```r
#==============================================================================
#============= STATISTICAL LEARNING, HW6 problem 2 ============================
#==============================================================================



# Necessary libraries
library(kknn)
library(fcd)
library(mclust)
library(speccalt)
library(dplyr)
library(kernlab)

# Set working directory
setwd("C:/Users/jrdha/OneDrive/Desktop/USU_Fa2018/Moon__SLDM2/hw6/problem2")

#======================== READ IN, FORMAT DATA ================================
#==============================================================================
# NOTE: used the helpful code that Matt shared on Piazza to do this part.
# need R.utils package installed and files downloaded in working directory.

# gunzip the files
R.utils::gunzip("train-images-idx3-ubyte.gz")
R.utils::gunzip("train-labels-idx1-ubyte.gz")
R.utils::gunzip("t10k-images-idx3-ubyte.gz")
R.utils::gunzip("t10k-labels-idx1-ubyte.gz")

# helper function for visualization
show_digit = function(arr784, col = gray(12:1 / 12), ...) {
  image(matrix(as.matrix(arr784[-785]), nrow = 28)[, 28:1], col = col, ...)
}

# load image files
load_image_file = function(filename) {
  ret = list()
  f = file(filename, 'rb')
  readBin(f, 'integer', n = 1, size = 4, endian = 'big')
  n    = readBin(f, 'integer', n = 1, size = 4, endian = 'big')
  nrow = readBin(f, 'integer', n = 1, size = 4, endian = 'big')
  ncol = readBin(f, 'integer', n = 1, size = 4, endian = 'big')
  x = readBin(f, 'integer', n = n * nrow * ncol, size = 1, signed = FALSE)
  close(f)
  data.frame(matrix(x, ncol = nrow * ncol, byrow = TRUE))
}
```

```r
# load label files
load_label_file = function(filename) {
  f = file(filename, 'rb')
  readBin(f, 'integer', n = 1, size = 4, endian = 'big')
  n = readBin(f, 'integer', n = 1, size = 4, endian = 'big')
  y = readBin(f, 'integer', n = n, size = 1, signed = FALSE)
  close(f)
  y
}

# load images
train = load_image_file("train-images-idx3-ubyte")
test  = load_image_file("t10k-images-idx3-ubyte")

# load labels
train$y = as.factor(load_label_file("train-labels-idx1-ubyte"))
test$y  = as.factor(load_label_file("t10k-labels-idx1-ubyte"))

# view test image
# show_digit(train[10000, ])


#================== SubSamp FUNCTION ========================================
#============================================================================
# SubSamp takes as arguments: givenData and sampSize. Generates (and returns) a
# random sub-sample of sampSize from givenData.
SubSamp <- function(givenData, sampSize){
  samp <- dplyr::sample_n(tbl = givenData, size = sampSize, replace = FALSE)
  return(samp)
}


#==== GENERATE SUBSAMPLES TO BE USED FOR K-MEANS AND SPECTRAL CLUSTERING =======
#============================================================================

# Set seed for reproducibility
set.seed(2345)
# Specify the number of sub-samples to generate
numSubSamp <- 20
# Specify the number of observations in each sub-sample
subSampSize <- 2000

# Generate a list ofsub-samples (will be used for k-means and spectral
# clustering)
subSampList <- list()
for (i in 1:numSubSamp) {
  subSampList[[i]] <- SubSamp(givenData = train, sampSize = subSampSize)
}
```

```r
#========= TRAINING K-MEANS CLUSTERING ON SUB-SAMPLED DATA ====================
#==============================================================================

# Store the ARI value for each sub-sample in this object
ARI.km.train <- double(numSubSamp)

# Loop over subsamples
counter <- 1 # Initialize counter var
for(ss in subSampList){
  km.train <- kmeans(x = select(ss, -y), centers = 10, nstart = 20)
  ARI.km.train[counter] <- adjustedRandIndex(km.train$cluster, ss$y)
  counter <- counter + 1
}
ARI.km.train.val <- mean(ARI.km.train)
ARI.km.train.val


#=============== APPLYING K-MEANS CLUSTERING TO TEST DATA ======================
#==============================================================================
ARI.km.test <- kmeans(x = select(test, -y), centers = 10, nstart = 20)
ARI.km.test.val <- adjustedRandIndex(ARI.km.test$cluster, test$y)
ARI.km.test.val


#========= TRAINING SPECTRAL CLUSTERING ON SUB-SAMPLED DATA ====================
#==============================================================================
# Selecting one sub-sample to be used for tuning of simga parameter
# (The sigma parameter is a proxy for bandwidth here)
set.seed(2345)
subSampTune <- SubSamp(train, 400)
# sigmaGrid <- 1*10^(-10:-5) # Initial rough grid: ARI max at 1e-06
sigmaGrid <- 1*10^(seq(-7, -5, by = 0.1)) # Narrowed down to this grid

ARI.sc.tune <- double(length(sigmaGrid))
counter <- 1
for(sg in sigmaGrid){
  sc.train <- specc(as.matrix(dplyr::select(subSampTune, -y)),
                    centers = 10,
                    kernel = "rbfdot",
                    kpar = list("sigma" = sg))
  ARI.sc.tune[counter] <- adjustedRandIndex(sc.train@.Data, subSampTune$y)
  counter <- counter + 1
}
max(ARI.sc.tune)

# selected sigma = 5.011872e-06
sigma = 5.011872 * (10^(-6))
```

```r
ARI.sc.train <- double(numSubSamp)
counter <- 1
for(ss in subSampList){
  sc.train <- specc(as.matrix(dplyr::select(ss, -y)),
                    centers = 10,
                    kernel = "rbfdot",
                    kpar = list("sigma" = sigma))
  ARI.sc.train[counter] <- adjustedRandIndex(sc.train@.Data, ss$y)
  counter <- counter + 1
}
ARI.sc.train.val <- mean(ARI.sc.train)
ARI.sc.train.val


#=============== APPLYING SPECTRAL CLUSTERING TO TEST DATA ====================
#==============================================================================
set.seed(2345)
ARI.sc.test <- specc(as.matrix(dplyr::select(test, -y)),
                     centers = 10,
                     kernel = "rbfdot",
                     kpar = list("sigma" = sigma))
ARI.sc.test.val <- adjustedRandIndex(ARI.sc.test@.Data, test$y)
ARI.sc.test.val

#========= TRAINING GMM ON SUB-SAMPLED DATA ===================================
#==============================================================================
# these seem to take longer to train, so at this point I regenerated the sub-sample
s,
# this time with 500 observations per sample.
set.seed(2345)

numSubSamp <- 20 # number of sub samples to generate
subSampSize <- 1000 # number of observations in each sub sample

# generate list of subsamples to be used for GMM
subSampList <- list()
for (i in 1:numSubSamp) {
  subSampList[[i]] <- SubSamp(givenData = train, sampSize = subSampSize)
}

ARI.gmm.train <- double(numSubSamp)
counter <- 1
for(ss in subSampList){
  gmm.train <- Mclust(dplyr::select(ss, -y), centers = 10)
  ARI.gmm.train[counter] <- adjustedRandIndex(gmm.train$classification, ss$y)
  counter <- counter + 1
}
ARI.gmm.train.val <- mean(ARI.gmm.train)
ARI.gmm.train.val
```

```r
#=============== APPLYING GMM TO TEST DATA ===================================
#=============================================================================

# Generate list of subsample from the test data
subSampList <- list()
for (i in 1:numSubSamp) {
  subSampList[[i]] <- SubSamp(givenData = test, sampSize = subSampSize)
}

ARI.gmm.test <- double(numSubSamp)
counter <- 1
for(ss in subSampList){
  gmm.test <- Mclust(dplyr::select(ss, -y), centers = 10)
  ARI.gmm.test[counter] <- adjustedRandIndex(gmm.test$classification, ss$y)
  counter <- counter + 1
}
ARI.gmm.test.val <- mean(ARI.gmm.test)
ARI.gmm.test.val
```