AML_HW3

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

Feb 10, 2018

Problem Statement

CIFAR-10 is a dataset of 32x32 images in 10 categories, collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. It is often used to evaluate machine learning algorithms. You can download this dataset from https://www.cs.toronto.edu/~kriz/cifar.html

Reference: https://stackoverflow.com/questions/32113942/importing-cifar-10-data-set-to-r

Implementation

For each category, compute the mean image and the first 20 principal components. Plot the error resulting from representing the images of each category using the first 20 principal components against the category.

```
# This is just for converting the binary files into csv to expedite data processing.
# Read binary file and convert to integer vectors
# [Necessary because reading directly as integer()
# reads first bit as signed otherwise]
#
# File format is 10000 records following the pattern:
# [label x 1][red x 1024][green x 1024][blue x 1024]
# NOT broken into rows, so need to be careful with "size" and "n"
#
# (See http://www.cs.toronto.edu/~kriz/cifar.html)
labels <- read.table("cifar-10-batches-bin/batches.meta.txt")
# renaming test_batch.bin to data_batch_6.bin


num_files  = 6
num_images = 10000
img_size = 1024

# Set to 10000 to retrieve all images per file to memory
# define the data frame to store the images & corresponding r,g & b value
data       = data.frame(matrix(nrow = num_images * 6, ncol = 1 + img_size * 3))


# Cycle through all 5 binary files
for (f in 1:num_files) {
  to.read <-
    file(paste("cifar-10-batches-bin/data_batch_", f, ".bin", sep = ""),
         "rb")
  for (i in 1:num_images) {
    index <- num_images * (f - 1) + i
    data[index, 1] <-
      readBin(to.read,
              integer(),
              size = 1,
              n = 1,
              endian = "big") + 1
    data[index, 2:img_size + 1] = as.integer(readBin(
      to.read,
      raw(),
      size = 1,
      n = img_size,
      endian = "big"
    ))
    data[index, img_size + 2:2 * img_size + 1] = as.integer(readBin(
      to.read,
      raw(),
      size = 1,
      n = img_size,
      endian = "big"
    ))
    data[index, 2 * img_size + 2:3 * img_size + 1] = as.integer(readBin(
      to.read,
      raw(),
      size = 1,
      n = img_size,
      endian = "big"
    ))
  }
  close(to.read)
  remove(f, i, index, to.read)
}



write.csv(data, file = 'data.csv')
save_data = data
```

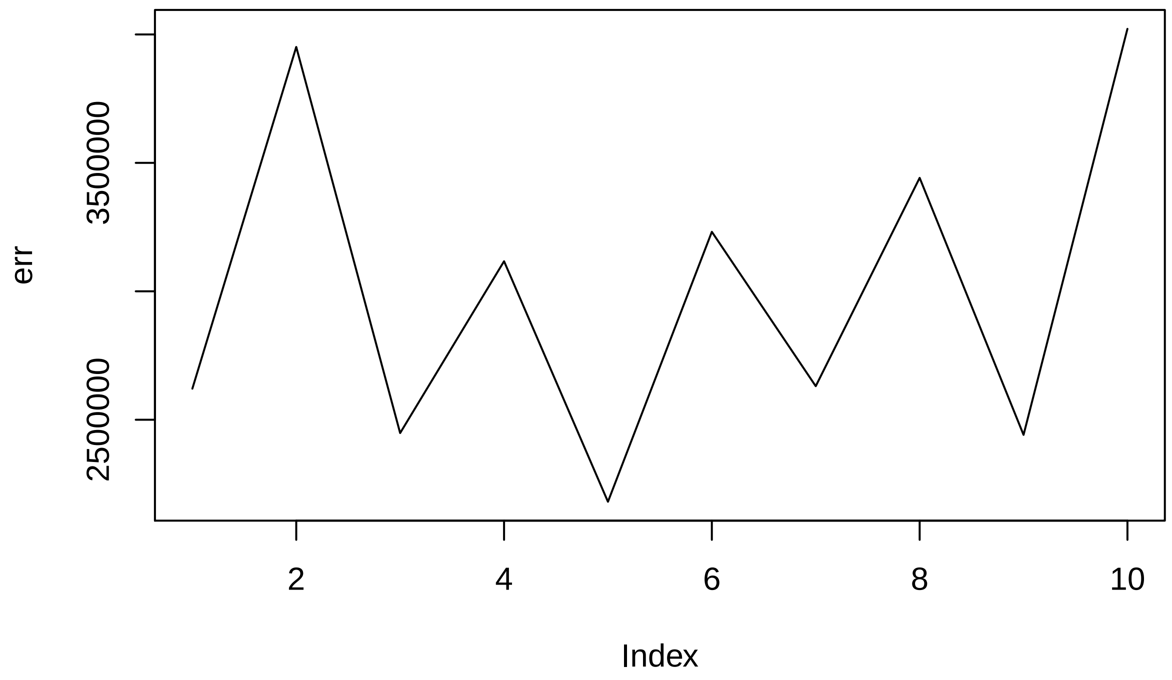
```
# Data preparation
labels <- read.table("cifar-10-batches-bin/batches.meta.txt")

data = read.csv("data.csv")
data = as.data.frame(data[, c(-1)])
```

```
# Get data frame by categories
cat_1 = data[data$X1 == 1, 2:ncol(data)]
cat_2 = data[data$X1 == 2, 2:ncol(data)]
cat_3 = data[data$X1 == 3, 2:ncol(data)]
cat_4 = data[data$X1 == 4, 2:ncol(data)]
cat_5 = data[data$X1 == 5, 2:ncol(data)]
cat_6 = data[data$X1 == 6, 2:ncol(data)]
cat_7 = data[data$X1 == 7, 2:ncol(data)]
cat_8 = data[data$X1 == 8, 2:ncol(data)]
cat_9 = data[data$X1 == 9, 2:ncol(data)]
cat_10 = data[data$X1 == 10, 2:ncol(data)]
```

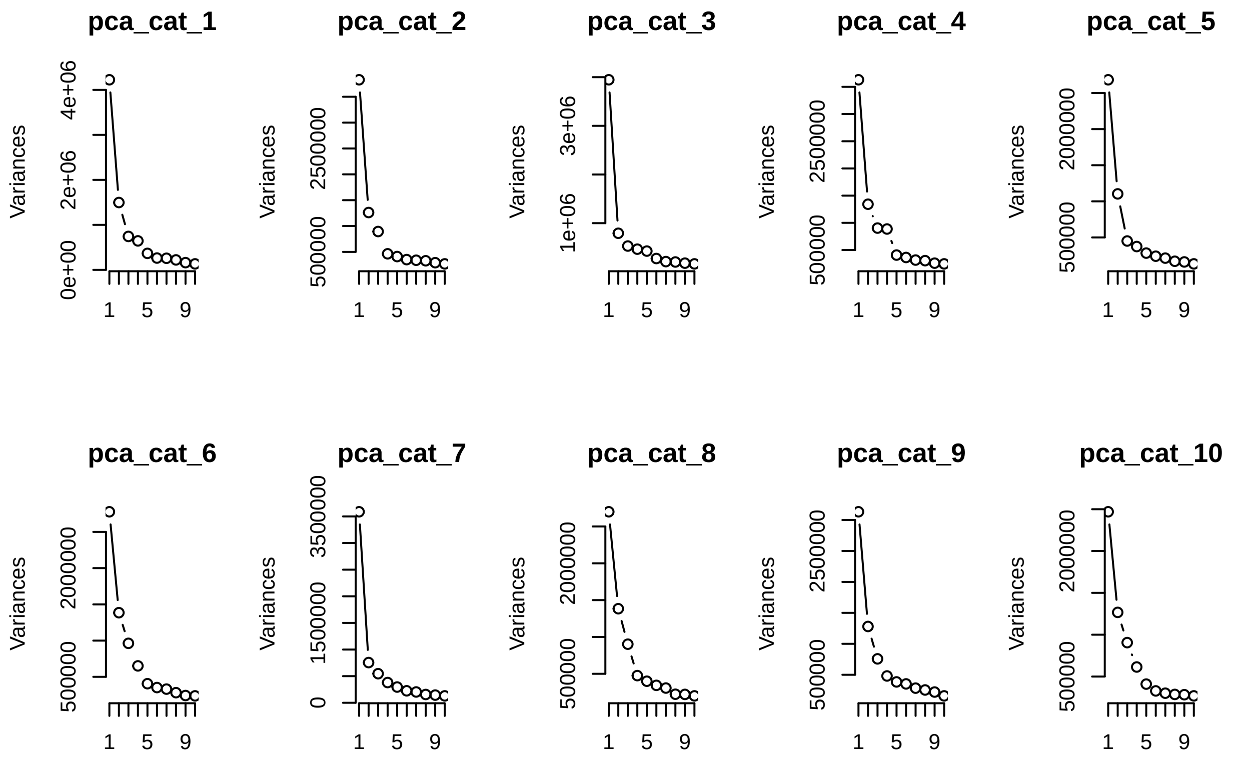
```r
# display a sample image
library(imager)
img = as.cimg(as.numeric(cat_10[4,]),
              x = 32,
              y = 32,
              cc = 3)
plot(img)
```

```r
# find the mean image for each category
mean_cat_1 = t(t(cat_1) - colMeans(cat_1))
mean_cat_2 = t(t(cat_2) - colMeans(cat_2))
mean_cat_3 = t(t(cat_3) - colMeans(cat_3))
mean_cat_4 = t(t(cat_4) - colMeans(cat_4))
mean_cat_5 = t(t(cat_5) - colMeans(cat_5))
mean_cat_6 = t(t(cat_6) - colMeans(cat_6))
mean_cat_7 = t(t(cat_7) - colMeans(cat_7))
mean_cat_8 = t(t(cat_8) - colMeans(cat_8))
mean_cat_9 = t(t(cat_9) - colMeans(cat_9))
mean_cat_10 = t(t(cat_10) - colMeans(cat_10))

# Compute errors for the 1st 20 PCAs for all Categories.
pca_cat_1 = prcomp(mean_cat_1, center = TRUE, scale. = FALSE)
err_pca_cat_1 = sum((pca_cat_1$sdev[21:length(pca_cat_1$sdev)]) ^ 2)

pca_cat_2 = prcomp(mean_cat_2, center = TRUE, scale. = FALSE)
err_pca_cat_2 = sum((pca_cat_2$sdev[21:length(pca_cat_2$sdev)]) ^ 2)
```

```r
pca_cat_3 = prcomp(mean_cat_3, center = TRUE, scale. = FALSE)
err_pca_cat_3 = sum((pca_cat_3$sdev[21:length(pca_cat_3$sdev)]) ^ 2)

pca_cat_4 = prcomp(mean_cat_4, center = TRUE, scale. = FALSE)
err_pca_cat_4 = sum((pca_cat_4$sdev[21:length(pca_cat_4$sdev)]) ^ 2)

pca_cat_5 = prcomp(mean_cat_5, center = TRUE, scale. = FALSE)
err_pca_cat_5 = sum((pca_cat_5$sdev[21:length(pca_cat_5$sdev)]) ^ 2)

pca_cat_6 = prcomp(mean_cat_6, center = TRUE, scale. = FALSE)
err_pca_cat_6 = sum((pca_cat_6$sdev[21:length(pca_cat_6$sdev)]) ^ 2)

pca_cat_7 = prcomp(mean_cat_7, center = TRUE, scale. = FALSE)
err_pca_cat_7 = sum((pca_cat_7$sdev[21:length(pca_cat_7$sdev)]) ^ 2)

pca_cat_8 = prcomp(mean_cat_8, center = TRUE, scale. = FALSE)
err_pca_cat_8 = sum((pca_cat_8$sdev[21:length(pca_cat_8$sdev)]) ^ 2)

pca_cat_9 = prcomp(mean_cat_9, center = TRUE, scale. = FALSE)
err_pca_cat_9 = sum((pca_cat_9$sdev[21:length(pca_cat_9$sdev)]) ^ 2)

pca_cat_10 = prcomp(mean_cat_10, center = TRUE, scale. = FALSE)
err_pca_cat_10 = sum((pca_cat_10$sdev[21:length(pca_cat_10$sdev)]) ^ 2)

# Storing the errors into a variable
err = c(
  err_pca_cat_1,
  err_pca_cat_2,
  err_pca_cat_3,
  err_pca_cat_4,
  err_pca_cat_5,
  err_pca_cat_6,
  err_pca_cat_7,
  err_pca_cat_8,
  err_pca_cat_9,
  err_pca_cat_10
)


# Error plot
plot(err, type = "l")
```

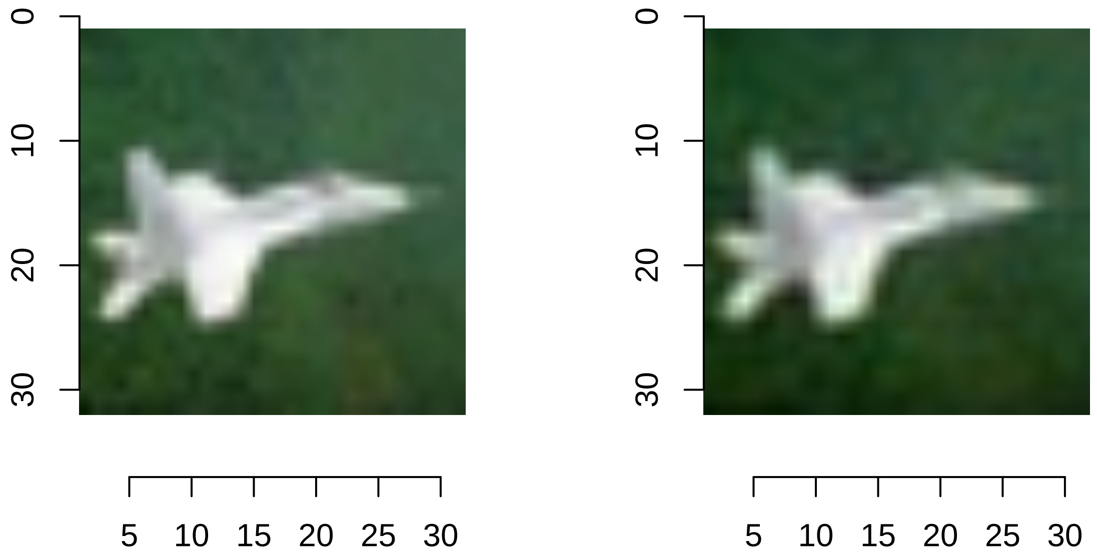


```r
# Plots of PCAs of each category
par(mfrow = c(2, 5))
plot(pca_cat_1, type = "l")
plot(pca_cat_2, type = "l")
plot(pca_cat_3, type = "l")
plot(pca_cat_4, type = "l")
plot(pca_cat_5, type = "l")
plot(pca_cat_6, type = "l")
plot(pca_cat_7, type = "l")
plot(pca_cat_8, type = "l")
plot(pca_cat_9, type = "l")
plot(pca_cat_10, type = "l")
```



Sample reconstruction of the image using the first 500 PCAs.

```
# Org & approximate image using PCA
m       = mean_cat_1
pca     = pca_cat_1
num_pca = 500
idx     = 5

p = ((m) %*% pca$rotation[, 1:num_pca])
xhat = p %*% t(pca$rotation[, 1:num_pca])

org = t(t(xhat) + colMeans(cat_1))

par(mfrow=c(1,2))
library(imager)
img = as.cimg(as.numeric(cat_1[idx,]),
              x = 32,
              y = 32,
              cc = 3)
plot(img)

img = as.cimg(as.numeric(org[idx,]),
              x = 32,
              y = 32,
               cc = 3)
plot(img)
```



Compute the distances between mean images for each pair of classes. Use principal coordinate analysis to make a 2D map of the means of each categories.

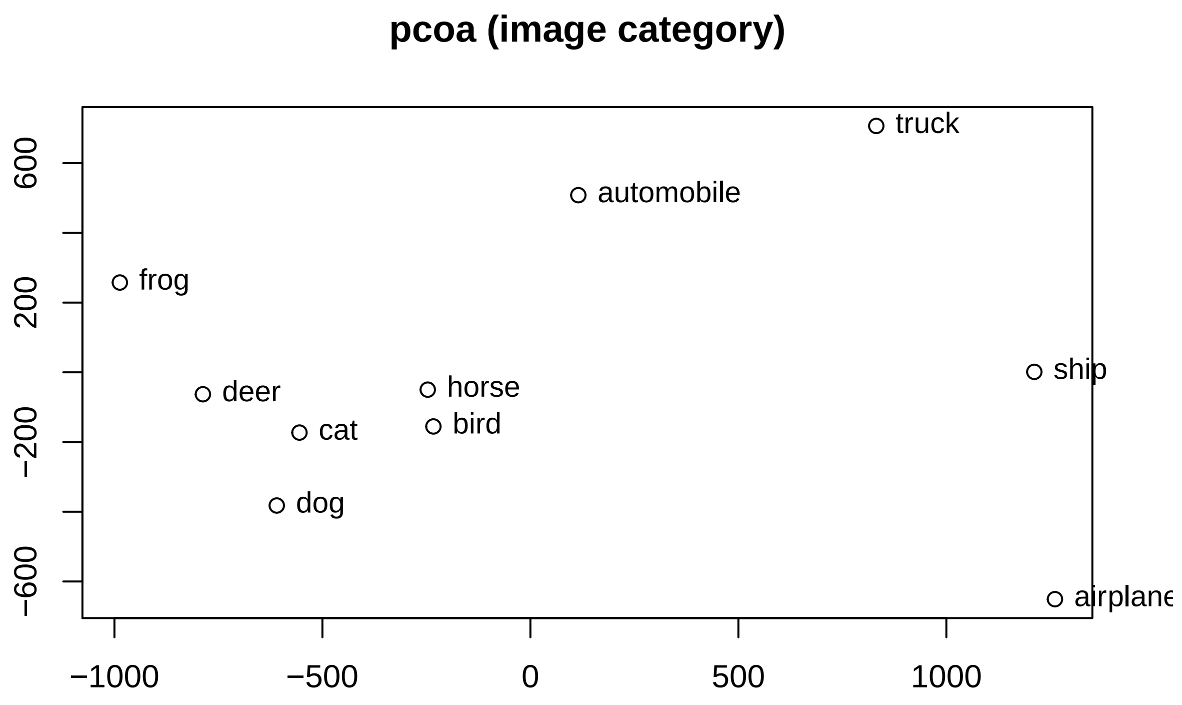
For this exercise, compute distances by thinking of the images as vectors.

```
library(ape)
m = rbind(
  colMeans(cat_1),
  colMeans(cat_2),
  colMeans(cat_3),
  colMeans(cat_4),
  colMeans(cat_5),
  colMeans(cat_6),
  colMeans(cat_7),
  colMeans(cat_8),
  colMeans(cat_9),
  colMeans(cat_10)
)

# Creating Euclidean distance matix
d = dist(m)
```

```r
# Diplay the 10 x 10 distance matrix
d
```

```
##            1         2         3         4         5         6         7
## 2  1683.6354                                                            
## 3  1605.0243  886.2367                                                  
## 4  1905.5353 1027.6498  517.3115                                        
## 5  2148.7634 1143.0814  601.2503  469.7917                              
## 6  1965.2215 1216.0794  701.4682  412.1817  617.6971                    
## 7  2445.6797 1191.1920  913.7475  677.4920  460.5109  828.5811          
## 8  1663.6459  950.7861  418.2763  596.3767  684.3469  843.6721  948.7040
## 9   945.5411 1303.4665 1557.7150 1851.2145 2065.6217 1897.5918 2249.1998
## 10 1449.0949  949.9958 1416.6747 1676.4679 1830.7409 1880.2438 1913.2409
##            8         9
## 2                     
## 3                     
## 4                     
## 5                     
## 6                     
## 7                     
## 8                     
## 9  1660.2681          
## 10 1347.3341 1066.9416
```

```r
pcoa1 = cmdscale(d, k = 2)

plot(
  pcoa1[, 1],
  pcoa1[, 2],
  type = "p",
  xlab = "",
  ylab = "",
  axes = TRUE,
  main = "pcoa (image category)"
)

text(
  pcoa1[, 1],
  pcoa1[, 2],
  pos = 4,
  labels = labels[, 1],
  cex = 0.9,
  pch = 1,
  lwd = 1,
  xpd = TRUE
)
```



Here is another measure of the similarity of two classes. For class A and class B, define E(A | B) to be the average error obtained by representing all the images of class A using the mean of class A and the first 20 principal components of class B. Now define the similarity between classes to be (1/2)(E(A | B) + E(B | A)). If A and B are very similar, then this error should be small, because A's principal components should be good at representing B. But if they are very different, then A's principal components should represent B poorly. In turn, the similarity measure should be big. Use principal coordinate analysis to make a 2D map of the classes. Compare this map to the map in the previous exercise? are they different? why?

```
# calculate the similarity measure given by (1/2)(E(A | B) + E(B | A))

get_similarity = function(mean_list, pca_list, i, j)
{
  m = mean_list[[i]]
  pca = pca_list[[j]]

  p = ((m) %*% pca$rotation[, 1:20])

  xhat = p %*% t(pca$rotation[, 1:20])

  s = m - xhat
  (e1 = sum(s ^ 2) / nrow(s))


  m = mean_list[[j]]
  pca = pca_list[[i]]

  p = ((m) %*% pca$rotation[, 1:20])

  xhat = p %*% t(pca$rotation[, 1:20])


  s = m - xhat


  (e2 = sum(s ^ 2) / nrow(s))

  (e1 + e2) / 2
}
```

```
# Storing means in a variable
mean_list = list(
  mean_cat_1,
  mean_cat_2,
  mean_cat_3,
  mean_cat_4,
  mean_cat_5,
  mean_cat_6,
  mean_cat_7,
  mean_cat_8,
  mean_cat_9,
  mean_cat_10
  )

# storing Pcas in a variable
pca_list = list(
  pca_cat_1,
  pca_cat_2,
  pca_cat_3,
  pca_cat_4,
  pca_cat_5,
  pca_cat_6,
  pca_cat_7,
  pca_cat_8,
  pca_cat_9,
  pca_cat_10
  )

# Creating similarity matrix
sim = matrix(nrow = 10, ncol = 10)
for (i in 1:10)
{
  for(j in 1:10)
  {
```

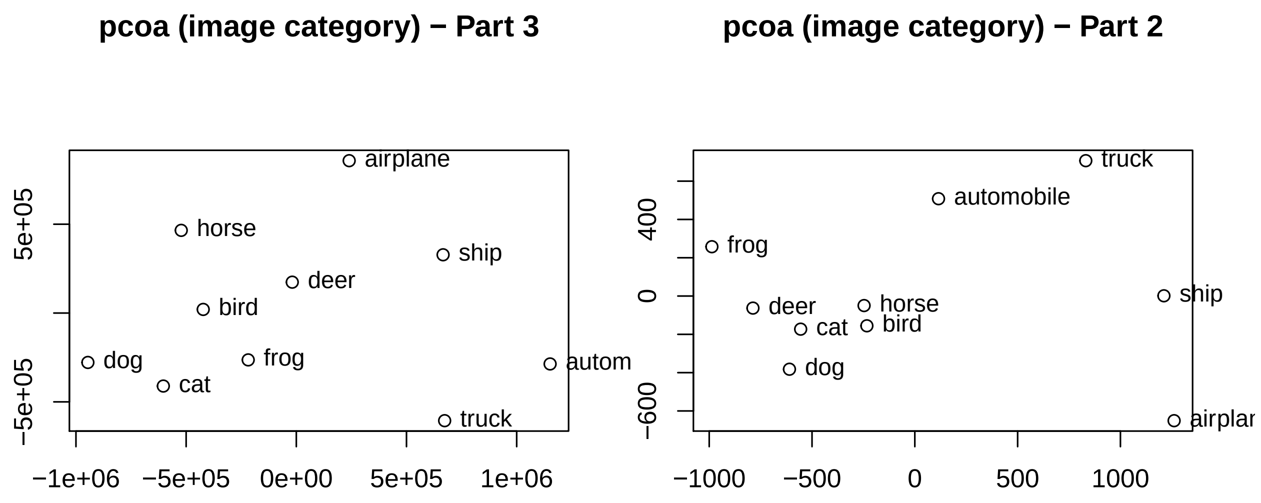
```r
    sim[i, j] = get_similarity(mean_list, pca_list, i, j)
  }
}


pcoa2 = cmdscale(sim, k = 2)

(r = get_similarity(mean_list, pca_list, 1, 10))
```

```
## [1] 3794218
```

```r
# Diplay the 10 x 10 distance matrix
sim
```

```
##          [,1]    [,2]    [,3]    [,4]    [,5]    [,6]    [,7]    [,8]
##  [1,] 2620501 3723676 2794365 3306932 2554925 3399592 2950223 3394717
##  [2,] 3723676 3950676 3699977 4039273 3450549 4211172 3685206 4230300
##  [3,] 2794365 3699977 2447698 2939377 2423654 2967088 2685785 3199991
##  [4,] 3306932 4039273 2939377 3116479 2889583 3262798 3028700 3546316
##  [5,] 2554925 3450549 2423654 2889583 2180391 2954958 2553877 3041230
##  [6,] 3399592 4211172 2967088 3262798 2954958 3231113 3100441 3610865
##  [7,] 2950223 3685206 2685785 3028700 2553877 3100441 2630244 3341395
##  [8,] 3394717 4230300 3199991 3546316 3041230 3610865 3341395 3441091
##  [9,] 2715857 3489125 2813363 3202341 2543500 3386717 2873799 3386206
## [10,] 3794218 4134930 3634776 3897344 3441359 4075472 3648283 4139677
##          [,9]   [,10]
##  [1,] 2715857 3794218
##  [2,] 3489125 4134930
##  [3,] 2813363 3634776
##  [4,] 3202341 3897344
##  [5,] 2543500 3441359
##  [6,] 3386717 4075472
##  [7,] 2873799 3648283
##  [8,] 3386206 4139677
##  [9,] 2440635 3541624
## [10,] 3541624 4021094
```

```r
# Plotting PCOA for part 3
par(mfrow=c(1,2), mar=c(8,2,8,2))

plot(
  pcoa2[, 1],
  pcoa2[, 2],
  type = "p",
  xlab = "",
  ylab = "",
  axes = TRUE,
  main = "pcoa (image category) - Part 3"
  )

text(pcoa2[, 1],
     pcoa2[, 2],
     pos = 4,
     labels = labels[, 1],
     cex = 0.9,
     pch= 1,
     lwd=1,
     xpd = TRUE)



# Plot from Part 2
plot(
  pcoa1[, 1],
  pcoa1[, 2],
  type = "p",
  xlab = "",
  ylab = "",
  axes = TRUE,
  main = "pcoa (image category) - Part 2"
)

text(
  pcoa1[, 1],
  pcoa1[, 2],
  pos = 4,
  labels = labels[, 1],
  cex = 0.9,
  pch = 1,
  lwd = 1,
  xpd = TRUE
)
```



- This is a computation of dis-similarity/distance (not similarity) since the values are low when they are similar instead of being high. This is why the plot looks similar to part 2. We can see the relative distance between different image categories are similar between the two plots (for ex: cat & dog, Truck & Aeroplane). They are though not exactly identical because the distance/dis-similarity is calculated differently in part 2 vs part 3. Also the images are kind of rotated for part 2 & part 3.