Stat 432 Homework 12

Giang Le

Assigned: Nov 15, 2021; Due: 11:59 PM CT, Nov 30, 2021

Contents

Question 1: K-Means

In our lecture, there is an example of clustering pixels in an image. For this question, we will replicate that procedure with your favorite image. To complete this question, perform the following steps:

- [10 Points] Pick your favorite image for this question. Plot the original image. Note that for computational concern, you probably want to avoid an extremely large image file.

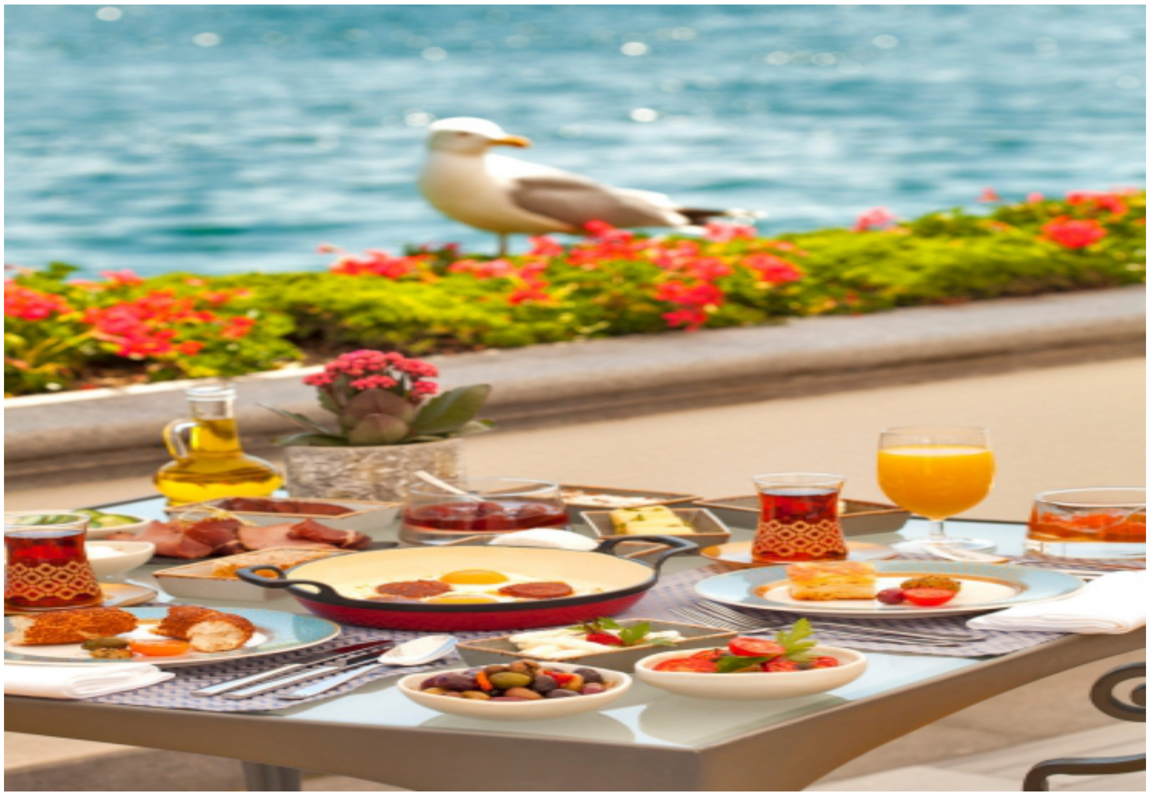
```r
## First I load my favorite image
install.packages("jpeg", repos='http://cran.us.r-project.org')
```

```
##
## The downloaded binary packages are in
##  /var/folders/9c/3_mgdyf12z7dvb8rt4d60nt80000gn/T//RtmpnWsK8p/downloaded_packages
```

```r
library(jpeg)
img<-readJPEG("/Users/gianghale/Desktop/fall-2021/stat-432/image.jpeg")

# generate a blank image
par(mar=rep(0.2, 4))

# Plot the image using rasterImage
plot(c(0, 100), c(0, 100), xaxt = 'n', yaxt = 'n',
        bty = 'n', pch = '', ylab = '', xlab = '')
rasterImage(img, 0, 0, 100, 100)
```



```
dim(img)
```

```
## [1] 500 400   3
```

- [10 Points] Report the following information of your data:
  - Dimension of the original image
  - Dimension of the data once you transform the image to a version that you could apply k-means
  - Total variations of your data

According to the code output below, the height of the original image is 500 pixels and the width of the original image is 400 pixels. The dimension of the transformed data is 200000 x3. The variation of r values from the mean is 80480.67 The variation of g values from the mean is 66516.56 The variation of b values from the mean is 37405.99

```
# Checking the dimension of the image
dim(img)
```

```
## [1] 500 400   3
```

```
# this apply function applies vectorization to each layer (r/g/b) of the image.
img_expand = apply(img, 3, c)

# and now we have the desired data matrix
dim(img_expand)
```

```
## [1] 200000      3
```

```
# Total variation of the data
means <- colMeans(img_expand)
means
```

```
## [1] 0.6780681 0.6242481 0.5160849
```

```
img_new <- setNames(as.data.frame((img_expand - means)^2), c("r", "g", "b"))
variation_r <- sum((img_new$r - means[1])^2)
variation_g <- sum((img_new$g - means[2])^2)
variation_b <- sum((img_new$b - means[3])^2)

variation_r
```

```
## [1] 80480.67
```

```
variation_g
```

```
## [1] 66516.56
```

```
variation_b
```

```
## [1] 37405.99
```

- [25 Points] Apply k-means to your data. Choose three unique k values to report the following result:
  - What is the within-cluster variance?
  - What are the cluster means?
  - Plot the image with each pixel replaced by its corresponding cluster mean

Use k = 5 (answers about variance and cluster means can be found in the code chunk.)

```
set.seed(432)
# choose value = 5
kmeanfit <- kmeans(img_expand, 5)

# the within cluster variance
print("within cluster variance")
```

```
## [1] "within cluster variance"
```

```
kmeanfit$withinss
```

```
## [1]  855.7218  790.5235 3826.6861  569.3825 1308.8383
```

```
# The cluster means
print("cluster means")
```

```
## [1] "cluster means"
```

```
kmeanfit$centers
```

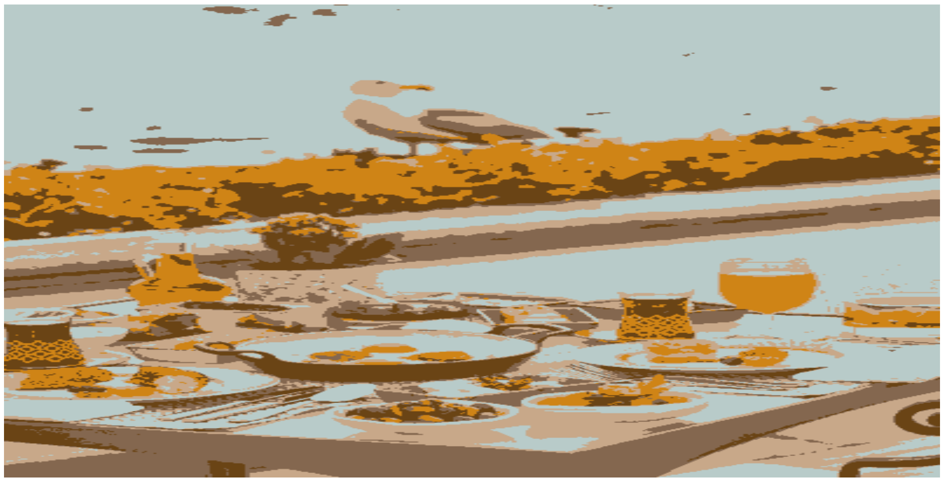
```
##        [,1]      [,2]       [,3]
## 1 0.7824658 0.6590773 0.53826643
## 2 0.4145895 0.2649142 0.08411499
## 3 0.7211284 0.7974984 0.78790124
## 4 0.5187149 0.4048987 0.30913240
## 5 0.8127339 0.5180558 0.08675361
```

```
# to produce the new graph, we simply replicate the cluster mean
# for all observations in the same cluster
new_img_expand = kmeanfit$centers[kmeanfit$cluster, ]
new_img = img
new_img[, , 1] = matrix(new_img_expand[,1], 500, 400)
new_img[, , 2] = matrix(new_img_expand[,2], 500, 400)
new_img[, , 3] = matrix(new_img_expand[,3], 500, 400)

# plot the new image
```

```
plot(c(0, 500), c(0, 400), xaxt = 'n', yaxt = 'n', bty = 'n',
       pch = '', ylab = '', xlab = '')

rasterImage(new_img, 0, 0, 500, 400)
```



Use k = 10 (answers about variance and cluster means can be found in the code chunk.)

```
set.seed(432)
# choose value = 10
kmeanfit <- kmeans(img_expand, 10)

# the within cluster variance
print("within cluster variance")
```

```
## [1] "within cluster variance"
```

```
kmeanfit$withinss
```

```
##  [1] 203.4624 364.6383 260.0683 260.8500 147.1165 361.7269 211.0988 252.1232
##  [9] 616.8001 358.8716
```

```
# The cluster means
print("cluster means")
```

```
## [1] "cluster means"
```

```
kmeanfit$centers
```

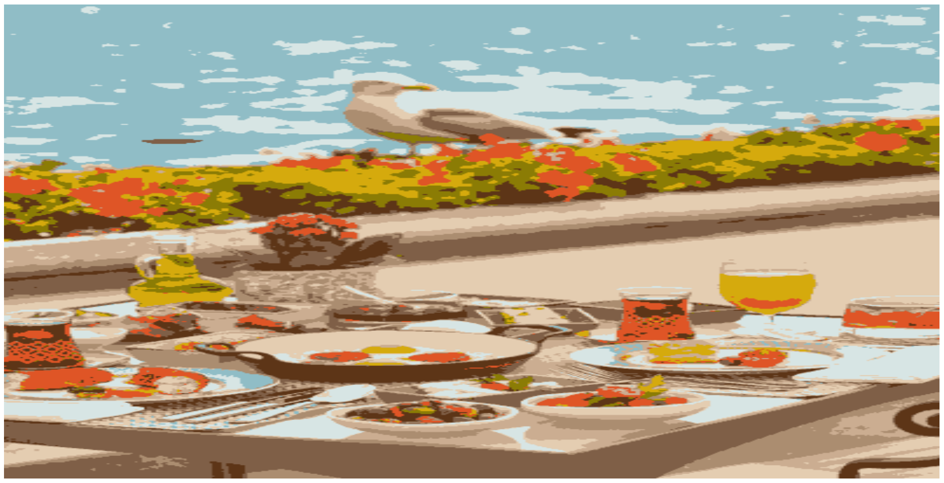
```
##         [,1]      [,2]       [,3]
## 1  0.6723696 0.5545764 0.43943983
## 2  0.8728248 0.3327254 0.14873877
## 3  0.7954794 0.6774724 0.56754412
## 4  0.4985244 0.3726896 0.27669844
## 5  0.5460196 0.4865138 0.01958030
## 6  0.8442459 0.8968626 0.89388283
## 7  0.8958480 0.8032947 0.69554223
## 8  0.8339975 0.6654545 0.05139966
## 9  0.5638425 0.7418970 0.77636392
## 10 0.3638726 0.2088434 0.09038110
```

```
# to produce the new graph, we simply replicate the cluster mean
# for all observations in the same cluster
```

```
new_img_expand = kmeanfit$centers[kmeanfit$cluster, ]
new_img = img
new_img[, , 1] = matrix(new_img_expand[,1], 500, 400)
new_img[, , 2] = matrix(new_img_expand[,2], 500, 400)
new_img[, , 3] = matrix(new_img_expand[,3], 500, 400)

# plot the new image
plot(c(0, 500), c(0, 400), xaxt = 'n', yaxt = 'n', bty = 'n',
       pch = '', ylab = '', xlab = '')

rasterImage(new_img, 0, 0, 500, 400)
```



Use k = 25 (answers about variance and cluster means can be found in the code chunk.)

```
set.seed(432)
# choose value = 25
kmeanfit <- kmeans(img_expand, 25)

# the within cluster variance
print("within cluster variance")
```

```
## [1] "within cluster variance"
```

```
kmeanfit$withinss
```

```
##  [1]  72.80888  86.25547  73.86611  64.97486  64.66541  16.10999  61.91153
##  [8]  99.97256  14.07507 105.44710  55.57529  13.37452  31.40797  65.90808
## [15]  12.25570  58.38859  56.74061  13.16350  71.07769  13.41949  59.23446
## [22]  32.45451  62.84779  16.08052  13.36083
```

```
# The cluster means
print("cluster means")
```

```
## [1] "cluster means"
```

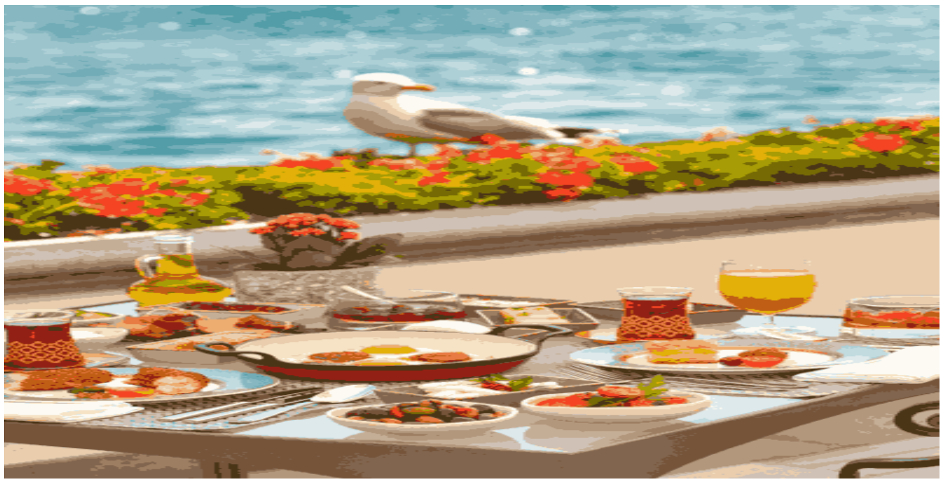
```
kmeanfit$centers
```

```
##          [,1]      [,2]       [,3]
## 1  0.6990565 0.5937359 0.48483595
## 2  0.7539400 0.3701013 0.08334350
## 3  0.9089017 0.4770738 0.26509043
## 4  0.6169612 0.4944280 0.38240254
## 5  0.6529496 0.6112382 0.02139511
## 6  0.6099235 0.7670169 0.79892322
## 7  0.9131322 0.8038287 0.67915298
## 8  0.8864573 0.6908820 0.03292191
## 9  0.5307237 0.7201923 0.75853996
## 10 0.2906379 0.2065425 0.09128589
## 11 0.9305422 0.6773217 0.43991538
## 12 0.4441356 0.6735545 0.71618544
## 13 0.9649314 0.8888167 0.77078726
## 14 0.5758398 0.1244568 0.07683820
## 15 0.6778480 0.8118765 0.83745068
## 16 0.5115160 0.3953683 0.30573056
## 17 0.7714106 0.6820165 0.58967393
## 18 0.7573674 0.8589901 0.87536184
## 19 0.4579881 0.3301066 0.22933152
## 20 0.8668835 0.9127718 0.92032931
## 21 0.4573945 0.4162018 0.01523977
## 22 0.8102812 0.7568263 0.70906219
## 23 0.9577024 0.2684481 0.13570504
## 24 0.3207576 0.6167025 0.65611172
## 25 0.9890208 0.9784419 0.94736203
```

```r
# to produce the new graph, we simply replicate the cluster mean
# for all observations in the same cluster
new_img_expand = kmeanfit$centers[kmeanfit$cluster, ]
new_img = img
new_img[, , 1] = matrix(new_img_expand[,1], 500, 400)
new_img[, , 2] = matrix(new_img_expand[,2], 500, 400)
new_img[, , 3] = matrix(new_img_expand[,3], 500, 400)

# plot the new image
plot(c(0, 500), c(0, 400), xaxt = 'n', yaxt = 'n', bty = 'n',
       pch = '', ylab = '', xlab = '')

rasterImage(new_img, 0, 0, 500, 400)
```



Question 2: Hierarchical Clustering

The same type of image compression approach can be done using hierarchical clustering. Using the data that you prepared for the k-means algorithm, to perform hierarchical clustering. However, instead of using the euclidean distance with `dist()` function, you need to provide the clustering algorithm a different distance matrix $D_{n \times n}$. The (i, j)th element in this matrix represents the distance between observations i and j, defined as

$$d(\mathbf{x}_i, \mathbf{x}_k) = \|\mathbf{x}_i - \mathbf{x}_j\|_1$$

To be able to use this matrix in the `hclust()` function, you need to convert the matrix into a dist object, using the `as.dist()` function. For more details, read the documentation here.

```
 # Preparing the data.
 # Use a smaller image in order to avoid vcector memory error.

img1<-readJPEG("/Users/gianghale/Desktop/fall-2021/stat-432/download.jpeg")

# generate a blank image
par(mar=rep(0.2, 4))

# Choosing only 100 pixels to avoid memory error.

img1 <- img1[1:100,1:100,]

#Plot the image
plot(c(0, 100), c(0, 100), xaxt = 'n', yaxt = 'n',
        bty = 'n', pch = '', ylab = '', xlab = '')
rasterImage(img1, 0, 0, 100, 100)
```

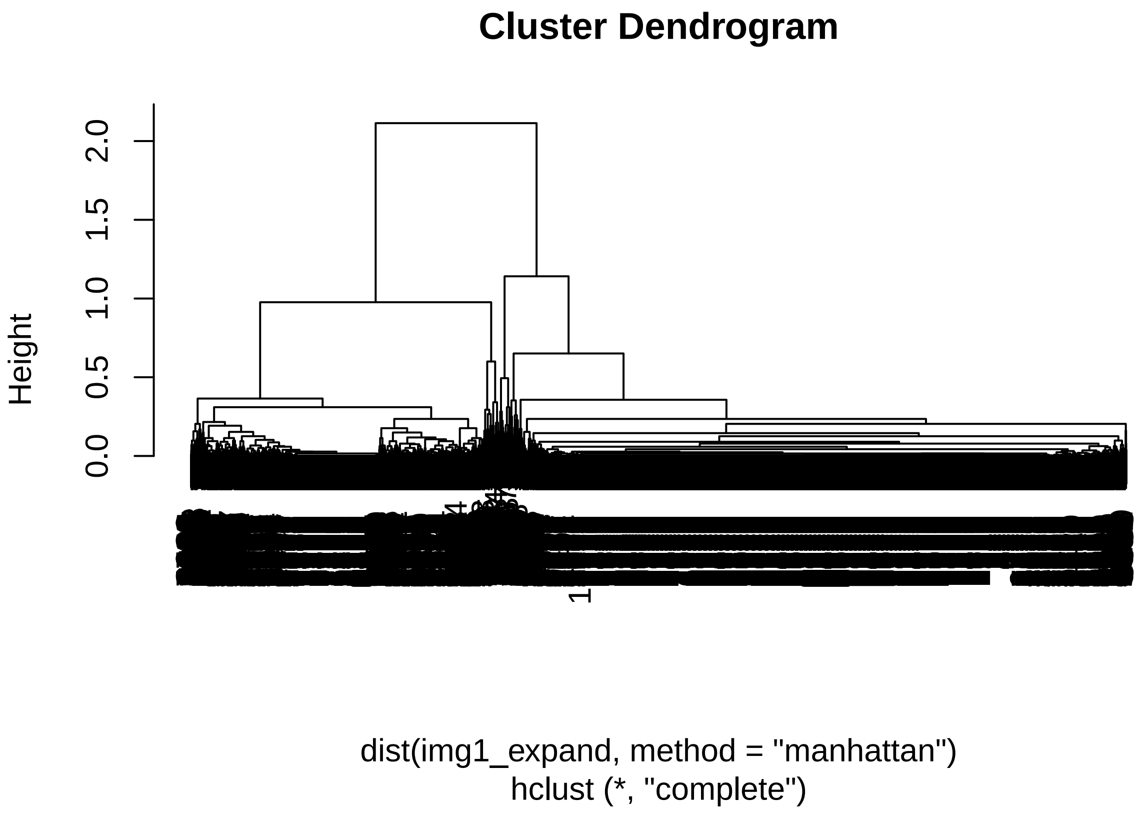
```r
img1_expand = apply(img1, 3, c)
dim(img1_expand)
```

```
## [1] 10000     3
```

```r
# Complete linkage
dim(img1_expand)
```

```
## [1] 10000     3
```

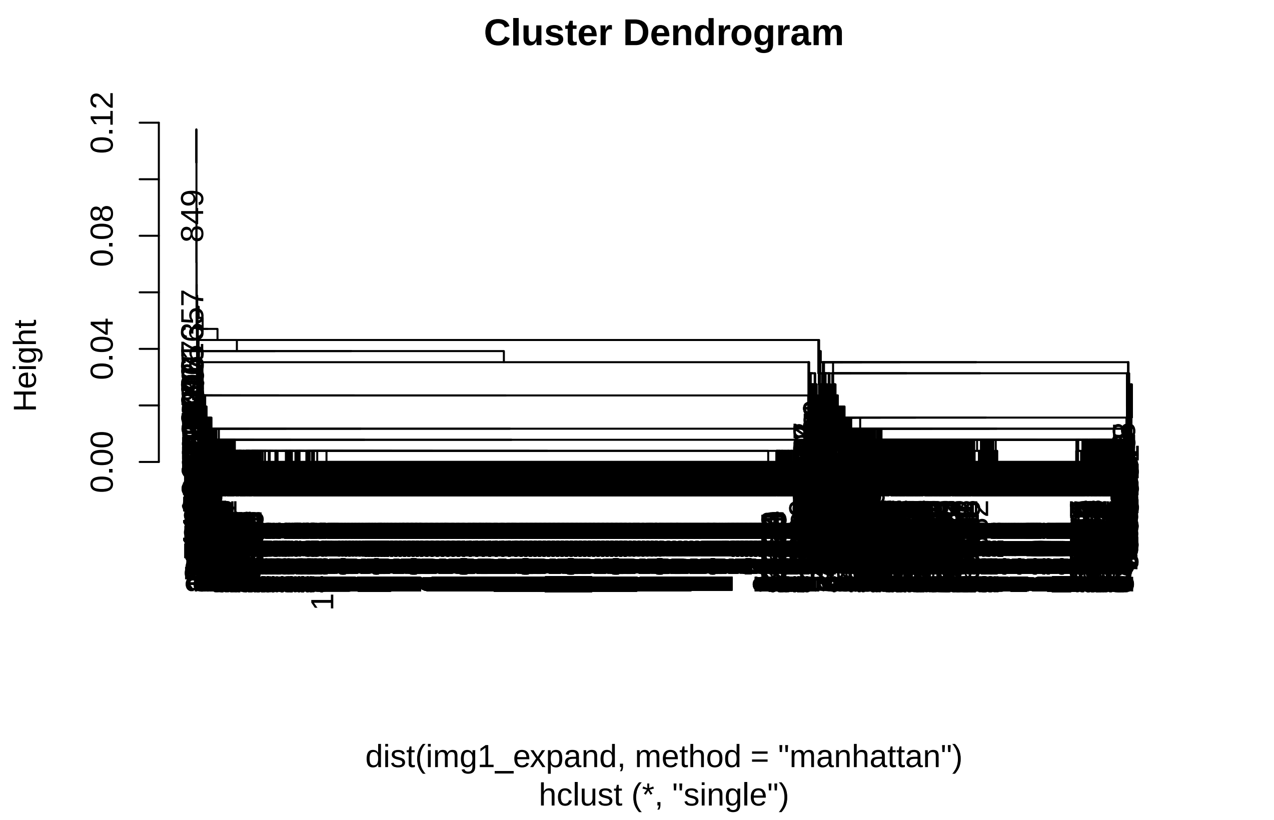
```r
complete <- hclust(dist(img1_expand, method = "manhattan"), method = "complete")
plot(complete)
```



```r
# Single linkage
dim(img1_expand)
```

```
## [1] 10000     3
```

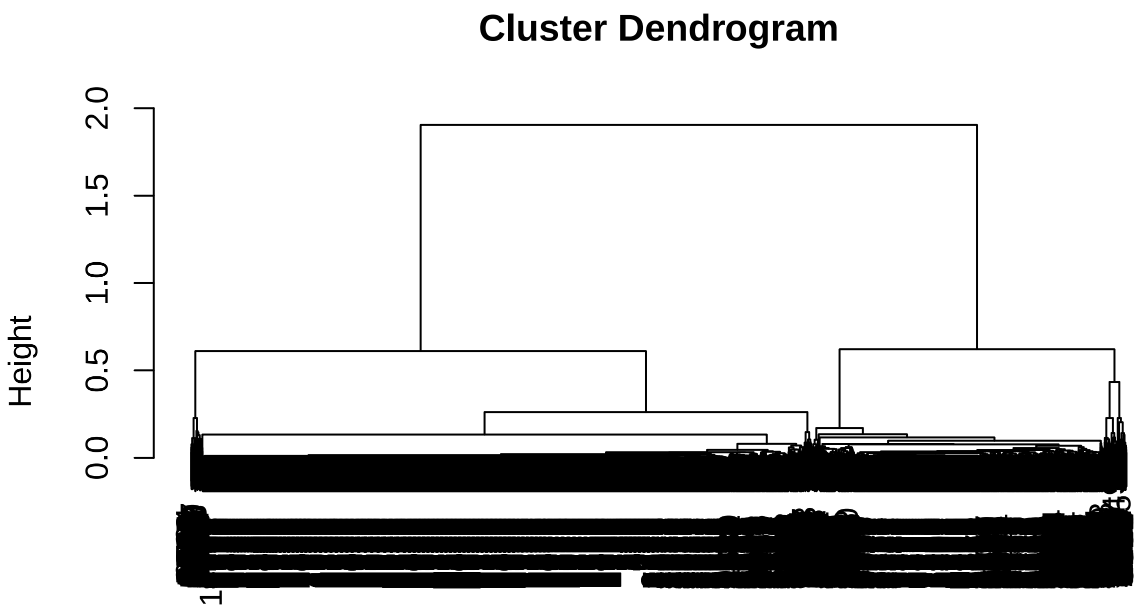
```r
sg <- hclust(dist(img1_expand, method = "manhattan"), method = "single")
plot(sg)
```



```
# Average linkage
dim(img1_expand)
```

```
## [1] 10000     3
```

```
ave <- hclust(dist(img1_expand, method = "manhattan"), method = "ave")
plot(ave)
```



dist(img1_expand, method = "manhattan")
hclust (*, "average")

Once you have all the component to perform the hierarchical clustering, do the following:

- [15 Points] Try both complete, single and average linkage. Provide a plot of the dendrogram for both methods.
- [10 Points] Based on what you have, pick one final clustering result. You need to explain the rational for your choice.
- [10 Points] Based on your final choice, calculate the cluster centers using the mean of all pixels in the cluster. Then replace all pixels in each cluster with their corresponding cluster mean. This step is similar to the k-means question.
- [10 Points] Plot this new image.

Based on what you have, pick one final clustering result. You need to explain the rational for your choice. -> Looking at the plots of both the complete linkage and the average linkage, we can see that two clusters is our best choice. The reason is because if we look at the height of the split, the height before merging the two last groups is small but if we merge them, the dendrogram's height is quite large. So we can stop at two clusters. I perform clustering using k-means below with k = 2 and plotted the new image below.

```
set.seed(432)
# choose value = 2
kmeanfit <- kmeans(img1_expand, 2)

# The cluster means
print("cluster means")
```

```
## [1] "cluster means"
```

```
kmeanfit$centers
```

```
##         [,1]      [,2]      [,3]
## 1 0.08591672 0.3562926 0.6255952
## 2 0.98866592 0.9924638 0.9940001
```

```r
# to produce the new graph, we simply replicate the cluster mean
# for all observations in the same cluster
new_img_expand = kmeanfit$centers[kmeanfit$cluster, ]
new_img = img1
new_img[, , 1] = matrix(new_img_expand[,1], 100, 100)
new_img[, , 2] = matrix(new_img_expand[,2], 100, 100)
new_img[, , 3] = matrix(new_img_expand[,3], 100, 100)

# plot the new image
plot(c(0, 100), c(0, 100), xaxt = 'n', yaxt = 'n', bty = 'n',
       pch = '', ylab = '', xlab = '')

rasterImage(new_img, 0, 0, 100, 100)
```