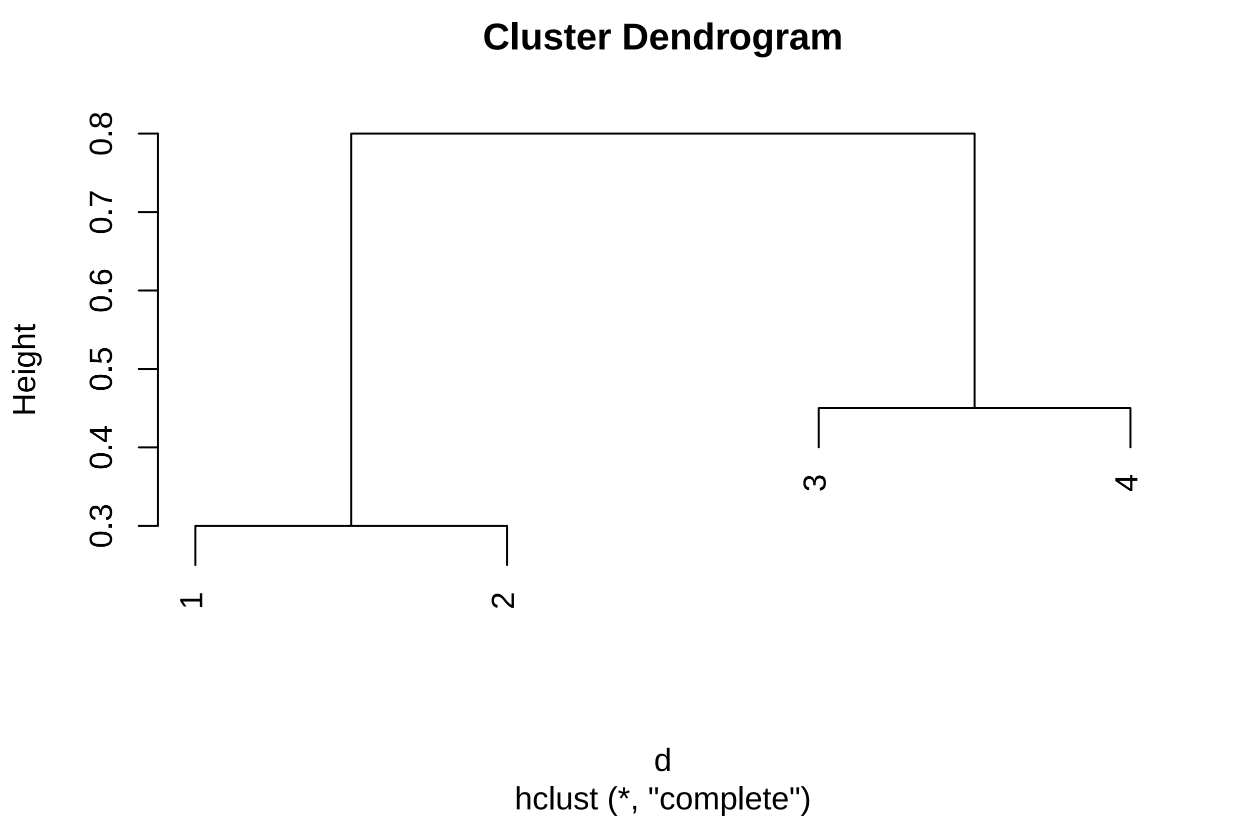
Book 2 (3 points)

Suppose that we have four observations, for which we compute a dissimilarity matrix.

For instance, the dissimilarity between the first and second observations is 0.3, and the dissimilarity between the second and fourth observations is 0.8.

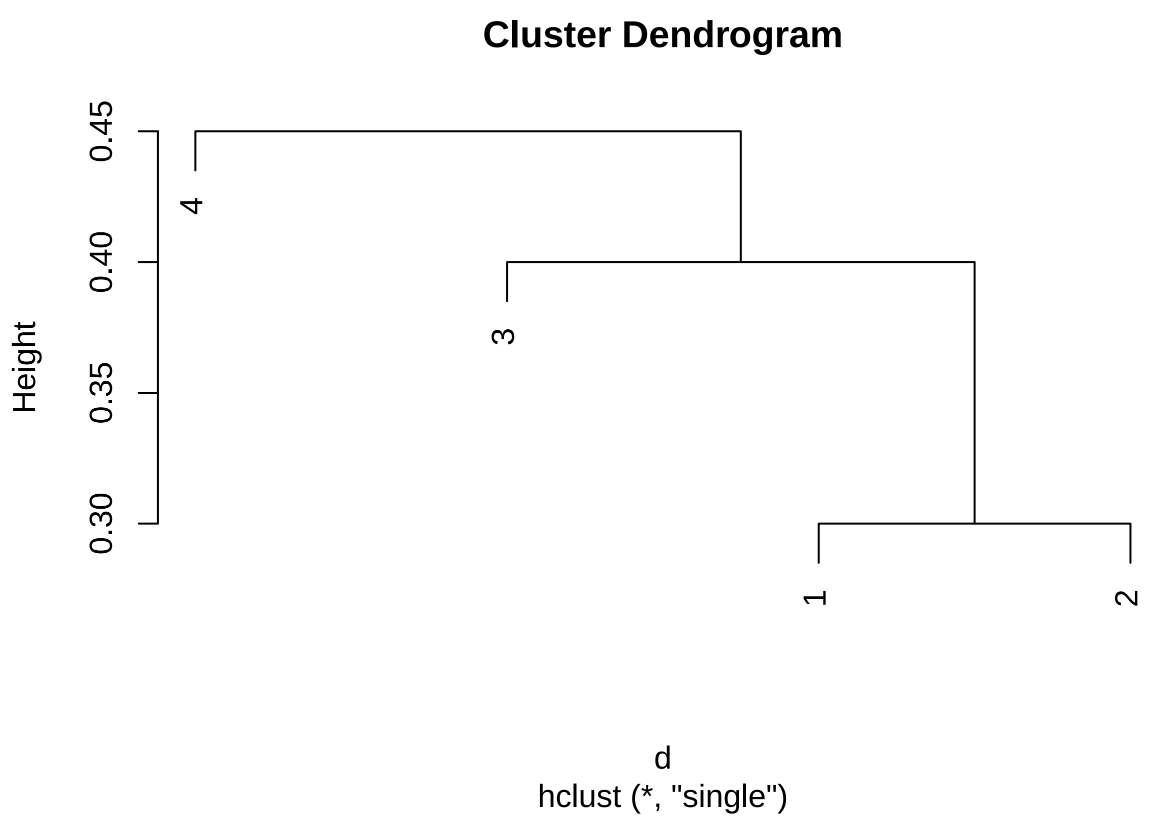
(a) On the basis of this dissimilarity matrix, sketch the dendrogram that results from hierarchically clustering these four observations using complete linkage. Be sure to indicate on the plot the height at which each fusion occurs, as well as the observations corresponding to each leaf in the dendrogram.

```r
d <- as.dist(matrix(c(0, 0.3, 0.4, 0.7,
                      0.3, 0, 0.5, 0.8,
                      0.4, 0.5, 0.0, 0.45,
                      0.7, 0.8, 0.45, 0.0), nrow = 4))
plot(hclust(d, method = "complete"))
```



(b) Repeat (a), this time using single linkage clustering.

```r
plot(hclust(d, method = "single"))
```



(c) Suppose that we cut the dendogram obtained in (a) such that two clusters result. Which observations are in each cluster?

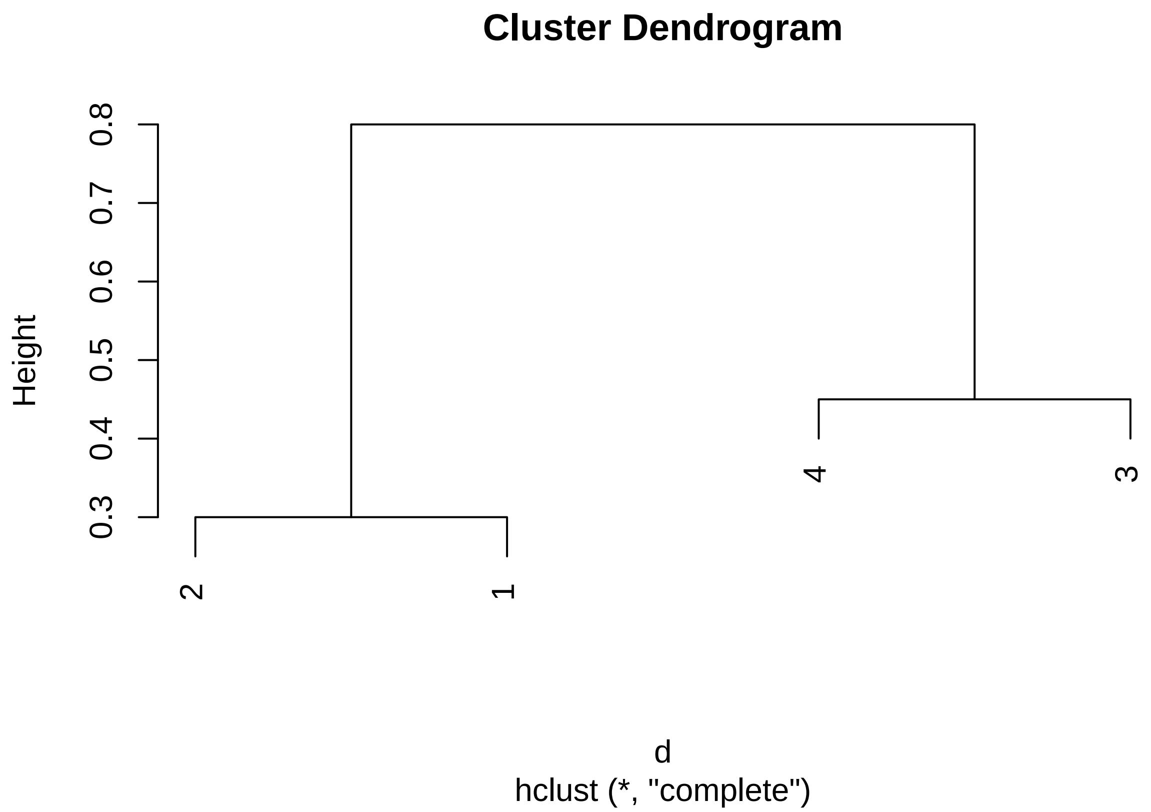
1 and 2 in cluster 1 and 3 and 4 in cluster 2

(d) Suppose that we cut the dendogram obtained in (b) such that two clusters result. Which observations are in each cluster?

1, 2, and 3 in cluster 1 and 4 in cluster 2

(e) It is mentioned in the chapter that at each fusion in the dendrogram, the position of the two clusters being fused can be swapped without changing the meaning of the dendrogram. Draw a dendrogram that is equivalent to the dendrogram in (a), for which two or more of the leaves are repositioned, but for which the meaning of the dendrogram is the same.

```r
plot(hclust(d, method = "complete"), labels = c(2,1,4,3))
```



-

Extra 72 (3 points)

Consider the concrete strength data from problem 37. There are eight numerical predictors and one numerical response. Load the data and split them into a training and test set (70% / 30%). We want to predict strength.

```
# Load in Data
concrete <- read_excel("Concrete_Data.xls")
# Clean up names
colnames(concrete) <- c("cementkg", "blustfur", "flyash", "water", "superplas", "courseagg", "fineagg",

# Create Train and Test
train <- sample(nrow(concrete),(nrow(concrete) * .70), replace = FALSE)
concrete_train <- concrete[train,]
concrete_test <- concrete[-train,]
```

a) Compute the principal components of the matrix of predictors for the training set. Fit a linear model to predict strength from the first principal component (simple regression).

```
# Create Matrix of Predictors
x <- model.matrix(CCS ~ .-1, concrete_train)

# Compute PC
pr.out <- prcomp(x, scale=TRUE)
```

```r
# Fit LM fromm first PC
train.data <- data.frame(CCS = concrete_train$CCS, pr.out$x)
train.data <- train.data[,1:2]
reg <- lm(CCS ~ ., data = train.data)
summary(reg)
```

```
##
## Call:
## lm(formula = CCS ~ ., data = train.data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -31.78 -12.50  -1.38  10.24  45.40
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   36.184      0.625   57.89   <2e-16 ***
## PC1            0.812      0.413    1.97     0.05 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.8 on 719 degrees of freedom
## Multiple R-squared:  0.00534,    Adjusted R-squared:  0.00396
## F-statistic: 3.86 on 1 and 719 DF,  p-value: 0.0498
```

b) Make predictions for the test set, using the same model. You have to use the loading vectors which were found from the principal component analysis of the training data.

```r
test.data <- predict(pr.out, newdata = concrete_test)
test.data <- data.frame(CCS = concrete_test$CCS, test.data)
test.data <- test.data[,1:2]

predict(reg, test.data)
```

```
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
## 35.7 32.8 32.4 32.5 33.9 35.4 34.0 33.3 33.2 35.2 33.4 33.0 32.6 33.5 33.4
##   16   17   18   19   20   21   22   23   24   25   26   27   28   29   30
## 32.3 35.7 33.8 35.5 35.5 33.6 33.2 34.0 36.4 36.1 38.9 37.7 37.3 36.4 36.4
##   31   32   33   34   35   36   37   38   39   40   41   42   43   44   45
## 37.0 36.4 36.1 36.2 39.2 37.3 36.4 36.1 36.1 35.2 37.4 37.6 38.8 37.3 36.3
##   46   47   48   49   50   51   52   53   54   55   56   57   58   59   60
## 36.9 36.3 35.1 37.5 37.3 35.2 38.9 37.4 36.0 36.1 36.8 36.5 36.2 37.7 36.2
##   61   62   63   64   65   66   67   68   69   70   71   72   73   74   75
## 36.0 37.6 37.6 36.7 38.2 36.1 36.0 35.9 36.3 36.8 36.6 36.8 37.9 37.7 36.4
##   76   77   78   79   80   81   82   83   84   85   86   87   88   89   90
## 36.2 36.5 37.1 37.1 37.7 37.1 38.3 38.2 38.1 37.8 38.4 38.1 38.1 38.0 37.9
##   91   92   93   94   95   96   97   98   99  100  101  102  103  104  105
## 37.3 37.2 36.9 36.7 37.3 37.7 37.7 37.6 35.0 34.9 34.8 37.1 36.7 34.6 37.5
##  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120
## 36.9 35.6 36.2 35.8 36.3 37.8 37.9 36.4 37.8 37.3 37.9 37.5 36.6 37.5 36.3
##  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135
## 37.2 36.1 37.3 37.2 37.7 36.4 36.4 37.4 37.2 37.0 37.0 37.0 37.0 36.7 37.9
##  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150
## 37.9 37.9 35.5 35.6 35.4 36.9 37.1 36.9 36.1 36.2 36.0 36.9 37.0 36.8 37.8
```

```
##  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165
## 37.8 37.7 35.2 34.9 35.2 36.0 36.3 36.0 36.2 35.7 35.8 35.2 34.7 35.0 34.7
##  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180
## 35.4 35.7 35.6 35.7 35.4 34.9 35.0 35.4 35.4 34.8 34.2 36.0 35.9 34.7 35.8
##  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195
## 35.9 35.2 35.7 36.0 36.0 35.9 35.9 35.8 35.6 35.5 34.4 34.3 35.1 35.4 34.6
##  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210
## 35.7 36.1 35.1 35.7 36.0 35.6 35.2 34.3 35.4 34.6 34.1 35.4 35.4 35.8 35.1
##  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225
## 36.1 33.8 35.4 35.7 35.7 35.6 35.6 35.5 35.4 34.6 35.1 34.9 34.2 35.1 34.7
##  226  227  228  229  230  231  232  233  234  235  236  237  238  239  240
## 35.5 35.1 34.5 35.6 35.6 35.7 35.7 35.4 35.6 35.0 34.3 34.0 35.2 35.6 35.5
##  241  242  243  244  245  246  247  248  249  250  251  252  253  254  255
## 35.4 35.6 35.8 34.7 33.9 35.4 35.4 35.4 35.2 37.3 36.9 36.5 37.0 35.5 36.9
##  256  257  258  259  260  261  262  263  264  265  266  267  268  269  270
## 36.7 36.0 35.5 37.9 36.2 36.3 36.8 35.3 36.2 37.0 36.3 36.2 37.6 36.5 37.3
##  271  272  273  274  275  276  277  278  279  280  281  282  283  284  285
## 36.5 35.7 36.1 36.0 35.9 36.0 36.0 35.7 36.1 34.6 35.4 36.8 35.8 36.4 37.6
##  286  287  288  289  290  291  292  293  294  295  296  297  298  299  300
## 36.8 36.0 35.8 37.4 36.6 38.1 35.7 37.1 36.5 37.2 34.9 36.5 37.7 35.4 37.1
##  301  302  303  304  305  306  307  308  309
## 36.1 36.7 36.1 36.2 37.2 34.8 36.2 36.0 36.8
```

Book 9 (5 points)

Consider the USArrests data. We will now perform hierarchical clustering on the states.

```r
data('USArrests')
head(USArrests)
```

```
##            Murder Assault UrbanPop Rape
## Alabama      13.2     236       58 21.2
## Alaska       10.0     263       48 44.5
## Arizona       8.1     294       80 31.0
## Arkansas      8.8     190       50 19.5
## California    9.0     276       91 40.6
## Colorado      7.9     204       78 38.7
```

(a) Using hierarchical clustering with complete linkage and Euclidean distance, cluster the states.

```r
x <- dist(USArrests)
clust.complete.euc <- hclust(x, method = "complete")
```

(b) Cut the dendrogram at a height that results in three distinct clusters. Which states belong to which clusters?

```r
# Complete Linkage
clust.complete.euc.cut <- cutree(clust.complete.euc,3)
clust.complete.euc.cut
```

```
##        Alabama         Alaska        Arizona       Arkansas     California
##              1              1              1              2              1
##       Colorado    Connecticut       Delaware        Florida        Georgia
##              2              3              1              1              2
##         Hawaii          Idaho       Illinois        Indiana           Iowa
```

```
##             3             3             1             3             3
##        Kansas      Kentucky     Louisiana         Maine      Maryland
##             3             3             1             3             1
##  Massachusetts      Michigan     Minnesota   Mississippi      Missouri
##             2             1             3             1             2
##       Montana      Nebraska        Nevada New Hampshire    New Jersey
##             3             3             1             3             2
##    New Mexico      New York North Carolina  North Dakota          Ohio
##             1             1             1             3             3
##      Oklahoma        Oregon  Pennsylvania  Rhode Island South Carolina
##             2             2             3             2             1
##  South Dakota     Tennessee         Texas          Utah       Vermont
##             3             2             2             3             3
##      Virginia    Washington West Virginia     Wisconsin       Wyoming
##             2             2             3             3             2
```

(c) Hierarchically cluster the states using complete linkage and Euclidean distance, after scaling the variables to have standard deviation one.

```
scaleddata <- scale(USArrests)
sx <- dist(scaleddata)
sclust.complete.euc <- hclust(sx, method = "complete")
sclust.complete.euc.cut <- cutree(sclust.complete.euc,3)
sclust.complete.euc.cut
```

```
##       Alabama        Alaska       Arizona      Arkansas    California
##             1             1             2             3             2
##      Colorado   Connecticut      Delaware       Florida       Georgia
##             2             3             3             2             1
##        Hawaii         Idaho      Illinois       Indiana          Iowa
##             3             3             2             3             3
##        Kansas      Kentucky     Louisiana         Maine      Maryland
##             3             3             1             3             2
##  Massachusetts      Michigan     Minnesota   Mississippi      Missouri
##             3             2             3             1             3
##       Montana      Nebraska        Nevada New Hampshire    New Jersey
##             3             3             2             3             3
##    New Mexico      New York North Carolina  North Dakota          Ohio
##             2             2             1             3             3
##      Oklahoma        Oregon  Pennsylvania  Rhode Island South Carolina
##             3             3             3             3             1
##  South Dakota     Tennessee         Texas          Utah       Vermont
##             3             1             2             3             3
##      Virginia    Washington West Virginia     Wisconsin       Wyoming
##             3             3             3             3             3
```

(d) What effect does scaling the variables have on the hierarchical clustering obtained? In your opinion, should the variables be scaled before the inter-observation dissimilarities are computed? Provide a justification for your answer.

Because statistics for each category of the US Arrests data are reported differently (percent out of an amount of the population versus the number of recorded incidents), the units are different and one variable therefore may have a disproportionate affect on the inter-state dissimilarities, which in turn influences the clustering. Scaling before the dissimilarities are computed is usually best because it gives equal importance to the hierarchical clustering performed. However, this is not always the case as it may give a variable a much greater or smaller effect on the inter-observation dissimilarities obtained. It therefore depends on the application.

Extra 69 (5 points)

In this problem, you will use k-means clustering for the smiley data, for different values of sd = sd1 = sd2. Use 500 points and four clusters throughout.

a) Demonstrate that for small values of sd k-means clustering recoversthe four clusters in the data reasonably well. Use confusion matrices to show this.

```
sdops <- c(0.001, 0.01, 0.1, 1)

kmsds <- function(x) {
  # Generate Data
  set.seed(1)
  smiley <- mlbench.smiley(n=500, sd1 = x, sd2 = x)
  # Run K Means
  set.seed(1)
  km.out <- kmeans(smiley$x,4)
  return(confusionMatrix(smiley$classes, km.out$cluster))
}

sdtest <- lapply(sdops, kmsds)

sdtest
```

```
## [[1]]
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2   3   4
##          1   0  83   0   0
##          2  83   0   0   0
##          3   0 119   0   6
##          4   0   1 123  85
##
## Overall Statistics
##
##                Accuracy : 0.17
##                  95% CI : (0.138, 0.206)
##     No Information Rate : 0.406
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : -0.081
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4
## Sensitivity             0.000    0.000    0.000    0.934
## Specificity             0.801    0.721    0.668    0.697
## Pos Pred Value          0.000    0.000    0.000    0.407
## Neg Pred Value          0.801    0.513    0.672    0.979
## Prevalence              0.166    0.406    0.246    0.182
## Detection Rate          0.000    0.000    0.000    0.170
## Detection Prevalence    0.166    0.166    0.250    0.418
## Balanced Accuracy       0.400    0.360    0.334    0.815
```

```
##
## [[2]]
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2   3   4
##          1   0  83   0   0
##          2  83   0   0   0
##          3   0 119   0   6
##          4   0   2 122  85
##
## Overall Statistics
##
##                Accuracy : 0.17
##                  95% CI : (0.138, 0.206)
##     No Information Rate : 0.408
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : -0.081
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4
## Sensitivity             0.000    0.000    0.000    0.934
## Specificity             0.801    0.720    0.669    0.697
## Pos Pred Value          0.000    0.000    0.000    0.407
## Neg Pred Value          0.801    0.511    0.675    0.979
## Prevalence              0.166    0.408    0.244    0.182
## Detection Rate          0.000    0.000    0.000    0.170
## Detection Prevalence    0.166    0.166    0.250    0.418
## Balanced Accuracy       0.400    0.360    0.335    0.815
##
## [[3]]
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2   3   4
##          1   0  83   0   0
##          2  83   0   0   0
##          3   0 119   0   6
##          4   0   7 118  84
##
## Overall Statistics
##
##                Accuracy : 0.168
##                  95% CI : (0.136, 0.204)
##     No Information Rate : 0.418
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : -0.082
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

```
##
##                      Class: 1 Class: 2 Class: 3 Class: 4
## Sensitivity             0.000    0.000    0.000    0.933
## Specificity             0.801    0.715    0.673    0.695
## Pos Pred Value          0.000    0.000    0.000    0.402
## Neg Pred Value          0.801    0.499    0.685    0.979
## Prevalence              0.166    0.418    0.236    0.180
## Detection Rate          0.000    0.000    0.000    0.168
## Detection Prevalence    0.166    0.166    0.250    0.418
## Balanced Accuracy       0.400    0.357    0.336    0.814
##
## [[4]]
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2   3   4
##          1  35  41   0   7
##          2  18  11   4  50
##          3   0 125   0   0
##          4   5  66 114  24
##
## Overall Statistics
##
##                Accuracy : 0.14
##                  95% CI : (0.111, 0.174)
##     No Information Rate : 0.486
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : -0.112
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4
## Sensitivity             0.603   0.0453    0.000    0.296
## Specificity             0.891   0.7198    0.673    0.558
## Pos Pred Value          0.422   0.1325    0.000    0.115
## Neg Pred Value          0.945   0.4436    0.685    0.804
## Prevalence              0.116   0.4860    0.236    0.162
## Detection Rate          0.070   0.0220    0.000    0.048
## Detection Prevalence    0.166   0.1660    0.250    0.418
## Balanced Accuracy       0.747   0.3826    0.336    0.427
```

The accuracy as reported by the Confusion Matices decreases as we increase the value of sd, proving that lower values of sd are better at recovering the original clusters.

b) Show that if sd becomes larger, the four clusters are no longer recovered well. Find an approximate value of sd for which this change occurs (two decimal digits is enough), and explain how k-means clustering behaves for larger values of sd, using colored plots and two different examples.

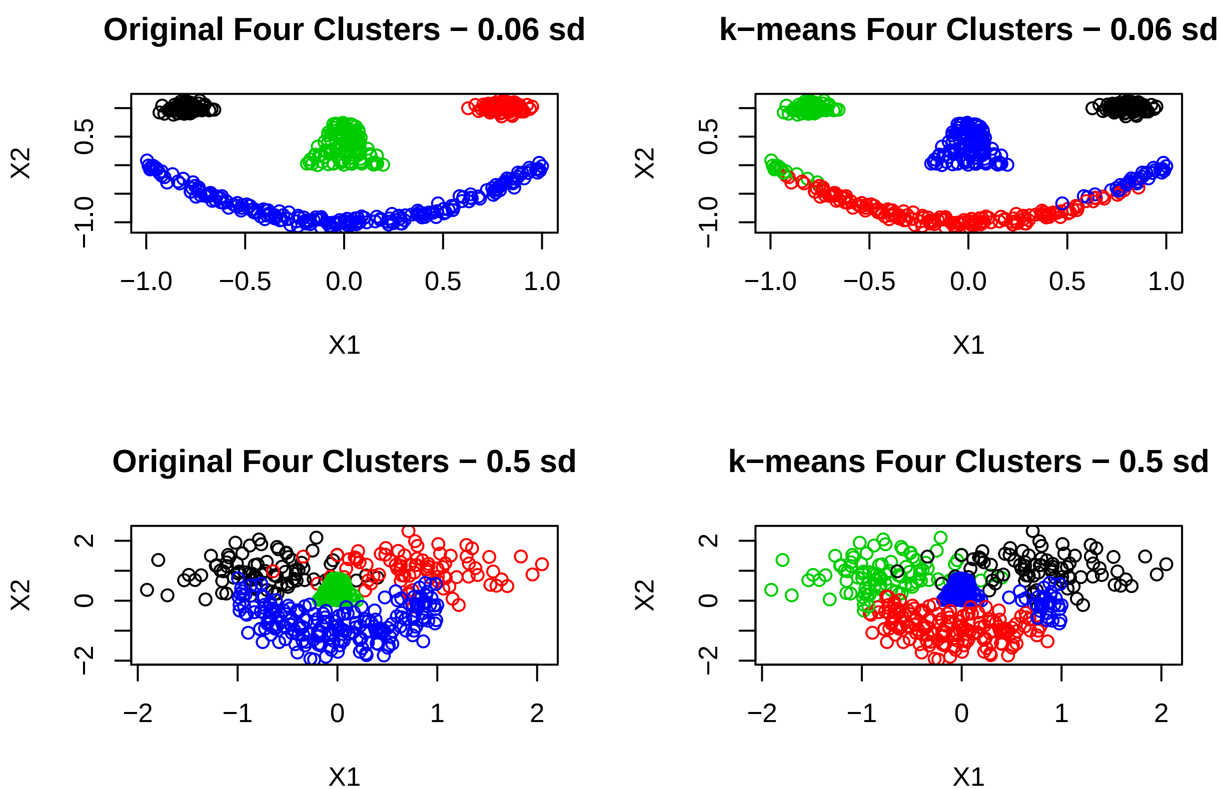
```
sdops <- c(0.01, 0.06, 0.07)

kmsds <- function(s) {
  # Generate Data
  set.seed(1)
```

```r
  smiley <- mlbench.smiley(n=500, sd1 = s, sd2 = s)
  # Run K Means
  set.seed(1)
  km.out <- kmeans(smiley$x,4)
  return(confusionMatrix(smiley$classes, km.out$cluster))
}

sdtest <- lapply(sdops, kmsds)

sdtest
```

```
## [[1]]
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2   3   4
##          1   0  83   0   0
##          2  83   0   0   0
##          3   0 119   0   6
##          4   0   2 122  85
##
## Overall Statistics
##
##                Accuracy : 0.17
##                  95% CI : (0.138, 0.206)
##     No Information Rate : 0.408
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : -0.081
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4
## Sensitivity             0.000    0.000    0.000    0.934
## Specificity             0.801    0.720    0.669    0.697
## Pos Pred Value          0.000    0.000    0.000    0.407
## Neg Pred Value          0.801    0.511    0.675    0.979
## Prevalence              0.166    0.408    0.244    0.182
## Detection Rate          0.000    0.000    0.000    0.170
## Detection Prevalence    0.166    0.166    0.250    0.418
## Balanced Accuracy       0.400    0.360    0.335    0.815
##
## [[2]]
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2   3   4
##          1   0  83   0   0
##          2  83   0   0   0
##          3   0 119   0   6
##          4   0   4 120  85
##
## Overall Statistics
```

```
##
##                Accuracy : 0.17
##                  95% CI : (0.138, 0.206)
##     No Information Rate : 0.412
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : -0.081
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4
## Sensitivity             0.000    0.000    0.000    0.934
## Specificity             0.801    0.718    0.671    0.697
## Pos Pred Value          0.000    0.000    0.000    0.407
## Neg Pred Value          0.801    0.506    0.680    0.979
## Prevalence              0.166    0.412    0.240    0.182
## Detection Rate          0.000    0.000    0.000    0.170
## Detection Prevalence    0.166    0.166    0.250    0.418
## Balanced Accuracy       0.400    0.359    0.336    0.815
##
## [[3]]
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2   3   4
##          1   0  83   0   0
##          2  83   0   0   0
##          3   0 119   0   6
##          4   0   6 119  84
##
## Overall Statistics
##
##                Accuracy : 0.168
##                  95% CI : (0.136, 0.204)
##     No Information Rate : 0.416
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : -0.082
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4
## Sensitivity             0.000    0.000    0.000    0.933
## Specificity             0.801    0.716    0.672    0.695
## Pos Pred Value          0.000    0.000    0.000    0.402
## Neg Pred Value          0.801    0.501    0.683    0.979
## Prevalence              0.166    0.416    0.238    0.180
## Detection Rate          0.000    0.000    0.000    0.168
## Detection Prevalence    0.166    0.166    0.250    0.418
## Balanced Accuracy       0.400    0.358    0.336    0.814
```

```r
par(mfrow = c(2,2))
# Generate Data at sd of 0.06
set.seed(1)
smiley <- mlbench.smiley(n=500, sd1 = 0.06, sd2 = 0.06)
# Plot the original clusters at sd of 0.06
plot(smiley$x[,1],smiley$x[,2], col = smiley$classes, main = "Original Four Clusters - 0.06 sd", xlab =
# Plot New Clusters at sd of 0.06
plot(smiley$x[,1],smiley$x[,2], col = km.out$cluster, main = "k-means Four Clusters - 0.06 sd", xlab =

 # Generate Data at sd of 0.07
set.seed(1)
smiley <- mlbench.smiley(n=500, sd1 = 0.5, sd2 = 0.5)
# Plot the original clusters at sd of 0.07
plot(smiley$x[,1],smiley$x[,2], col = smiley$classes, main = "Original Four Clusters - 0.5 sd", xlab =
# Plot New Clusters at sd of 0.07
plot(smiley$x[,1],smiley$x[,2], col = km.out$cluster, main = "k-means Four Clusters - 0.5 sd", xlab = "
```



We start to see the accuracy decline in moving from an sd of 0.06 to an sd of 0.07. To make the point that the accuracy declines for larger sds, we use plots to exagerrate this trend.

Extra 71 (5 points)

This problem uses the MNIST image classification data, available as mnist_all.RData that were used earlier. We use the training data only for all digits. Extract the training data and place them in suitable data frames.

```r
mnist <-load('mnist_all.RData')

mnist_train <- data.frame(train$x, train$y)
```

a) Apply k-means clustering with two clusters. Can you tell which digits tend to be clustered together?

```r
km.out <- kmeans(mnist_train$train.y,2,nstart=25)
mnist_train$cluster <- km.out$cluster

clustertabs <- mnist_train %>%
  group_by(train.y) %>%
dplyr::summarize(Cluster = mean(cluster))

clustertabs
```

```
## # A tibble: 10 x 2
##    train.y Cluster
##      <int>   <dbl>
##  1       0       1
##  2       1       1
##  3       2       1
##  4       3       1
##  5       4       1
##  6       5       2
##  7       6       2
##  8       7       2
##  9       8       2
## 10       9       2
```

The first 5 digits chronologically were put in cluster 1 and the next 5 in cluster 2.

b) Apply k-means clustering with 10 clusters. How well do the cluster labels agree with the actual digits labels? Use a confusion matrix to answer this question.

```r
km.out <- kmeans(mnist_train$train.y,10,nstart=25)
mnist_train$cluster <- km.out$cluster

# Mismatch due to the existance of the digit 0 and cluster 10 - Eliminate those two levels and compare
mnist_train2 <- mnist_train[mnist_train$train.y != 0 | mnist_train$cluster != 10,]

# Confusion Matrix - Done manually because there is at least one number that is not predicted
table(factor(mnist_train2$cluster, levels=min(mnist_train2$train.y):max(mnist_train2$train.y)),
      factor(mnist_train2$train.y, levels=min(mnist_train2$train.y):max(mnist_train2$train.y)))
```

```
##
##        0    1    2    3    4    5    6    7    8    9
##   0    0    0    0    0    0    0    0    0    0    0
##   1    0    0    0    0    0    0    0 6265    0    0
##   2    0    0    0    0    0    0    0    0    0 5949
##   3 5923    0    0    0    0    0    0    0    0    0
##   4    0    0    0    0    0 5421    0    0    0    0
##   5    0    0    0 6131    0    0    0    0    0    0
##   6    0    0 5958    0    0    0    0    0    0    0
##   7    0    0    0    0    0    0    0    0 5851    0
##   8    0    0    0    0 5842    0    0    0    0    0
##   9    0 6742    0    0    0    0    0    0    0    0
```

```r
# Accuracy
sum(mnist_train2$cluster == mnist_train2$train.y) / nrow(mnist_train2)
```

```
## [1] 0
```

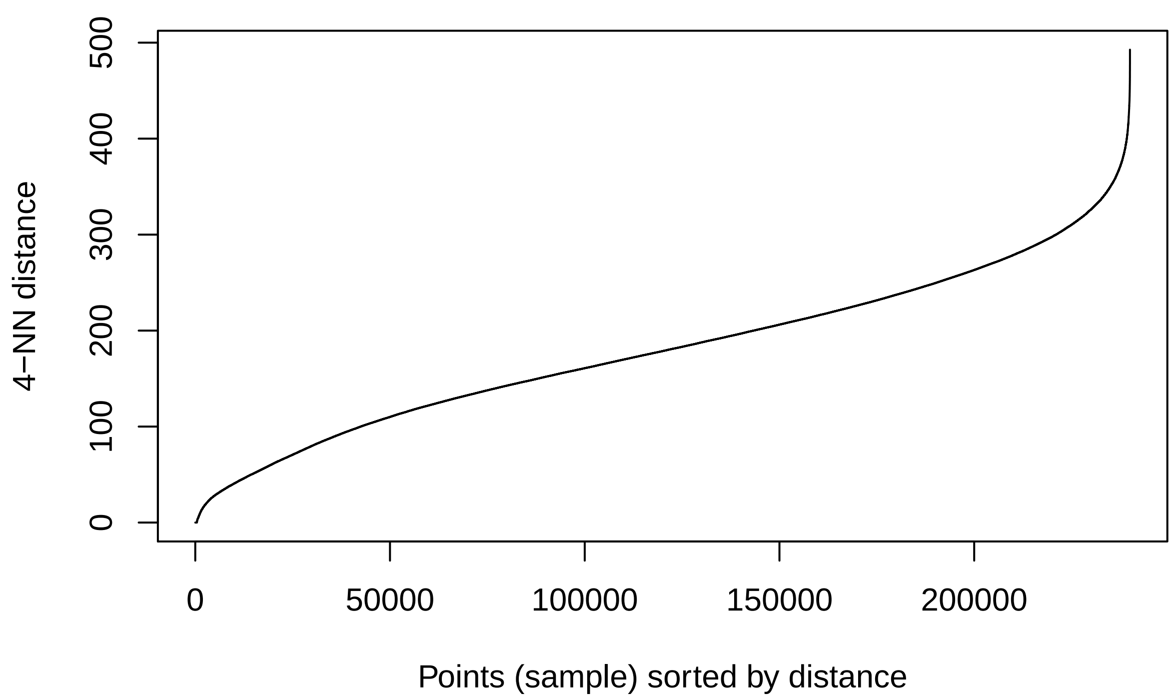
Accuracy is 0.0992, which is not great.

c) Apply dbscan clustering, with suitable choices of eps and minPtsobtained from a k-nearest neighbor plot. Justify your choices.Then determine how well the cluster labels agree with the actual digit labels, using a confusion matrix.

```r
# Downsample to decrease kNNdistplot runtime
downsamp <- seq(1, ncol(mnist_train), 16)
train2 <- mnist_train[ downsamp]
names(train2)
```

```
##  [1] "X1"      "X17"     "X33"     "X49"     "X65"     "X81"     "X97"
##  [8] "X113"    "X129"    "X145"    "X161"    "X177"    "X193"    "X209"
## [15] "X225"    "X241"    "X257"    "X273"    "X289"    "X305"    "X321"
## [22] "X337"    "X353"    "X369"    "X385"    "X401"    "X417"    "X433"
## [29] "X449"    "X465"    "X481"    "X497"    "X513"    "X529"    "X545"
## [36] "X561"    "X577"    "X593"    "X609"    "X625"    "X641"    "X657"
## [43] "X673"    "X689"    "X705"    "X721"    "X737"    "X753"    "X769"
## [50] "train.y"
```

```r
kNNdistplot(as.matrix(train2[,-50]), k=4)
```




```r
dbscan_clust <- dbscan(as.matrix(train2[,-50]), eps=300)
```

```r
# Mismatch due to the existance of the digit 0 and cluster 10 - Eliminate those two levels and compare
#mnist_train2 <- mnist_train[mnist_train$train.y != 0 | mnist_train$cluster != 10,]

# Confusion Matrix - Done manually because there is at least one number that is not predicted
table(factor(dbscan_clust$cluster, levels=min(train2$train.y):max(train2$train.y)),
      factor(train2$train.y, levels=min(train2$train.y):max(train2$train.y)))
```

```
##
##        0    1    2    3    4    5    6    7    8    9
##   0  372   11  829  488  206  399  200   98  539  149
##   1 5548 6731 5119 5640 5636 5016 5718 6162 5312 5800
##   2    0    0    0    0    0    0    0    5    0    0
##   3    0    0    5    0    0    0    0    0    0    0
##   4    3    0    0    0    0    1    0    0    0    0
##   5    0    0    0    0    0    4    0    0    0    0
##   6    0    0    0    3    0    1    0    0    0    0
##   7    0    0    5    0    0    0    0    0    0    0
##   8    0    0    0    0    0    0    0    0    0    0
##   9    0    0    0    0    0    0    0    0    0    0
```

```r
# Accuracy
sum(dbscan_clust$cluster == train2$train.y) / nrow(train2)
```

```
## [1] 0.118
```

The kink in the 4-NN distance plot appears to be located approximately at 300, so we use this for eps. We find an accuracy of 0.118.