HomeWork4

Vishal Dalmiya (Dalmiya2); Himanshu Shah (Hs8); Deepak Nagarajan (deepakn2)

February 22, 2018

Problem 1

You can find a dataset dealing with European employment in 1979 at http://lib.stat.cmu.edu/DASL/Stories/EuropeanJobs.html. This dataset gives the percentage of people employed in each of a set of areas in 1979 for each of a set of European countries. Notice this dataset contains only 26 data points. That's fine; it's intended to give you some practice in visualization of clustering.

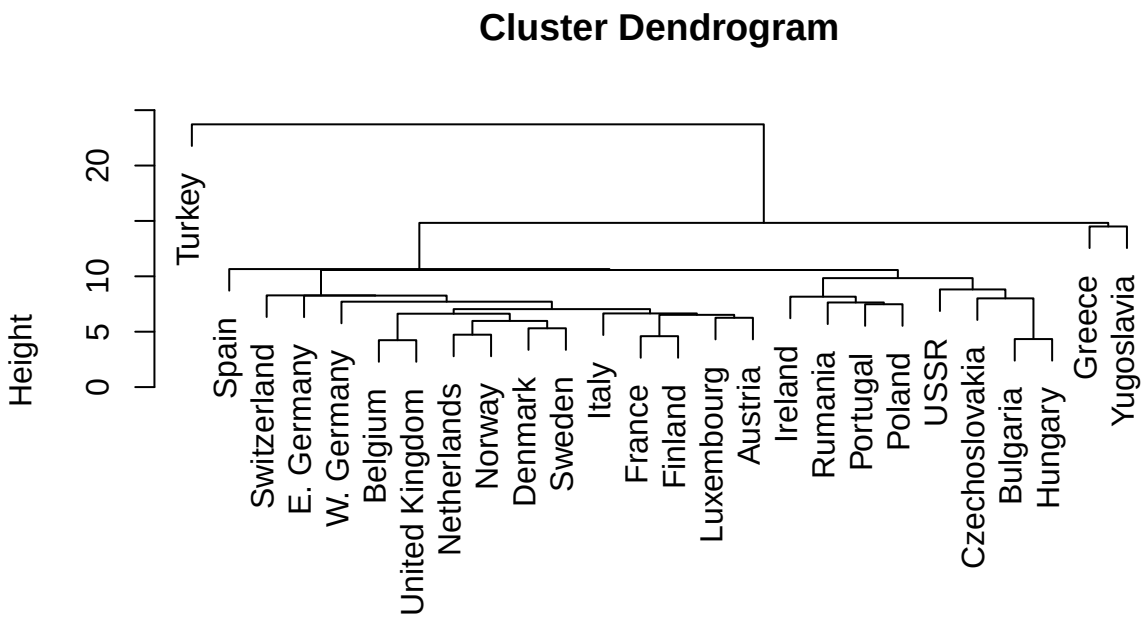
Use an agglomerative clusterer to cluster this data. Produce a dendrogram of this data for each of single link, complete link, and group average clustering. You should label the countries on the axis. What structure in the data does each method expose? it's fine to look for code, rather than writing your own. Hint: I made plots I liked a lot using R's hclust clustering function, and then turning the result into a phylogenetic tree and using a fan plot, a trick I found on the web; try plot(as.phylo(hclustresult), type='fan'). You should see dendrograms that "make sense" (at least if you remember some European history), and have interesting differences.

```
job = read.csv(".\\euJob.csv", header = TRUE)
d = dist(job[, 2:ncol(job)])
library(ape)
```

```
## Warning: package 'ape' was built under R version 3.4.3
```

Single Link

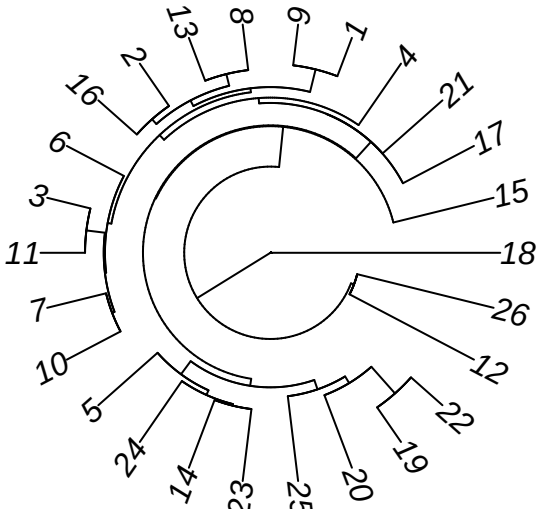
```
single = hclust(d , method = "single")
plot(single, labels = job[, 1])
```




```
single$labels
```

```
## NULL
```

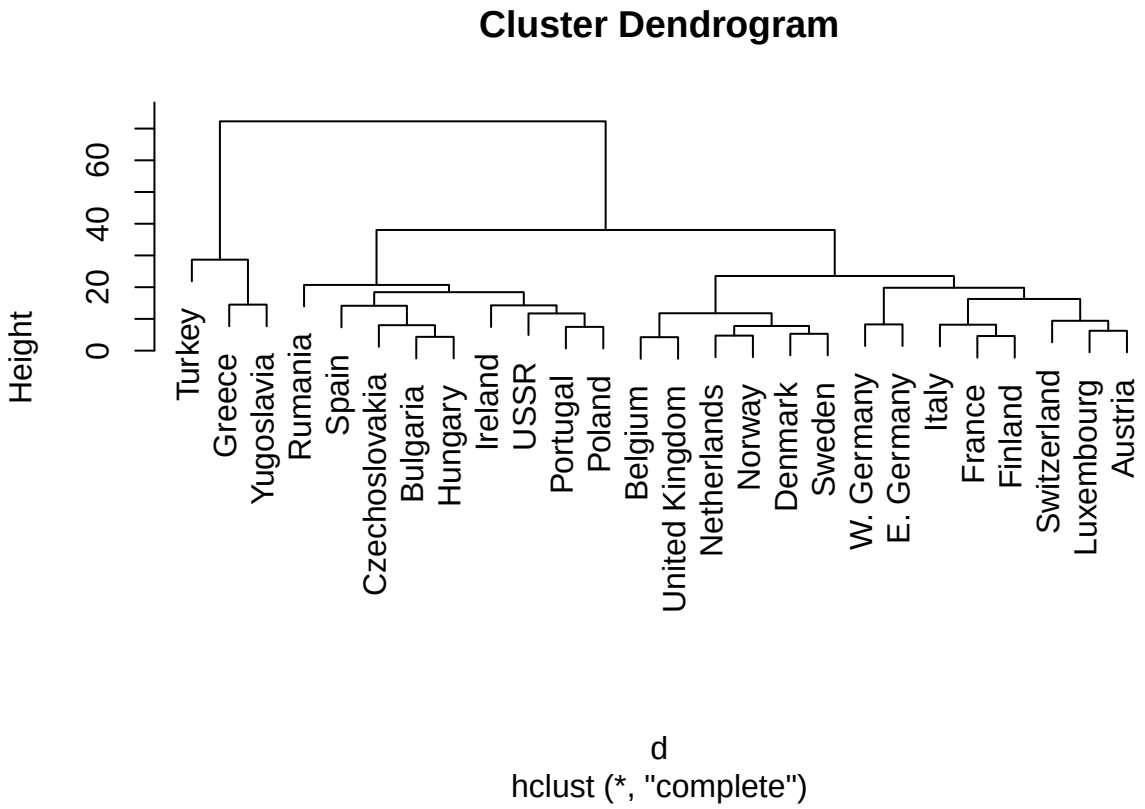
```
plot(as.phylo(single, labels = single$labels), type = 'fan')
```



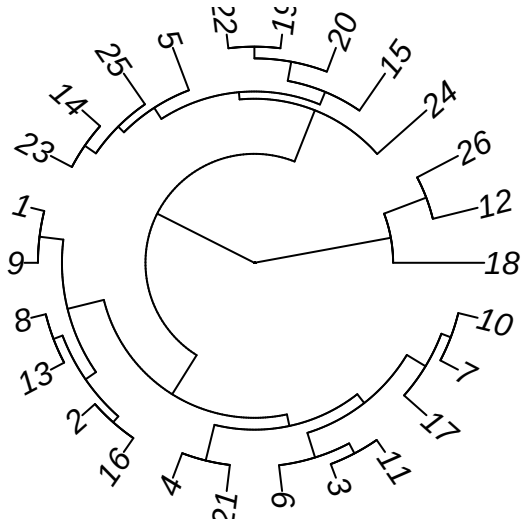
- For single link, If We make cut at height of 13, then We can see there are 4 unique clusters being exposed.
- Looking at raw data, it seems that Turkey's agriculture/manufacturing is odd one out compared to rest of EU. Greece and Yugoslavia also seems to fall between Turkey and rest of EU, which results in separate cluster. Hence above clustering of 4 aligns with raw data.

Complete Link

```
complete = hclust(d , method = "complete")
plot(complete, labels = job[, 1])
```



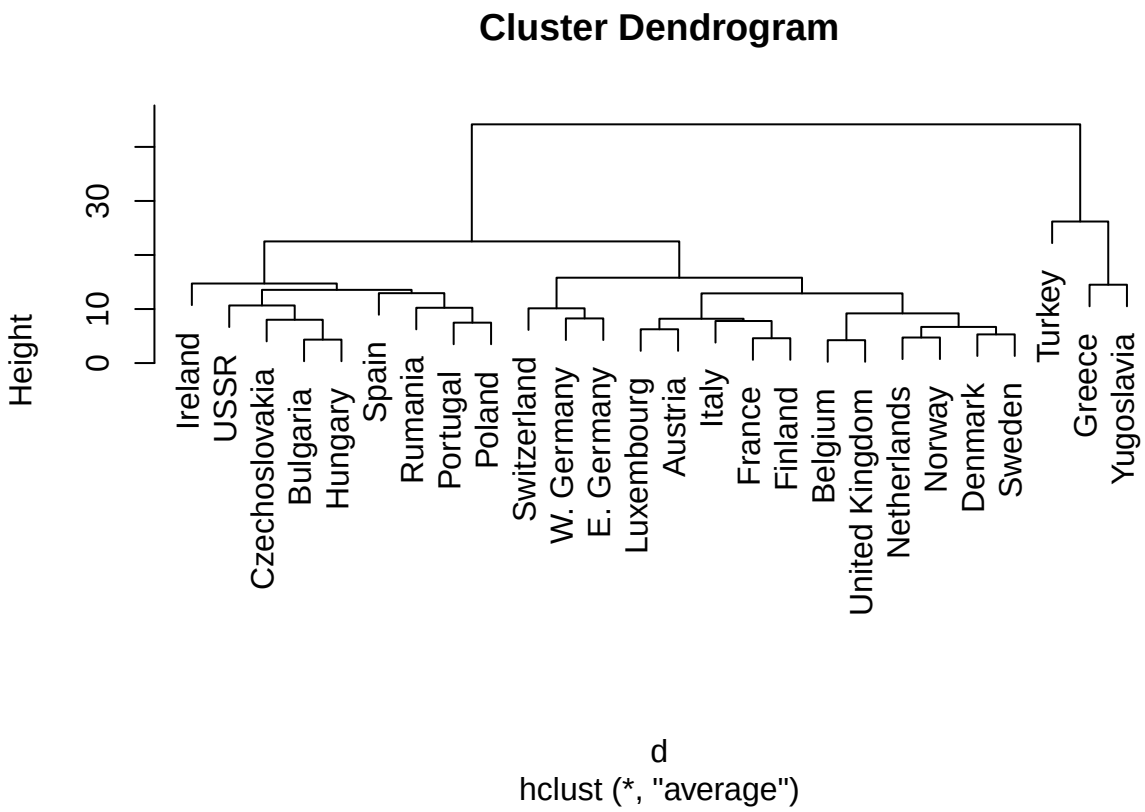

```
plot(as.phylo(complete), type = 'fan')
```



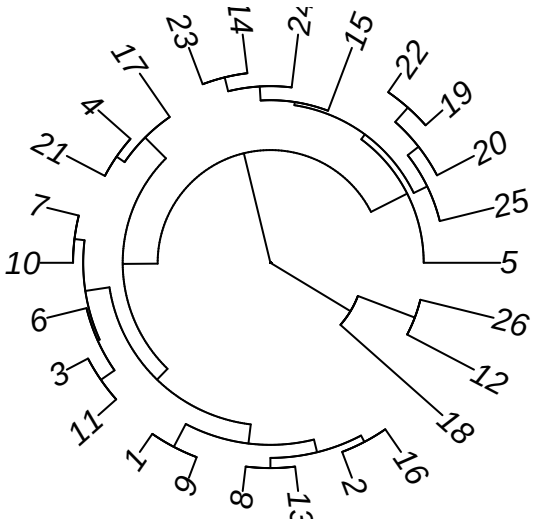
- For complete link, If We make cut at height of 27, then We can see there are 4 unique clusters being exposed.
- Looking at raw data, it seems that Turkey's agriculture/manufacturing is odd one out compared to rest of EU. Greece and Yugoslavia also seems to fall between Turkey and rest of EU, which results in separate cluster. Hence above clustering of 4 aligns with raw data.

Group Average

```
avg = hclust(d , method = "average")
plot(avg, labels = job[, 1])
```




```
plot(as.phylo(avg, labels = job[, 1]), type = 'fan')
```



- For average link, If We make cut at height of 20, then We can see there are 4 unique clusters being exposed.
- Looking at raw data, it seems that Turkey's agriculture/manufacturing is odd one out compared to rest of EU. Greece and Yugoslavia also seems to fall between Turkey and rest of EU, which results in separate cluster. Hence above clustering of 4 aligns with raw data.
- In general, We can clearly see from all three dendrograms, these are the following clusters. – Turkey – Greece/Yugoslavia – Rest of EU

Using k-means, cluster this dataset. What is a good choice of k for this data and why?

```
two = kmeans(job[,2:ncol(job)],2)
two$size
```

```
## [1] 21  5
```

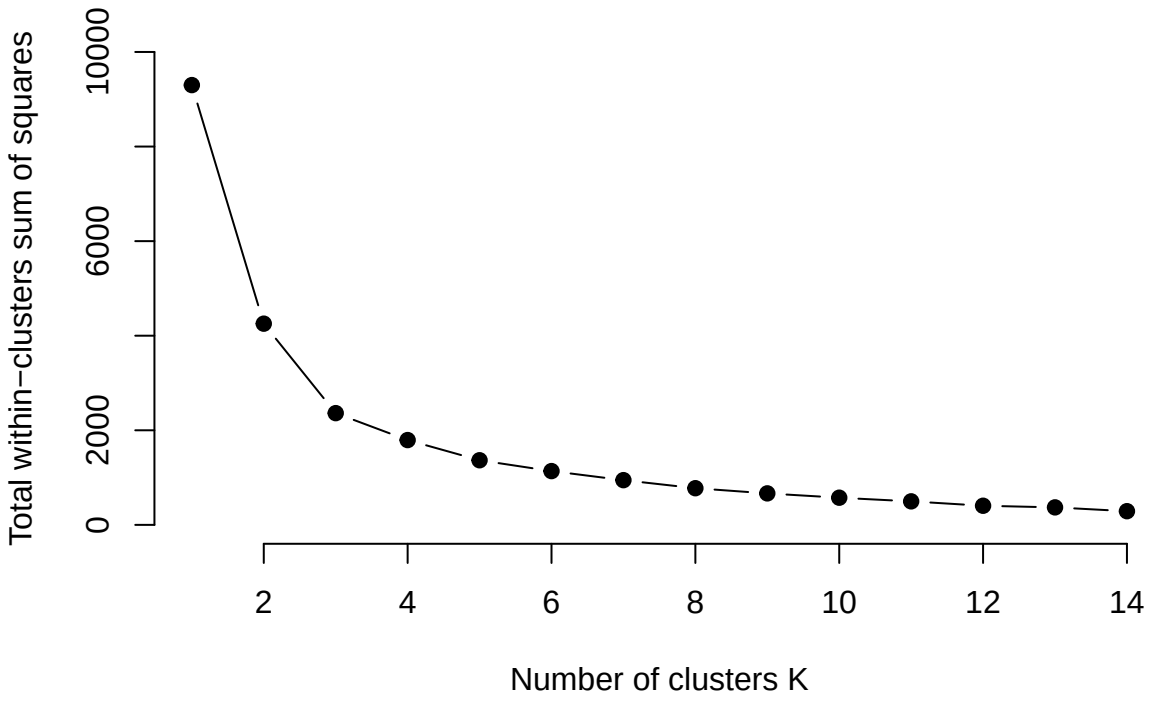
```
three = kmeans(job[,2:ncol(job)],3)
three$size
```

```
## [1] 14  3  9
```

```
k.max = 14
data = job[,2:ncol(job)]
wss = sapply(1:k.max,
             function(k){kmeans(data, k, nstart=50,iter.max = 15 )$tot.withinss})

plot(1:k.max, wss,
```

```
     type="b", pch = 19, frame = FALSE,
     xlab="Number of clusters K",
     ylab="Total within-clusters sum of squares",
     ylim= c(0, 10000))
```



- From the plot above its eveident that the total within-clusters sum of squares goes down with number of cluster and the knee of the above plot indicates **k=4** is a good optimal choice.

Problem 2

Do exercise 6.2 in the Jan 15 version of the course text Obtain the activities of daily life dataset from the UC Irvine machine learning website (https://archive.ics.uci.edu/ml/datasets/Dataset+for+ADL+Recognition+with+Wrist-worn+Accelerometer data provided by Barbara Bruno, Fulvio Mastrogiovanni and Antonio Sgor-bissa).

```
# Slice function to split the each file into chunks of 32*3 = 96
slice = function(input, by=32){
  length = trunc(nrow(input)/by) * by
  starts = seq(1,length,by)
  tt = lapply(starts, function(y) unlist(input[y:(y+(by-1)),]))
}
```

```
# Read all the data
num_seg  = 64 # 12, 32, 64
combined = data.frame()
```

```
combined_test = data.frame()

dirs = list.dirs(path = "HMP_Dataset\\.",
                 full.names = TRUE,
                 recursive = TRUE)
dirs = dirs[-1]
num_cat = length(dirs)

# Read all directories and the files from those sub-directories
for(d_idx in 1:length(dirs)){
  d = dirs[d_idx]
  dr = gsub("/", "\\", d, fixed = TRUE)
  dr = paste(dr, "\\*.txt", sep = "")
  files = (Sys.glob(dr))

  # Cerate all indexes of the files
  all_idx = 1:length(files)
  # Need to do the 80/20 split here
  train_idx = 1:floor(length(all_idx) * 0.8)
  test_idx  = all_idx[-train_idx]

  # For train data
  for (f_idx in train_idx)
  {
    f = files[f_idx]
    read_file = read.table(f)
    slc = slice(read_file,num_seg)
    segment = as.data.frame(t(as.data.frame(slc)))
    segment = unname(segment)
    segment[, ncol(segment) + 1] = f_idx
    segment[, ncol(segment) + 1] = d_idx
    combined = rbind(combined, segment)
  }

  # For test data
  for (f_idx in test_idx)
  {
    f = files[f_idx]
    read_file = read.table(f)
    slc = slice(read_file,num_seg)
    segment = as.data.frame(t(as.data.frame(slc)))
    segment = unname(segment)
    segment[, ncol(segment) + 1] = f_idx
    segment[, ncol(segment) + 1] = d_idx
    combined_test = rbind(combined_test, segment)
  }
}

rownames(combined) = NULL
combined = unname(combined)

rownames(combined_test) = NULL
combined_test = unname(combined_test)
```

```
# Apply K-Means to entire dataset
k = 480 #1000, 1500, 3000
#k = floor(nrow(combined) / 10)

# Normal kmeans
#cluster = kmeans(combined[, 1:(ncol(combined) - 2)], k, iter.max = 100, nstart = 20)

# Heirarchical k-means
library("factoextra")
```

```
## Warning: package 'factoextra' was built under R version 3.4.3
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.4.1
```

```
## Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at https://goo.gl/13EFCZ
```

```
cluster = hkmeans(combined[, 1:(ncol(combined) - 2)], k)

result      = cbind(combined, clusterNum = cluster$cluster)
classify_df = data.frame(matrix(ncol = k + 1))
classify_df_test = data.frame(matrix(ncol = k + 1))
idx         = 1
start_pos   = 0
inst_col    = 3 * num_seg + 1
cat_col     = 3 * num_seg + 2
clust_col   = 3 * num_seg + 3


clusters <- function(test_data, centers) {
  euc_dist = sapply(seq_len(nrow(centers)), function(x)
    apply(test_data, 1, function(v)
      sum((
        as.double(v) - cluster$centers[x, ]
      ) ^ 2)))

  sapply(seq_len(nrow(euc_dist)), function(x)
    which.min(euc_dist[x, ]))
}


test_centers = clusters(combined_test[, 1:(ncol(combined_test) - 2)],cluster$centers)

result_test  = cbind(combined_test, clusterNum = test_centers)

for (cat_idx in 1:num_cat)
{
  # Find the number of entries for an entire category across all files
  # for the given category
  num_entries_cat = sum(result[, cat_col] == cat_idx)

  # Find the start & end index in the result data frame per category
  end_pos   = start_pos + num_entries_cat
  start_pos = start_pos + 1
```

```
  # Find the number of instance per category. Should match the num of files
  # in each category folder
  num_inst = max(as.data.frame(result[start_pos:end_pos, inst_col]))
  # Contains entries for only this category
  res_cat  = result[start_pos:end_pos, ]

  for (inst_idx in 1:num_inst)
  {
    # Find the num of entries per instance
    num_entries_inst = sum(res_cat[, inst_col] == inst_idx)
    clus_res = res_cat[1:num_entries_inst, clust_col]
    num_entries_inst = num_entries_inst + 1
    # Remove the processed instance
    res_cat = res_cat[num_entries_inst:nrow(res_cat), ]
    # Reset the feature values to 0
    classify_df[idx,]  = 0
    # Compute the freq for each feature
    hist = table(clus_res)
    # Find the list of feature set
    feature_idx = as.integer(rownames(hist)) + 1
    # Set the freq for the feature set identified above
    classify_df[idx, feature_idx] = as.integer(hist)
    # Set the category idx for labeling
    classify_df[idx, 1] = cat_idx
    idx = idx + 1
  }
  start_pos = start_pos + num_entries_cat - 1
}

idx         = 1
start_pos   = 0

for (cat_idx in 1:num_cat)
{
  # Find the number of entries for an entire category across all files
  # for the given category
  num_entries_cat = sum(result_test[, cat_col] == cat_idx)

  # Find the start & end index in the result_test data frame per category
  end_pos   = start_pos + num_entries_cat
  start_pos = start_pos + 1

  # Find the number of instance per category. Should match the num of files
  # in each category folder
  num_inst = max(as.data.frame(result_test[start_pos:end_pos, inst_col])) -
    min(as.data.frame(result_test[start_pos:end_pos, inst_col])) + 1
  min_val = min(as.data.frame(result_test[start_pos:end_pos, inst_col])) - 1

  # Contains entries for only this category
  res_cat  = result_test[start_pos:end_pos, ]

  for (inst_idx in 1:num_inst)
  {
```

```
    # Find the num of entries per instance
    num_entries_inst = sum(res_cat[, inst_col] == (inst_idx + min_val))
    clus_res = res_cat[1:num_entries_inst, clust_col]
    num_entries_inst = num_entries_inst + 1
    # Remove the processed instance
    res_cat = res_cat[num_entries_inst:nrow(res_cat), ]
    # Reset the feature values to 0
    classify_df_test[idx,]  = 0
    # Compute the freq for each feature
    hist = table(clus_res)
    # Find the list of feature set
    feature_idx = as.integer(rownames(hist)) + 1
    # Set the freq for the feature set identified above
    classify_df_test[idx, feature_idx] = as.integer(hist)
    # Set the category idx for labeling
    classify_df_test[idx, 1] = cat_idx
    idx = idx + 1
  }
  start_pos = start_pos + num_entries_cat - 1
}
```

(a) Build a classifier that classifies sequences into one of the 14 activities pro-vided. To make features, you should vector quantize, then use a histogram of cluster centers (as described in the subsection; this gives a pretty ex-plicit set of steps to follow). You will find it helpful to use hierarchical k-means to vector quantize. You may use whatever multi-class classifieryou wish, though I'd start with R's decision forest, because it's easy touse and effective. You should report (a) the total error rate and (b) the class confusion matrix of your classifier.

```
# Random forest classifier
set.seed(03042018)

library(h2o)
library(klaR)
library(caret)
library(randomForest)
h2o.init()
```

```
##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         1 days 9 hours
##     H2O cluster timezone:       America/New_York
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.18.0.1
##     H2O cluster version age:    20 days
##     H2O cluster name:           H2O_started_from_R_dnagarajan_ruc736
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   2.03 GB
##     H2O cluster total cores:    4
##     H2O cluster allowed cores:  4
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
```

```
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     H2O API Extensions:         Algos, AutoML, Core V3, Core V4
##     R Version:                  R version 3.4.0 (2017-04-21)
```

```
rf_train = classify_df[, 2:ncol(classify_df)]
rf_train["class"] = as.factor(classify_df[, 1])

rf_test = classify_df_test[, 2:ncol(classify_df_test)]
rf_test["class"] = as.factor(classify_df_test[, 1])

num_trees = c(10, 20, 30)
depth = c(4, 8, 16)

for (nt in num_trees)
{
  for (d in depth)
  {
    rf =
      h2o.randomForest(
        x = colnames(rf_train),
        y = "class",
        training_frame = as.h2o(rf_train),
        ntrees = nt,
        max_depth = d
      )

    labels = as.data.frame(h2o.predict(rf, as.h2o(rf_test)))

    cm       = confusionMatrix(data = as.factor(labels[, 1]), rf_test$class)

    print(paste0("#trees = ", nt))
    print(paste0("#depth = ", d))
    print(paste0("accuracy = ", cm$overall[1] * 100))
    print(paste0("Error rate = ", (1 - cm$overall[1]) * 100))
  }
}
```

```
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
##
  |
  |                                                                      |   0%
  |
  |=========================                                             |  40%
  |
  |======================================================================| 100%
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
```

```
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
## [1] "#trees = 10"
## [1] "#depth = 4"
## [1] "accuracy = 65.8959537572254"
## [1] "Error rate = 34.1040462427746"
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
## [1] "#trees = 10"
## [1] "#depth = 8"
## [1] "accuracy = 64.7398843930636"
## [1] "Error rate = 35.2601156069364"
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
## [1] "#trees = 10"
```

```
## [1] "#depth = 16"
## [1] "accuracy = 72.2543352601156"
## [1] "Error rate = 27.7456647398844"
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
##
  |
  |                                                                      |   0%
  |
  |======                                                                |  10%
  |
  |======================================================================| 100%
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
## [1] "#trees = 20"
## [1] "#depth = 4"
## [1] "accuracy = 67.6300578034682"
## [1] "Error rate = 32.3699421965318"
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
## [1] "#trees = 20"
## [1] "#depth = 8"
## [1] "accuracy = 71.0982658959538"
## [1] "Error rate = 28.9017341040462"
##
```

```
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
##
  |
  |                                                                      |   0%
  |
  |===                                                                   |   5%
  |
  |================================================                      |  75%
  |
  |======================================================================| 100%
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
## [1] "#trees = 20"
## [1] "#depth = 16"
## [1] "accuracy = 72.2543352601156"
## [1] "Error rate = 27.7456647398844"
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
##
  |
  |                                                                      |   0%
  |
  |==                                                                    |   3%
  |
  |======================================================================| 100%
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
## [1] "#trees = 30"
## [1] "#depth = 4"
## [1] "accuracy = 67.0520231213873"
## [1] "Error rate = 32.9479768786127"
##
```

```
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
##
  |
  |                                                                      |   0%
  |
  |=================================================                     |  80%
  |
  |======================================================================| 100%
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
## [1] "#trees = 30"
## [1] "#depth = 8"
## [1] "accuracy = 72.8323699421965"
## [1] "Error rate = 27.1676300578035"
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
##
  |
  |                                                                      |   0%
  |
  |============================                                          |  43%
  |
  |=======================================================               |  87%
  |
  |======================================================================| 100%
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
## [1] "#trees = 30"
## [1] "#depth = 16"
## [1] "accuracy = 75.1445086705202"
## [1] "Error rate = 24.8554913294798"
```

```
# Print confusion matrix
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2  3  4  5  6  7  8  9 10 11 12 13 14
##         1   3  0  0  0  0  0  0  0  0  0  0  0  0  0
##         2   0 18  0  0  0  0  0  2  0  0  1  0  0  0
##         3   0  0  5  0  0  0  0  0  0  0  0  0  0  0
##         4   0  0  0  6  0  0  0  0  0  0  0  0  0  0
##         5   0  0  2  0 20  0  0  1  0  1  1  0  2  0
##         6   0  0  0  0  0  1  0  0  0  0  0  0  0  0
##         7   0  0  0  0  0  0  1  0  0  0  0  0  0  0
##         8   0  0  0  0  0  0  0 10  3  0  4  4  0  0
##         9   0  0  0  0  0  0  0  0  0  0  0  0  0  0
##         10  0  0  0  0  0  0  0  1  0 19  1  2  0  0
##         11  0  0  0  1  0  0  0  1  3  0 12  1  0  0
##         12  0  0  0  0  0  0  0  6  0  0  1 14  0  0
##         13  0  0  0  0  0  0  0  0  0  0  0  0  1  0
##         14  0  3  0  2  0  0  0  0  0  0  0  0  0 20
##
## Overall Statistics
##
##                Accuracy : 0.7514
##                  95% CI : (0.6802, 0.8139)
##     No Information Rate : 0.1214
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7208
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5 Class: 6
## Sensitivity           1.00000   0.8571  0.71429  0.66667   1.0000  1.00000
## Specificity           1.00000   0.9803  1.00000  1.00000   0.9542  1.00000
## Pos Pred Value        1.00000   0.8571  1.00000  1.00000   0.7407  1.00000
## Neg Pred Value        1.00000   0.9803  0.98810  0.98204   1.0000  1.00000
## Prevalence            0.01734   0.1214  0.04046  0.05202   0.1156  0.00578
## Detection Rate        0.01734   0.1040  0.02890  0.03468   0.1156  0.00578
## Detection Prevalence  0.01734   0.1214  0.02890  0.03468   0.1561  0.00578
## Balanced Accuracy     1.00000   0.9187  0.85714  0.83333   0.9771  1.00000
##                      Class: 7 Class: 8 Class: 9 Class: 10 Class: 11
## Sensitivity           1.00000   0.4762  0.00000    0.9500   0.60000
## Specificity           1.00000   0.9276  1.00000    0.9739   0.96078
## Pos Pred Value        1.00000   0.4762      NaN    0.8261   0.66667
## Neg Pred Value        1.00000   0.9276  0.96532    0.9933   0.94839
## Prevalence            0.00578   0.1214  0.03468    0.1156   0.11561
## Detection Rate        0.00578   0.0578  0.00000    0.1098   0.06936
## Detection Prevalence  0.00578   0.1214  0.00000    0.1329   0.10405
## Balanced Accuracy     1.00000   0.7019  0.50000    0.9619   0.78039
##                      Class: 12 Class: 13 Class: 14
## Sensitivity            0.66667   0.33333    1.0000
```

```
## Specificity            0.95395   1.00000    0.9673
## Pos Pred Value         0.66667   1.00000    0.8000
## Neg Pred Value         0.95395   0.98837    1.0000
## Prevalence             0.12139   0.01734    0.1156
## Detection Rate         0.08092   0.00578    0.1156
## Detection Prevalence   0.12139   0.00578    0.1445
## Balanced Accuracy      0.81031   0.66667    0.9837
```

(b) Now see if you can improve your classifier by (a) modifying the number of cluster centers in your hierarchical k-means and (b) modifying the size of the fixed length samples that you use.

```
# Print confusion matrix from the best Accuracy (k=480, Segment size =64)
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2  3  4  5  6  7  8  9 10 11 12 13 14
##         1   3  0  0  0  0  0  0  0  0  0  0  0  0  0
##         2   0 18  0  0  0  0  0  2  0  0  1  0  0  0
##         3   0  0  5  0  0  0  0  0  0  0  0  0  0  0
##         4   0  0  0  6  0  0  0  0  0  0  0  0  0  0
##         5   0  0  2  0 20  0  0  1  0  1  1  0  2  0
##         6   0  0  0  0  0  1  0  0  0  0  0  0  0  0
##         7   0  0  0  0  0  0  1  0  0  0  0  0  0  0
##         8   0  0  0  0  0  0  0 10  3  0  4  4  0  0
##         9   0  0  0  0  0  0  0  0  0  0  0  0  0  0
##         10  0  0  0  0  0  0  0  1  0 19  1  2  0  0
##         11  0  0  0  1  0  0  0  1  3  0 12  1  0  0
##         12  0  0  0  0  0  0  0  6  0  0  1 14  0  0
##         13  0  0  0  0  0  0  0  0  0  0  0  0  1  0
##         14  0  3  0  2  0  0  0  0  0  0  0  0  0 20
##
## Overall Statistics
##
##                Accuracy : 0.7514
##                  95% CI : (0.6802, 0.8139)
##     No Information Rate : 0.1214
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7208
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5 Class: 6
## Sensitivity           1.00000   0.8571  0.71429  0.66667   1.0000  1.00000
## Specificity           1.00000   0.9803  1.00000  1.00000   0.9542  1.00000
## Pos Pred Value        1.00000   0.8571  1.00000  1.00000   0.7407  1.00000
## Neg Pred Value        1.00000   0.9803  0.98810  0.98204   1.0000  1.00000
## Prevalence            0.01734   0.1214  0.04046  0.05202   0.1156  0.00578
## Detection Rate        0.01734   0.1040  0.02890  0.03468   0.1156  0.00578
```

```
## Detection Prevalence  0.01734   0.1214  0.02890  0.03468   0.1561  0.00578
## Balanced Accuracy     1.00000   0.9187  0.85714  0.83333   0.9771  1.00000
##                      Class: 7 Class: 8 Class: 9 Class: 10 Class: 11
## Sensitivity           1.00000   0.4762  0.00000    0.9500   0.60000
## Specificity           1.00000   0.9276  1.00000    0.9739   0.96078
## Pos Pred Value        1.00000   0.4762      NaN    0.8261   0.66667
## Neg Pred Value        1.00000   0.9276  0.96532    0.9933   0.94839
## Prevalence            0.00578   0.1214  0.03468    0.1156   0.11561
## Detection Rate        0.00578   0.0578  0.00000    0.1098   0.06936
## Detection Prevalence  0.00578   0.1214  0.00000    0.1329   0.10405
## Balanced Accuracy     1.00000   0.7019  0.50000    0.9619   0.78039
##                      Class: 12 Class: 13 Class: 14
## Sensitivity            0.66667   0.33333    1.0000
## Specificity            0.95395   1.00000    0.9673
## Pos Pred Value         0.66667   1.00000    0.8000
## Neg Pred Value         0.95395   0.98837    1.0000
## Prevalence             0.12139   0.01734    0.1156
## Detection Rate         0.08092   0.00578    0.1156
## Detection Prevalence   0.12139   0.00578    0.1445
## Balanced Accuracy      0.81031   0.66667    0.9837
```

Table of observations

value of K	Segment Size	Depth of the Tree	No of Trees	Error rate
500	12	8	30	28.90
1000	12	16	20	31.79
480	32	8	20	26.01
1000	32	4	20	27.16
1500	32	16	20	27.74
480	64	16	30	23.90
1000	64	8	20	30.05

- From this table it is inferred that when K=480, Segment size = 64 and Depth = 16 and no of trees = 30, gives us the best Accuracy and lower error rate. Most times we got error rates below 23.9%.