HW Clustering

Kevin Toney

October 20, 2017

Problem 3: K-Means Clustering

```
#load the iris dataset
iris_copy <- iris[,c(1:4)]
k2 <- kmeans(iris_copy, 2)
k2$size
```

```
## [1] 97 53
```

```
#for k=2, the size of the clusters is 53 and 97.
k2clust <- k2$cluster
sil2 <- silhouette(k2$cluster, dist(iris_copy))
y2 <- summary(sil2)[[4]] #average silhouette width
#the f-measure is 77.6%

k3 <- kmeans(iris_copy, 3)
sil3 <- silhouette(k3$cluster, dist(iris_copy))
y3 <- summary(sil3)[[4]]
#for k=3, the size of the clusters is 62, 38, and 50.
#The f-measure is 88.4%

k4 <- kmeans(iris_copy, 4)
sil4 <- silhouette(k4$cluster, dist(iris_copy))
y4 <- summary(sil4)[[4]]
#For k=4, the size of the clusters is 27, 28, 50, and 45.
#The f-measure is 91.6%

kmeans(iris_copy, 5)
```

```
## K-means clustering with 5 clusters of sizes 62, 23, 8, 38, 19
##
## Cluster means:
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1     5.901613    2.748387     4.393548    1.433871
## 2     5.100000    3.513043     1.526087    0.273913
## 3     5.512500    4.000000     1.475000    0.275000
## 4     6.850000    3.073684     5.742105    2.071053
## 5     4.678947    3.084211     1.378947    0.200000
##
## Clustering vector:
##   [1] 2 5 5 5 2 3 5 2 5 5 3 2 5 5 3 3 3 2 3 2 2 2 5 2 2 5 2 2 2 5 5 2 3 3 5
##  [36] 5 2 2 5 2 2 5 5 2 2 5 2 5 2 2 1 1 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [71] 1 1 1 1 1 1 1 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 4 1 4 4 4
## [106] 4 1 4 4 4 4 4 4 1 1 4 4 4 4 1 4 1 4 1 4 4 1 1 4 4 4 4 4 1 4 4 4 4 1 4
## [141] 4 4 1 4 4 4 1 4 4 1
##
## Within cluster sum of squares by cluster:
```

```
## [1] 39.820968  2.094783  0.958750 23.879474  2.488421
##  (between_SS / total_SS =  89.8 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

```r
#For k=5, the size of the clusters is 38, 19, 62, 8, and 23.
#The F-measure is 89.8%

kmeans(iris_copy, 7)
```

```
## K-means clustering with 7 clusters of sizes 19, 19, 17, 28, 12, 33, 22
##
## Cluster means:
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1     6.036842    2.705263     5.000000   1.7789474
## 2     6.442105    2.978947     4.594737   1.4315789
## 3     5.370588    3.800000     1.517647   0.2764706
## 4     5.532143    2.635714     3.960714   1.2285714
## 5     7.475000    3.125000     6.300000   2.0500000
## 6     4.818182    3.236364     1.433333   0.2303030
## 7     6.568182    3.086364     5.536364   2.1636364
##
## Clustering vector:
##   [1] 6 6 6 6 6 3 6 6 6 6 3 6 6 6 3 3 3 6 3 3 3 3 6 6 6 6 6 3 6 6 6 3 3 3 6
##  [36] 6 3 6 6 6 6 6 6 6 3 6 3 6 3 6 2 2 2 4 2 4 2 4 2 4 4 4 4 2 4 2 4 4 1 4
##  [71] 1 4 1 2 2 2 2 2 2 4 4 4 4 1 4 2 2 2 4 4 4 2 4 4 4 4 4 2 4 4 7 1 5 7 7
## [106] 5 4 5 7 5 7 1 7 1 1 7 7 5 5 1 7 1 5 1 7 5 1 1 7 5 5 5 7 1 1 5 7 7 1 7
## [141] 7 7 1 7 7 7 1 7 7 1
##
## Within cluster sum of squares by cluster:
## [1] 4.125263 3.708421 2.630588 9.749286 4.655000 5.428485 4.315455
##  (between_SS / total_SS =  94.9 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

```r
#For k=7, the size of the clusters is 50, 22, 19, 19, 12, 21, and 7.
#The F-measure is 94.5%

kmeans(iris_copy, 9)
```

```
## K-means clustering with 9 clusters of sizes 14, 29, 7, 14, 10, 5, 22, 21, 28
##
## Cluster means:
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1     5.007143    3.442857     1.414286   0.2714286
## 2     6.196552    2.882759     5.182759   1.9344828
## 3     5.528571    4.042857     1.471429   0.2857143
## 4     4.778571    3.157143     1.507143   0.2000000
```

```
## 5     5.260000    3.630000     1.550000   0.2700000
## 6     4.400000    2.880000     1.280000   0.2000000
## 7     7.122727    3.113636     6.031818   2.1318182
## 8     6.423810    2.919048     4.604762   1.4380952
## 9     5.532143    2.635714     3.960714   1.2285714
##
## Clustering vector:
##   [1] 1 4 4 4 1 3 4 1 6 4 5 4 4 6 3 3 3 1 3 5 5 5 1 1 4 4 1 5 1 4 4 5 3 3 4
##  [36] 1 5 1 6 1 1 6 6 1 5 4 5 4 5 1 8 8 8 9 8 9 8 9 8 9 9 9 9 8 9 8 9 9 8 9
##  [71] 2 9 8 8 8 8 8 8 8 9 9 9 9 2 9 8 8 8 9 9 9 8 9 9 9 9 9 8 9 9 7 2 7 2 7
## [106] 7 9 7 7 7 2 2 7 2 2 2 2 7 7 2 7 2 7 2 7 7 2 2 2 7 7 7 2 2 2 7 2 2 2 7
## [141] 7 2 2 7 7 2 2 2 2 2
##
## Within cluster sum of squares by cluster:
## [1]  1.0892857  8.7379310  0.8342857  0.8271429  0.7710000  0.5560000
## [7] 11.5400000  4.6495238  9.7492857
##  (between_SS / total_SS =  94.3 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

```r
#For k=9, the size of the clusters is 23, 22, 28, 8, 19, 12, 19, 8, and 11.
#The F-measure is 95.4%

kmeans(iris_copy, 11)
```

```
## K-means clustering with 11 clusters of sizes 17, 7, 19, 5, 16, 19, 7, 5, 21, 12, 22
##
## Cluster means:
##    Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1      5.170588    3.552941     1.464706   0.2352941
## 2      5.528571    4.042857     1.471429   0.2857143
## 3      6.036842    2.705263     5.000000   1.7789474
## 4      5.000000    3.480000     1.740000   0.4200000
## 5      4.793750    3.181250     1.425000   0.2000000
## 6      6.442105    2.978947     4.594737   1.4315789
## 7      5.242857    2.371429     3.442857   1.0285714
## 8      4.400000    2.880000     1.280000   0.2000000
## 9      5.628571    2.723810     4.133333   1.2952381
## 10     7.475000    3.125000     6.300000   2.0500000
## 11     6.568182    3.086364     5.536364   2.1636364
##
## Clustering vector:
##   [1]  1  5  5  5  1  2  5  1  8  5  1  5  5  8  2  2  2  1  2  1  1  1  5
##  [24]  4  4  5  4  1  1  5  5  1  2  2  5  5  1  1  8  1  1  8  8  4  4  5
##  [47]  1  5  1  5  6  6  6  9  6  9  6  7  6  9  7  9  9  6  9  6  9  9  3
##  [70]  9  3  9  3  6  6  6  6  6  6  7  7  7  9  3  9  6  6  6  9  9  9  6
##  [93]  9  7  9  9  9  6  7  9 11  3 10 11 11 10  9 10 11 10 11  3 11  3  3
## [116] 11 11 10 10  3 11  3 10  3 11 10  3  3 11 10 10 10 11  3  3 10 11 11
## [139]  3 11 11 11  3 11 11 11  3 11 11  3
##
## Within cluster sum of squares by cluster:
```

```
##  [1] 1.0552941 0.8342857 4.1252632 0.3880000 1.2037500 3.7084211 1.2628571
##  [8] 0.5560000 4.1771429 4.6550000 4.3154545
##  (between_SS / total_SS =  96.1 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

```
#For k=11, the size of the clusters is 3, 4, 12, 12, 11, 19, 17, 19, 12, 22, and 19.
#The F-measure is 96.3%

#The value of k that produces the highest F-score is k=11.
#I think it is interesting the value of k=5 produced a lower
#F-measure than most of the other values of k.
#That amount of centroids proved to be the least accurate.


###Cool automatic k-means clustering program

k <- c(2, 3, 4, 5, 7, 9, 11)
km.out <- list()
sil.out <- list()
x <- vector()
y <- vector()
for (i in k){
  set.seed(5)
  km.out[i] <- list(kmeans(iris_copy, centers=i))
  sil.out[i] <- list(silhouette(km.out[[i]]$cluster, dist(iris_copy)))


  x[i] <- i
  y[i] <- summary(sil.out[[i]])[[4]]
}

#only get the silhouette widths for the k values we are working with.
y <- y[!is.na(y)]
y <- y[y>0]
y
```

```
## [1] 0.6810462 0.5528190 0.4152074 0.3711254 0.3226230 0.3036873 0.2984184
```

```
#I think the silhouette widths are telling me that a cluster of 11
#most accurate size of clusters.
```

For k=2, the size of the clusters is 53 and 97. The f-measure is 77.6%.

For k=3, the size of the clusters is 62, 38, and 50. The f-measure is 88.4%.

For k=4, the size of the clusters is 27, 28, 50, and 45. The f-measure is 91.6%.

For k=5, the size of the clusters is 38, 19, 62, 8, and 23. The F-measure is 89.8%.

For k=7, the size of the clusters is 50, 22, 19, 19, 12, 21, and 7. The F-measure is 94.5%.

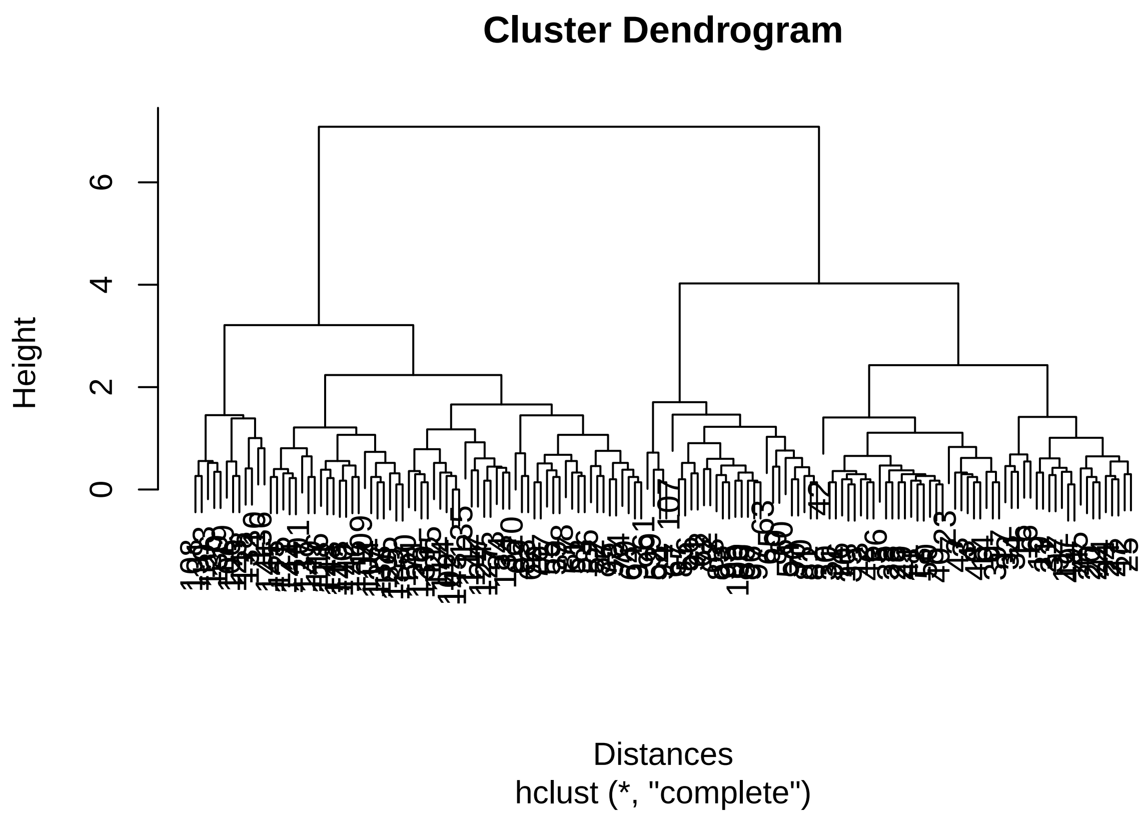
For k=9, the size of the clusters is 23, 22, 28, 8, 19, 12, 19, 8, and 11. The F-measure is 95.4%.

For k=11, the size of the clusters is 3, 4, 12, 12, 11, 19, 17, 19, 12, 22, and 19. The F-measure is 96.3%.

The value of k that produces the highest F-score is k=11. I think it is interesting the value of k=5 produced a lower F-measure than most of the other values of k. That amount of centroids proved to be the least accurate.

Problem 4: Hierarchical Agglomerative Clustering

```r
agglom <- hclust(dist(iris_copy))

#plot the dendogram
plot(agglom, xlab="Distances")
```




```r
#check the heights
agglom$height
```

```
##   [1] 0.0000000 0.1000000 0.1000000 0.1000000 0.1000000 0.1000000 0.1414214
##   [8] 0.1414214 0.1414214 0.1414214 0.1414214 0.1414214 0.1414214 0.1414214
##  [15] 0.1414214 0.1414214 0.1414214 0.1414214 0.1414214 0.1414214 0.1414214
##  [22] 0.1732051 0.1732051 0.1732051 0.1732051 0.1732051 0.1732051 0.2000000
##  [29] 0.2000000 0.2000000 0.2000000 0.2000000 0.2000000 0.2236068 0.2236068
##  [36] 0.2449490 0.2449490 0.2449490 0.2449490 0.2449490 0.2449490 0.2449490
##  [43] 0.2449490 0.2645751 0.2645751 0.2645751 0.2645751 0.2645751 0.2645751
##  [50] 0.2645751 0.2645751 0.2645751 0.2828427 0.2828427 0.3000000 0.3000000
##  [57] 0.3000000 0.3000000 0.3000000 0.3162278 0.3162278 0.3162278 0.3316625
##  [64] 0.3316625 0.3316625 0.3316625 0.3316625 0.3316625 0.3464102 0.3464102
##  [71] 0.3464102 0.3464102 0.3605551 0.3605551 0.3741657 0.3741657 0.3741657
##  [78] 0.3872983 0.3872983 0.3872983 0.4000000 0.4000000 0.4123106 0.4123106
```

```
##  [85] 0.4242641 0.4242641 0.4358899 0.4472136 0.4472136 0.4582576 0.4582576
##  [92] 0.4690416 0.4690416 0.4690416 0.5099020 0.5196152 0.5196152 0.5196152
##  [99] 0.5196152 0.5196152 0.5477226 0.5477226 0.5477226 0.5567764 0.5567764
## [106] 0.5567764 0.6000000 0.6082763 0.6082763 0.6164414 0.6164414 0.6480741
## [113] 0.6480741 0.6557439 0.6782330 0.6855655 0.7071068 0.7211103 0.7348469
## [120] 0.7549834 0.7615773 0.7874008 0.8062258 0.8062258 0.8306624 0.9055385
## [127] 0.9219544 1.0049876 1.0099505 1.0295630 1.0677078 1.0677078 1.1090537
## [134] 1.1747340 1.2124356 1.2247449 1.3892444 1.4071247 1.4177447 1.4491377
## [141] 1.4525839 1.4628739 1.6613248 1.7058722 2.2360680 2.4289916 3.2109189
## [148] 4.0249224 7.0851958
```

b

I would say an optimal threshold is 6 clusters. There is a large jump in the distance function starting at heights close to 1.7. All of the distance/height values below 1.8 will be optimal clusters, which will make 6 clusters.

c

```r
mean(cutree(agglom, k=2) == km.out[[2]][[1]])
```

```
## [1] 0.8333333
```

```r
mean(cutree(agglom, k=3) == km.out[[3]][[1]])
```

```
## [1] 0.56
```

```r
mean(cutree(agglom, k=4) == km.out[[4]][[1]])
```

```
## [1] 0.4133333
```

```r
mean(cutree(agglom, k=5) == km.out[[5]][[1]])
```

```
## [1] 0.1
```

```r
mean(cutree(agglom, k=7) == km.out[[7]][[1]])
```

```
## [1] 0.12
```

```r
mean(cutree(agglom, k=9) == km.out[[9]][[1]])
```

```
## [1] 0.12
```

```r
mean(cutree(agglom, k=11) == km.out[[11]][[1]])
```

```
## [1] 0.1533333
```

The number and nature are significantly different from those we were given in K-means for the values of k greater than 4. You can use a cutree command to see the cluster assignments in hierarchical clustering and compare them to the assignments given in k-means.

Problem 5: DBSCAN

```r
iris_mat <- as.matrix(iris_copy)
eps <- c(0.2, 0.3, 0.4, 0.5, 0.6,
         0.8, 1)
```

```r
scan <- list()
sizes <- list()
testpreds <- list()
for(i in 1:length(eps)){
  scan[[i]] <- dbscan(iris_mat, eps = eps[i])
  testpreds[[i]] <- predict(scan[[i]], type="class")

}
classes <- as.integer(iris$Species)

scan[[1]]
```

```
## DBSCAN clustering for 150 objects.
## Parameters: eps = 0.2, minPts = 5
## The clustering contains 2 cluster(s) and 133 noise points.
##
##   0   1   2
## 133  10   7
##
## Available fields: cluster, eps, minPts
```

```r
confusionMatrix(testpreds[[1]], classes-1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1  2
##          0 33 50 50
##          1 10  0  0
##          2  7  0  0
##
## Overall Statistics
##
##                Accuracy : 0.22
##                  95% CI : (0.1565, 0.2949)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : 0.9991
##
##                   Kappa : -0.17
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0 Class: 1 Class: 2
## Sensitivity            0.6600  0.00000  0.00000
## Specificity            0.0000  0.90000  0.93000
## Pos Pred Value         0.2481  0.00000  0.00000
## Neg Pred Value         0.0000  0.64286  0.65035
## Prevalence             0.3333  0.33333  0.33333
## Detection Rate         0.2200  0.00000  0.00000
## Detection Prevalence   0.8867  0.06667  0.04667
## Balanced Accuracy      0.3300  0.45000  0.46500
```

```r
scan[[2]]
```

```
## DBSCAN clustering for 150 objects.
## Parameters: eps = 0.3, minPts = 5
## The clustering contains 3 cluster(s) and 96 noise points.
##
##  0  1  2  3
## 96 37 12  5
##
## Available fields: cluster, eps, minPts
```

```r
#confusionMatrix(testpreds[[2]], classes-1)

scan[[3]]
```

```
## DBSCAN clustering for 150 objects.
## Parameters: eps = 0.4, minPts = 5
## The clustering contains 4 cluster(s) and 32 noise points.
##
##  0  1  2  3  4
## 32 46 36 14 22
##
## Available fields: cluster, eps, minPts
```

```r
#confusionMatrix(testpreds[[3]], classes-1)

scan[[4]]
```

```
## DBSCAN clustering for 150 objects.
## Parameters: eps = 0.5, minPts = 5
## The clustering contains 2 cluster(s) and 17 noise points.
##
##  0  1  2
## 17 49 84
##
## Available fields: cluster, eps, minPts
```

```r
confusionMatrix(testpreds[[4]], classes-1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1  2
##          0  1  6 10
##          1 49  0  0
##          2  0 44 40
##
## Overall Statistics
##
##                Accuracy : 0.2733
##                  95% CI : (0.2038, 0.352)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : 0.952
##
##                   Kappa : -0.09
##  Mcnemar's Test P-Value : <2e-16
```

```
##
## Statistics by Class:
##
##                      Class: 0 Class: 1 Class: 2
## Sensitivity          0.020000   0.0000   0.8000
## Specificity          0.840000   0.5100   0.5600
## Pos Pred Value       0.058824   0.0000   0.4762
## Neg Pred Value       0.631579   0.5050   0.8485
## Prevalence           0.333333   0.3333   0.3333
## Detection Rate       0.006667   0.0000   0.2667
## Detection Prevalence 0.113333   0.3267   0.5600
## Balanced Accuracy    0.430000   0.2550   0.6800
```

```r
scan[[5]]
```

```
## DBSCAN clustering for 150 objects.
## Parameters: eps = 0.6, minPts = 5
## The clustering contains 2 cluster(s) and 9 noise points.
##
##  0  1  2
##  9 49 92
##
## Available fields: cluster, eps, minPts
```

```r
confusionMatrix(testpreds[[5]], classes-1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1  2
##          0  1  4  4
##          1 49  0  0
##          2  0 46 46
##
## Overall Statistics
##
##                Accuracy : 0.3133
##                  95% CI : (0.2402, 0.3941)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : 0.7257
##
##                   Kappa : -0.03
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: 0 Class: 1 Class: 2
## Sensitivity          0.020000   0.0000   0.9200
## Specificity          0.920000   0.5100   0.5400
## Pos Pred Value       0.111111   0.0000   0.5000
## Neg Pred Value       0.652482   0.5050   0.9310
## Prevalence           0.333333   0.3333   0.3333
## Detection Rate       0.006667   0.0000   0.3067
## Detection Prevalence 0.060000   0.3267   0.6133
## Balanced Accuracy    0.470000   0.2550   0.7300
```

```r
scan[[6]]
```

```
## DBSCAN clustering for 150 objects.
## Parameters: eps = 0.8, minPts = 5
## The clustering contains 2 cluster(s) and 2 noise points.
##
##  0  1  2
##  2 50 98
##
## Available fields: cluster, eps, minPts
```

```r
confusionMatrix(testpreds[[6]], classes-1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1  2
##          0  0  0  2
##          1 50  0  0
##          2  0 50 48
##
## Overall Statistics
##
##                Accuracy : 0.32
##                  95% CI : (0.2463, 0.401)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : 0.6646
##
##                   Kappa : -0.02
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: 0 Class: 1 Class: 2
## Sensitivity           0.00000   0.0000   0.9600
## Specificity           0.98000   0.5000   0.5000
## Pos Pred Value        0.00000   0.0000   0.4898
## Neg Pred Value        0.66216   0.5000   0.9615
## Prevalence            0.33333   0.3333   0.3333
## Detection Rate        0.00000   0.0000   0.3200
## Detection Prevalence  0.01333   0.3333   0.6533
## Balanced Accuracy     0.49000   0.2500   0.7300
```

```r
scan[[7]]
```

```
## DBSCAN clustering for 150 objects.
## Parameters: eps = 1, minPts = 5
## The clustering contains 2 cluster(s) and 0 noise points.
##
##   1   2
##  50 100
##
## Available fields: cluster, eps, minPts
```

```r
confusionMatrix(testpreds[[7]], classes-1)
```

```
## Warning in levels(reference) != levels(data): longer object length is not a
## multiple of shorter object length

## Warning in confusionMatrix.default(testpreds[[7]], classes - 1): Levels are
## not in the same order for reference and data. Refactoring data to match.

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1  2
##          0  0  0  0
##          1 50  0  0
##          2  0 50 50
##
## Overall Statistics
##
##                Accuracy : 0.3333
##                  95% CI : (0.2586, 0.4148)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : 0.5307
##
##                   Kappa : 0
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0 Class: 1 Class: 2
## Sensitivity            0.0000   0.0000   1.0000
## Specificity            1.0000   0.5000   0.5000
## Pos Pred Value            NaN   0.0000   0.5000
## Neg Pred Value         0.6667   0.5000   1.0000
## Prevalence             0.3333   0.3333   0.3333
## Detection Rate         0.0000   0.0000   0.3333
## Detection Prevalence   0.0000   0.3333   0.6667
## Balanced Accuracy      0.5000   0.2500   0.7500
```

b

For epsilon equaling 0.2, the size of the clusters is 133, 10, and 7, with a F-measure of 0.22.

For epsilon equaling 0.3, the size of the clusters is 96, 37, 12, and 5.

For epsilon equaling 0.4, the size of the clusters is 32, 46, 36, 14, and 22.

For epsilon equaling 0.5, the size of the clusters is 17, 49, and 84. The F-measure is 0.2733.

For epsilon equaling 0.6, the size of the clusters is 9, 49, and 92. The F-measure is 0.3133.

For epsilon equaling 0.8, the size of the clusters is 2, 50, and 98. The F-measure is 0.32.

For epsilon equaling 1, the size of the clusters is 50, 100. The F-measure is 0.333.

c

The epsilon value with the highest F-measure is an epsilong of 1.

d

I think the confusion matrix is super interesting. I want to learn more of what the results mean and how the matrix works. Also, I think it is interesting the F-measures continued to get better as epsilon increased.

e

The number and nature of clusters is different than k-means and heirarchical agglomerative clustering. K-means sees what points are within a certain radius from a point, but DBSCAN uses the number of points within a certain radius. Hierarchical agglomerative clustering figures out the pairwise differences between points and clusters the data points based on those differences, whether they be maximum differences, mininmum or average differences.

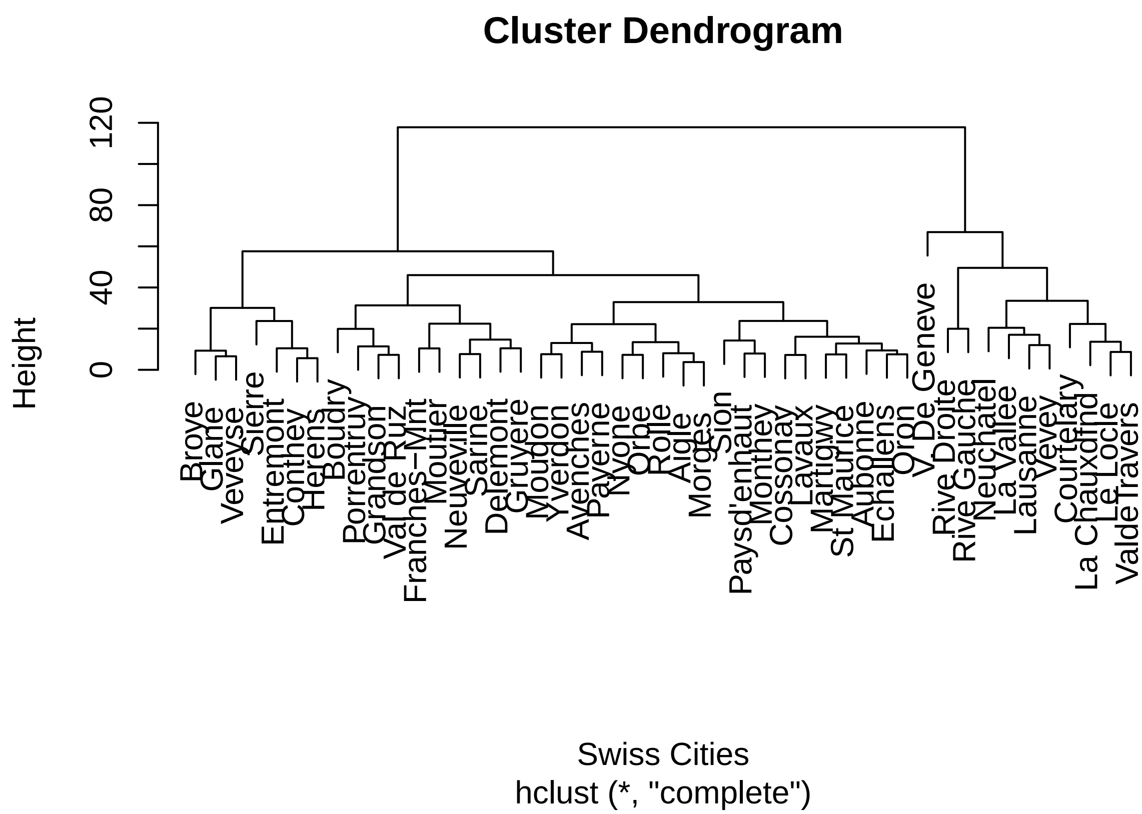
Problem 6: Swiss Dataset: Hierarchical Agglomerative Clustering

```
swiss.dat <- swiss


plot(swiss.dat$Agriculture, swiss.dat$Catholic, xlab = "Males in Agriculture",
     ylab="Percent of Cathoics", main = "Agriculture Workers vs. Catholics")
```


```
colos <- c("red", "blue", "brown")

hierach <- hclust(dist(swiss.dat[,-5]))
hierach$height
```

```
##  [1]   3.748333   5.649779   6.568105   7.206941   7.264296   7.269113
##  [7]   7.489993   7.525955   7.584853   7.637408   7.909488   8.055433
## [13]   8.681590   8.778383   9.295698   9.415944  10.399038  10.448445
## [19]  10.454186  11.384639  12.000417  12.780454  13.028047  13.435029
## [25]  13.622041  14.228844  14.678215  16.066736  17.016756  19.894472
## [31]  19.950689  20.411026  22.181298  22.285197  22.404464  23.701266
## [37]  23.746579  30.083717  31.306709  32.890576  33.526258  46.000543
## [43]  49.522722  57.591145  66.897833 117.840104
```

```r
plot(hierach, xlab = "Swiss Cities")
```




```r
cutree(hierach, k=2)
```

```
##   Courtelary     Delemont Franches-Mnt      Moutier   Neuveville
##            1            2            2            2            2
##   Porrentruy        Broye        Glane      Gruyere       Sarine
##            2            2            2            2            2
##      Veveyse        Aigle      Aubonne     Avenches     Cossonay
##            2            2            2            2            2
##    Echallens     Grandson     Lausanne    La Vallee       Lavaux
##            2            2            1            1            2
##       Morges       Moudon        Nyone         Orbe         Oron
##            2            2            2            2            2
##      Payerne Paysd'enhaut        Rolle        Vevey      Yverdon
##            2            2            2            1            2
##      Conthey    Entremont       Herens     Martigwy      Monthey
##            2            2            2            2            2
```

```
##   St Maurice        Sierre          Sion       Boudry La Chauxdfnd
##            2             2             2            2            1
##     Le Locle     Neuchatel   Val de Ruz ValdeTravers V. De Geneve
##            1             1             2            1            1
##  Rive Droite  Rive Gauche
##            1             1
```

```
#plot(swiss.dat$Infant.Mortality, y, xlab = "Infant Mortality",
     #ylab="Fertility", main = "Fertility vs. Infant Mortality", #col=cols[cutree(hierach, k=2)])
plot(swiss.dat$Agriculture, swiss.dat$Catholic, xlab = "Males in Agriculture",
     ylab="Percent of Cathoics", main = "Agriculture Workers vs. Catholics",
     col=colos[cutree(hierach, k=2)])
```