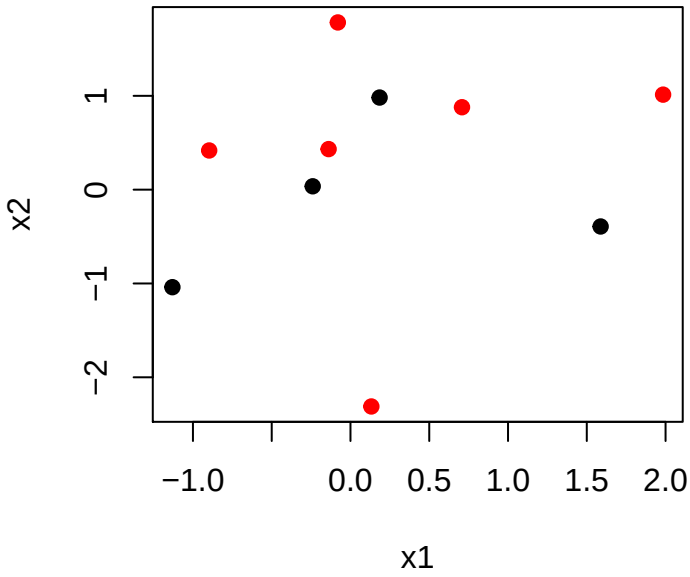
Stat 432 Homework 2

Assigned: Feb 3, 2019; Due: 11:59pm Feb 8, 2019

Question 1 (understand k-means)

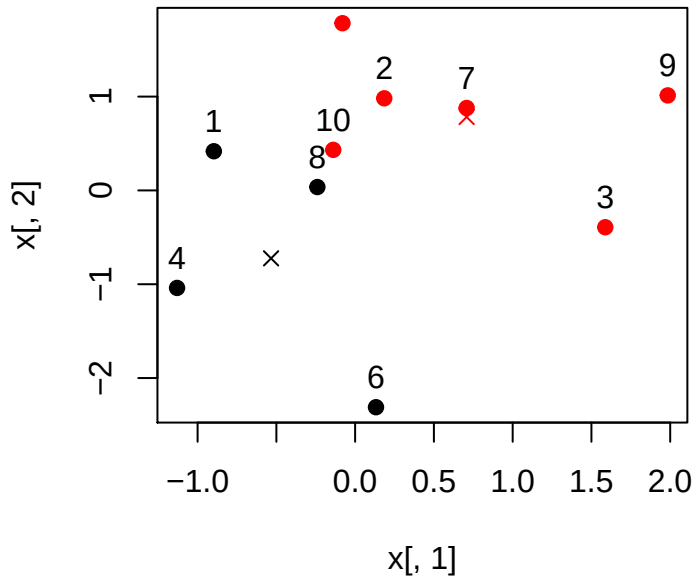
```
##  [1] 2 1 1 1 2 2 2 1 2 2
```



a. [2 points] We include those two updates in a function `kmeans_1step` defined below. And then we run for one iteration of those two steps and output the results.

```
# pairwise distance function
# cited from https://www.r-bloggers.com/pairwise-distances-in-r/
pdist <- function(A,B) {
  an = apply(A, 1, function(rvec) crossprod(rvec,rvec))
  bn = apply(B, 1, function(rvec) crossprod(rvec,rvec))

  m = nrow(A)
  n = nrow(B)

  tmp = matrix(rep(an, n), nrow=m)
  tmp = tmp +  matrix(rep(bn, m), nrow=m, byrow=TRUE)
  sqrt( tmp - 2 * tcrossprod(A,B) )
}

kmeans_1step=function(C,x,sync=FALSE,PLOT=FALSE){
  # given the cluster assignment, update the cluster means
  cntrs=sort(unique(C))
  K=length(cntrs)
  m=matrix(0,K,dim(x)[2])
  for(k in 1:K)m[k,]=colMeans(x[C==cntrs[k],])
  # given the cluster means, update the cluster assignment
  pdist_xm=pdist(x,m)
  C=apply(pdist_xm,1,which.min)
  if(sync){ # synchronize the results
    for(k in 1:K)m[k,]=colMeans(x[C==cntrs[k],])
```

```
  }
  if(PLOT){
    plot(x[,1], x[,2], col = C, pch = 19)
    points(m[,1],m[,2],col=cntrs,pch=4);text(x[,1], x[,2],1:length(C),pos=3)}
  return(list(cluster=C,centers=m,dist2cntr=pdist_xm))
}

# initialization
x <- cbind(x1,x2)
C <- C0

# now we run one iteration and output the result
upd <- kmeans_1step(C,x,TRUE,TRUE)
```




```
# the centers of the clusters
(m <- upd$centers)
```

```
##            [,1]       [,2]
## [1,] -0.5336420 -0.7243201
## [2,]  0.7076807  0.7824975
```

```
# the cluster assignment
(C <- upd$cluster)
```

```
##  [1] 1 2 2 1 2 1 2 1 2 2
```

Focus on codes. No unique answer.

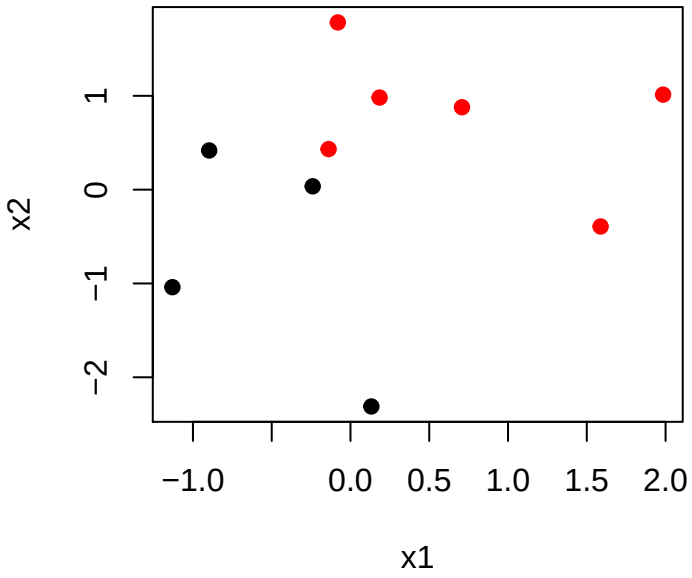
b. [2 points] Now we iterate the two-step updates in (a) until it does not change. Then we output the final result. In this instance, the algorithm converges in 1 iteration.

```
# iterate until the cluster assignment does not change
num_iter=0
while(1){
  C0 <- C
  upd <- kmeans_1step(C,x)
  C <- upd$cluster
  num_iter <- num_iter+1
```

```
  if(identical(C0,C))break
}
# output the final result
C
```

```
##  [1] 1 2 2 1 2 1 2 1 2 2
```

```
plot(x1, x2, col = C, pch = 19)
```



c. [2 points] One can directly calculate the within-cluster distance based on the cluster centers:

```
(withinss = sum(upd$dist2cntr[cbind(seq_along(C), C)]))
```

```
## [1] 10.0194
```

Or one can calculate the within-cluster distance based on the definition on page 5 of lecture 2.

```
cntrs=unique(C)
K=length(cntrs)
withinss1=0
for(k in 1:K)withinss1<-withinss1+sum(pdist(x[C==cntrs[k],],x[C==cntrs[k],]))/2
withinss1
```

```
## [1] 33.35176
```

Both are right. No unique answer.

d. [2 points] Now we generate another random initialization of `C` and repeat the steps. In this instance, the algorithm converges in 3 iterations.

```
# record the results in (b)
C1 <- C
set.seed(1)
# generate another initial value of the cluster assignments
(C = sample(1:2, n, replace = TRUE))
```

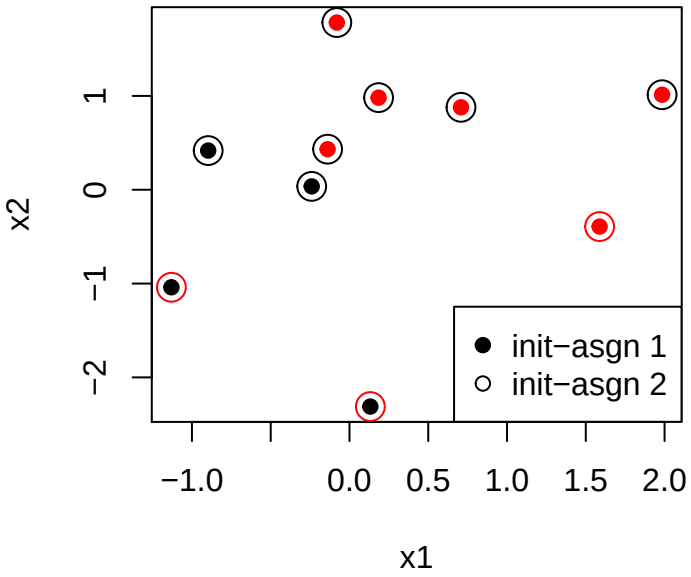
```
##  [1] 1 1 2 2 1 2 2 2 2 1
```

```
# repeat the kmeans updates in (a)
num_iter=0
while(1){
```

```
  C0 <- C
  upd <- kmeans_1step(C,x)
  (C <- upd$cluster)
  num_iter <- num_iter+1
  if(identical(C0,C))break
}
# new cluster assignment
C
```

```
##  [1] 1 1 2 2 1 2 1 1 1 1
```

```
# plot results on the same graph
plot(x1, x2, col = C1, pch = 19)
points(x1, x2, col = C, pch = 1, cex=2)
legend('bottomright',legend=c("init-asgn 1","init-asgn 2"),pch=c(19,1))
```



Note that kmeans algorithm depends on the initial cluster assignment. Therefore you may not get the same results for different initializations, as illustrated here. However, if you change the random seed number to be 10, you can get the same results as in (b).

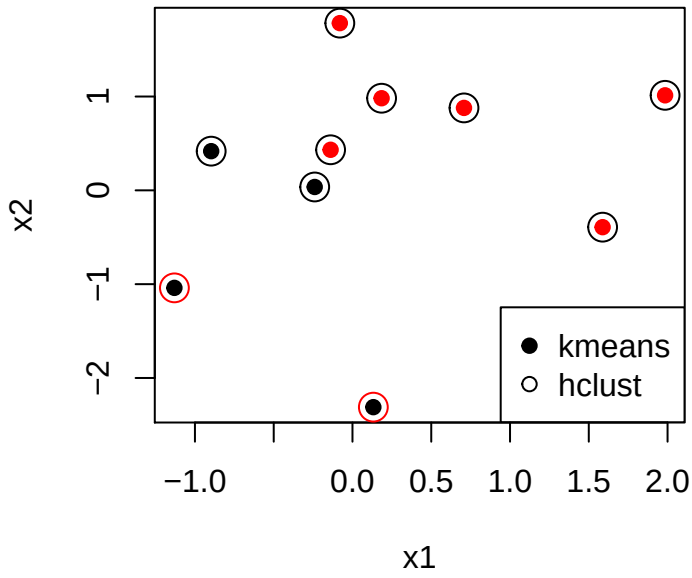
Again there is no unique answer.

e. [2 points] Now we use hierachical clustering and cut the level at $k = 2$.

```
# use hierachical clustering
x.hclust=hclust(dist(x))
(C2 = cutree(x.hclust,2))
```

```
##  [1] 1 1 1 2 1 2 1 1 1 1
```

```
# plot results on the same graph
plot(x1, x2, col = C1, pch = 19)
points(x1, x2, col = C2, pch = 1, cex=2)
legend('bottomright',legend=c("kmeans","hclust"),pch=c(19,1))
```



The results by two different algoirthms are different. Each algorithm emphasizes a different clustering feature of the data set.

Question 2 (bonus: k-means of a picture)

a. [1 points] We use `magick` library. Load and print an image from GNU website.

```
library(magick)
```

```
## Linking to ImageMagick 6.9.9.39
## Enabled features: cairo, fontconfig, freetype, lcms, pango, rsvg, webp
## Disabled features: fftw, ghostscript, x11
```

```
tiger <- image_read('http://jeroen.github.io/images/tiger.svg')
tiger <- image_convert(tiger,'png')
tiger <- image_scale(tiger, "300x300")
print(tiger)
```

```
##   format width height colorspace matte filesize density
## 1    PNG   300    300       sRGB  TRUE        0   96x96
```

b. [2 points] For $k = 2, 3, 5, 10$, perform k-means on the pixels and plot the resulting picture by replacing each pixel with their corresponding cluster mean.

We then extract raw bitmap array. Then perform k-means clustering on the pixels and replace each pixel with its cluster mean.

```
# obtain its bitmap values:
# for the last dimension, the first 3 values are RGB, the last coordinate is alpha
tiger_bitmap=as.integer(tiger[[1]])
tiger_imsz=dim(tiger_bitmap)
tiger_2d=matrix(tiger_bitmap[,,-tiger_imsz[3]],prod(tiger_imsz[1:2]),tiger_imsz[3]-1)
# kmeans clustering and plot images with pixel RGB replaced by cluster centers
#layout(matrix(1:4,2,2))
par(mfrow=c(1,4),oma=c(2,0,0,0),mar=c(2,0,1,1))
for(k in c(2,3,5,10)){
  tiger_kmeans=kmeans(tiger_2d,centers=k,nstart=10)
  tiger_colclst=array(tiger_kmeans$centers[tiger_kmeans$cluster,],dim=c(tiger_imsz[1:2],tiger_imsz[3]-1
  plot(as.raster(tiger_colclst,max=255L))
  title(main=paste('k=',k,sep=''))
}
```

k=2 k=3 k=5 k=10

1 point off if the color does not change.

c. [1 points] To determine the optimal number of cluster k, there are direct methods `elbow` and `average silhouette`; and statistical testing methods e.g. based on `gap statistics`. Read more here.

Anything cited or discussed works.