# Stat 432 Homework 12

Sharvi Tomar

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.

```
library(jpeg)
img<-readJPEG("hund2.jpg")

par(mar = rep(0.2, 4))
plot(
  c(0, 400),
  c(0, 500),
  xaxt = 'n',
  yaxt = 'n',
  bty = 'n',
  pch = '',
  ylab = '',
  xlab = ''
)

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

- [10 Points] Report the following information of your data:
  - Dimension of the original image

```
# Dimension of the original image
print("The dimension of original image is:")
```

```
## [1] "The dimension of original image is:"
```

```
dim(img)
```

```
## [1] 100 100   3
```

- Dimension of the data once you transform the image to a version that you could apply k-means

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

# Dimension of data to apply k-means
print("The dimension of data to apply k-means is:")
```

```
## [1] "The dimension of data to apply k-means is:"
```

```
dim(img_expand)
```

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

- Total variations of your data

```
# Total variations of data
variation=sum(dist(img_expand)^2)/nrow(img_expand)
print("The Total variations of data is:")
```

```
## [1] "The Total variations of data is:"
```

```
variation
```

```
## [1] 2262.813
```

- [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

The 3 values of k chosen for K-means clustering are 2,5 and 15.

```
library(usefun)
# Function to perform K-means, report within-cluster variance, cluster means,
# plot image with pixel replaced by its corresponding cluster mean

pixals.clustered <- function(img, k)
{
  img_expand = apply(img, 3, c)
  kmeanfit <- kmeans(img_expand, k)

  within_cluster_variance=kmeanfit$withinss/kmeanfit$size

  print(paste(c("Within-cluster variance for k=", k)))
  print(paste(c(kmeanfit$withinss)))
  print_empty_line()
  print(paste(c("Cluster means for k=",k)))
  print(paste(c(kmeanfit$centers)))

  new_img_expand = kmeanfit$centers[kmeanfit$cluster,]

  new_img = img
  for (j in 1:3)
    new_img[, , j] = matrix(new_img_expand[, j], dim(img)[1], dim(img)[2])

  return(new_img)
}
```

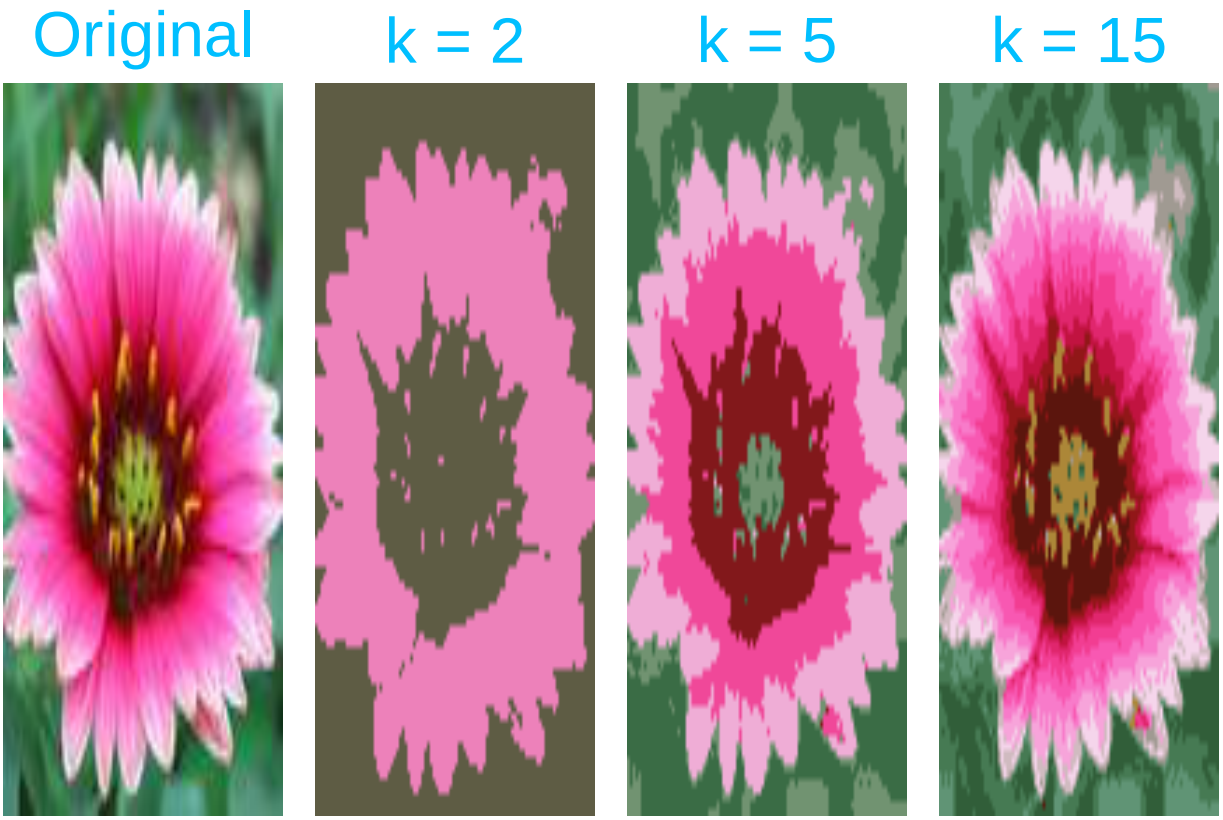
```
  par(mfrow=c(1,4))
  par(mar=rep(0.2, 4))
  plot(c(0, 400), c(0, 540), xaxt = 'n', yaxt = 'n', bty = 'n',
       pch = '', ylab = '', xlab = '')
  rasterImage(img, 0, 0, 400, 500)
```

```
  text(200, 530, "Original", col = "deepskyblue", cex = 3)

  for (k in c(2, 5, 15))
  {
      par(mar=rep(0.2, 4))
      plot(c(0, 400), c(0, 540), xaxt = 'n', yaxt = 'n', bty = 'n',
           pch = '', ylab = '', xlab = '')
      rasterImage(pixals.clustered(img, k), 0, 0, 400, 500)
      text(200, 530, paste("k =", k), col = "deepskyblue", cex = 3)
  }
```

```
## [1] "Within-cluster variance for k=" "2"
## [1] "410.939631679504" "475.807241557047"
##
## [1] "Cluster means for k=" "2"
## [1] "0.931184528605992" "0.370494011742901" "0.505475502046104"
## [4] "0.359547726954159" "0.729455214801482" "0.266004765687054"
```

```
## [1] "Within-cluster variance for k=" "5"
## [1] "75.957006979577"  "49.2737982903886" "80.7563252815164" "63.8719233355196"
## [5] "35.9860891863317"
##
## [1] "Cluster means for k=" "5"
##  [1] "0.940581520861499" "0.444939201003751" "0.938434708763643"
##  [4] "0.510117756218703" "0.22936811253352"  "0.283949106300518"
##  [7] "0.578296560285109" "0.678065905403151" "0.092881411437227"
## [10] "0.421774416263553" "0.598529624246752" "0.443655147559364"
## [13] "0.841044538465037" "0.107145878532346" "0.272225637188088"
```




```
## [1] "Within-cluster variance for k=" "15"
##  [1] "11.4565339038888" "3.95863124804343" "6.00717679654982" "6.51074919902597"
##  [5] "4.39834653575473" "8.53013314512158" "12.198674255721"  "5.60252138530074"
##  [9] "5.53495808399487" "7.93259359522355" "3.82624351496457" "2.85718796715831"
## [13] "10.0955974362656" "3.34380412834107" "4.32833518389166"
##
## [1] "Cluster means for k=" "15"
##  [1] "0.190604271448357"  "0.947835852607298"  "0.959990409207161"
##  [4] "0.567552650689905"  "0.969528841919859"  "0.376651447980768"
##  [7] "0.67463768115942"   "0.629337013397733"  "0.817282583621683"
## [10] "0.357254901960785"  "0.973159144893117"  "0.755705376706556"
## [13] "0.272663989290497"  "0.879831932773108"  "0.972155489967285"
## [16] "0.366677600408194"  "0.239774175924241"  "0.837153665814151"
## [19] "0.0898184458968773" "0.642489703939699"  "0.579236418977039"
## [22] "0.530434782608695"  "0.603484560979817"  "0.732281430219146"
## [25] "0.0842329873125725" "0.343467933491687"  "0.0757570650036517"
## [28] "0.479518072289157"  "0.149728126544736"  "0.480746791131856"
## [31] "0.224342153218165"  "0.572579372306198"  "0.922271952259165"
## [34] "0.0932171387073348" "0.856093454554873"  "0.460720351132579"
## [37] "0.217732310315431"  "0.572031510551434"  "0.752544405997693"
## [40] "0.0527277970011534" "0.698900842997531"  "0.251250070228665"
## [43] "0.327124970470117"  "0.427096721041358"  "0.79063765529549"
```

### 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.

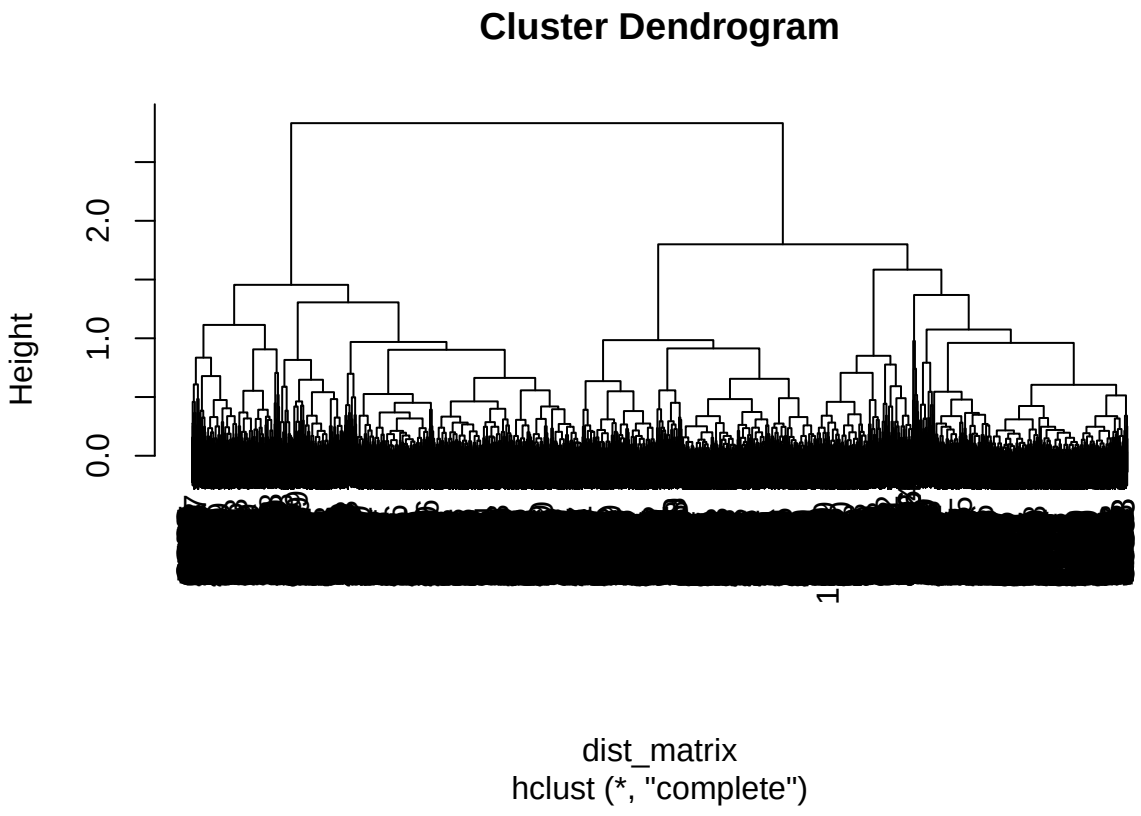
```
dist_matrix=dist(img_expand, method = "manhattan")
```

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

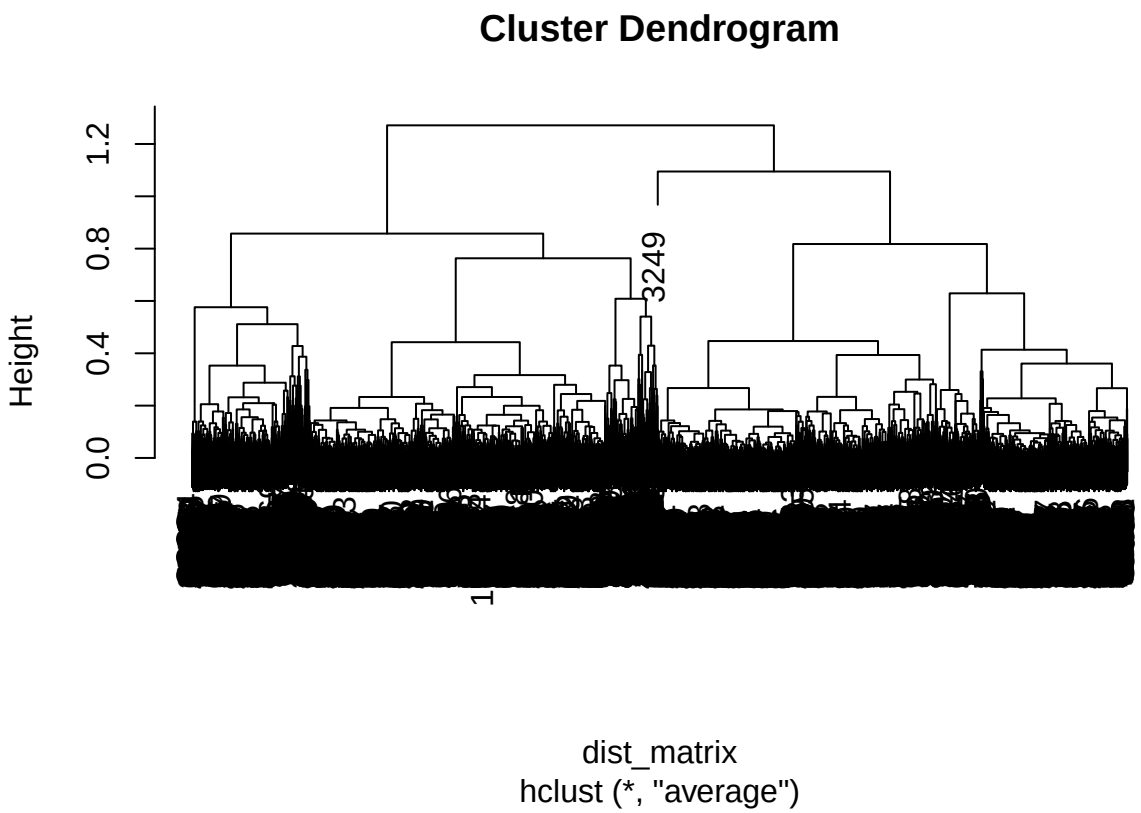
- [15 Points] Try both complete, single and average linkage. Provide a plot of the dendrogram for both methods.

```
hc_comp = hclust(dist_matrix, method = "complete")
hc_sing = hclust(dist_matrix, method = "single")
hc_avg = hclust(dist_matrix, method = "average")
```

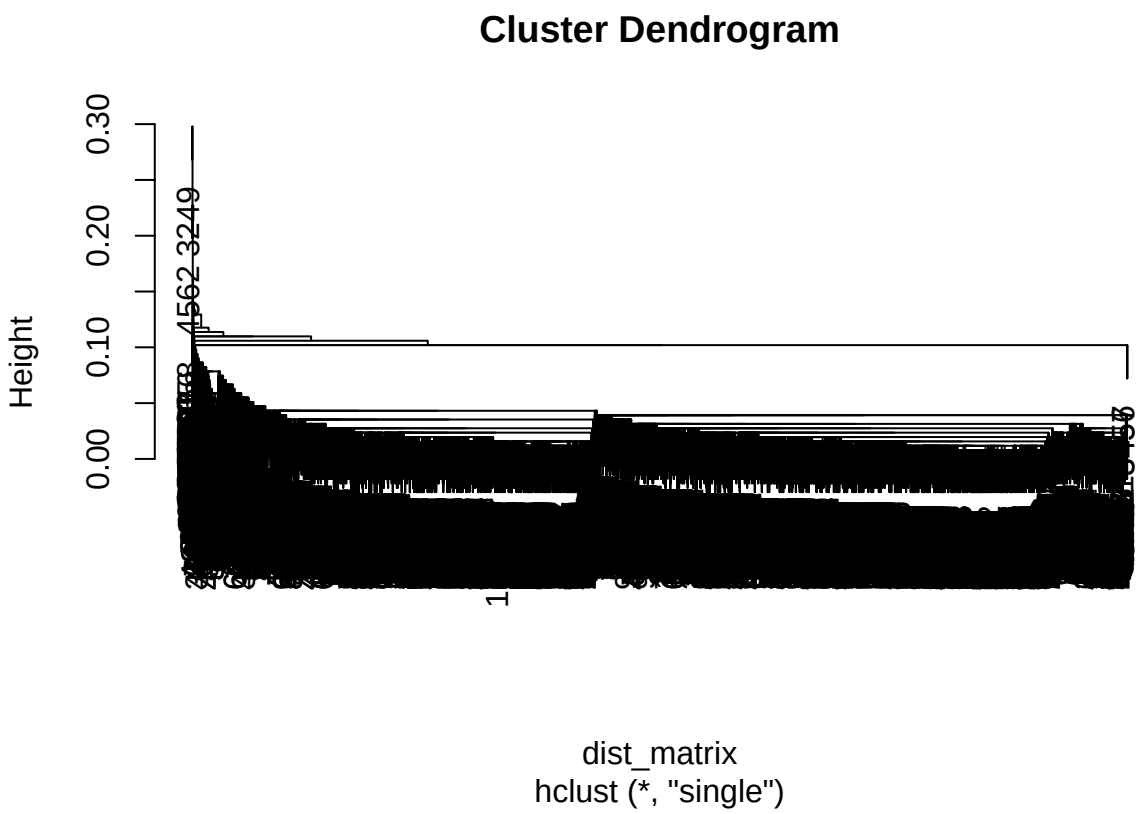
```
# Dendrogram for hierarchical clustering using complete method
plot(hc_comp)
```




```
# Dendrogram for hierarchical clustering using average method
plot(hc_avg)
```




```
# Dendrogram for hierarchical clustering using single method
plot(hc_sing)
```



- [10 Points] Based on what you have, pick one final clustering result. You need to explain the rational for your choice.

Since this is an image compression exercise we are so the choice of clustering result is based on which clustering result produces a better image in terms of image quality.

1) Selecting the number of clusters (tricky problem)

There are many different approaches depending upon the problem at hand. One could select the optimal number of clusters by:

Picking a cutoff where the height of the next split is short. The height of each split represents how separated the two subsets are (the distance when they are merged).

Equivalently, how much can the depth cut-off line slide without vertically without touching the dendrogram's horizontal lines.

2) Selecting the best linkage method

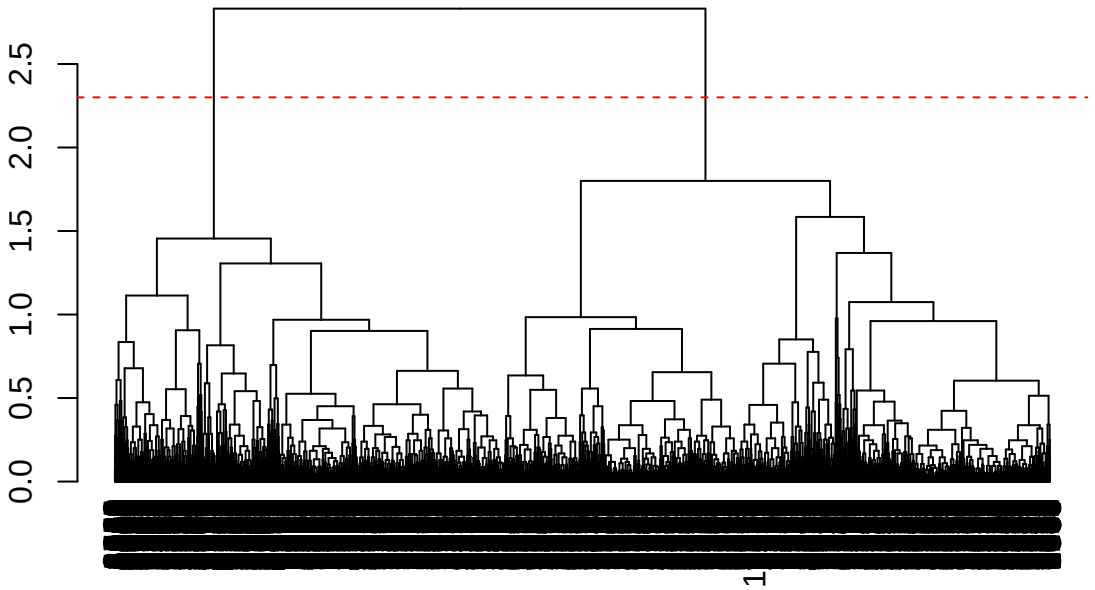
Eliminating 'single' linkage result as it the dengrogram does not provide any clear information.

Choosing the dendogram that shows "nice" clustering then we may prefer to use that linkage. "nice" clustering-gives number of clusters that are consistent with the truth.

```
sort(hc_comp$height,decreasing = TRUE)[1:5]
```

```
## [1] 2.831373 1.800000 1.584314 1.454902 1.368627
```

```
dend <- as.dendrogram(hc_comp)
depth.cutoff <- 2.3
plot(dend)
abline(h=depth.cutoff,col="red",lty=2)
```



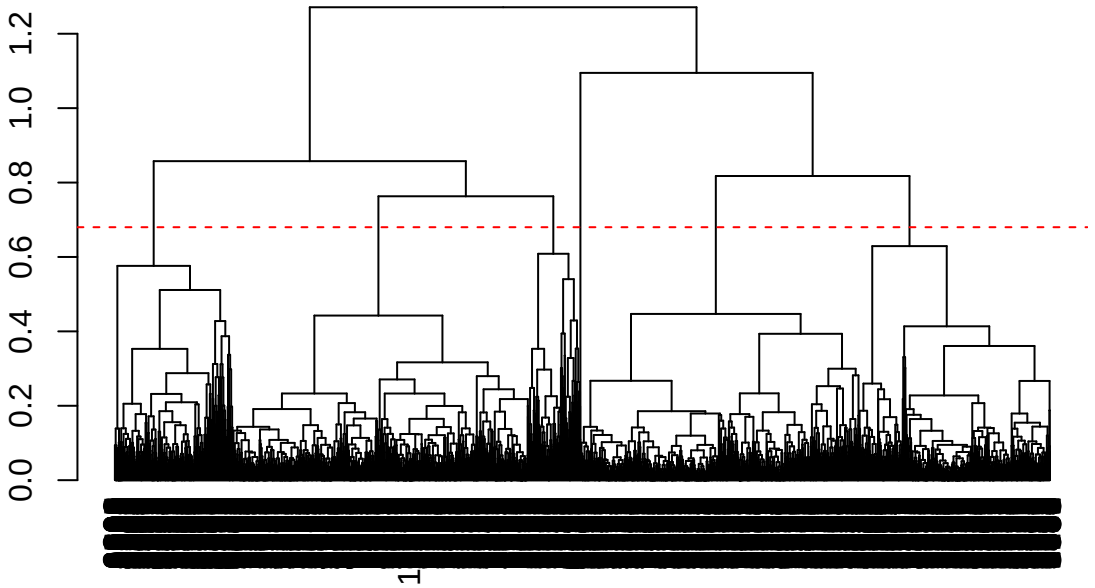

From the dendrogram of 'complete' linkage method, we can derive that k=2 or 2 clusters is a good choice. On increasing the value of k the separability between clusters reduces drastically.No other depth cut-off line

proves suitable. However, for the flower image, k=2 doesn't seem a reasonable choice as the picture contains several colors such as pink, green, yellow, white and other shades. So reducing it a 2-shade picture would not be a great choice.

```
sort(hc_avg$height,decreasing = TRUE)[1:5]
```

```
## [1] 1.2711905 1.0948104 0.8575026 0.8176291 0.7630934
```

```
dend <- as.dendrogram(hc_avg)
depth.cutoff <- 0.68
plot(dend)
abline(h=depth.cutoff,col="red",lty=2)
```




From the dendrogram of 'complete' linkage method, we can derive k=3 and k=6 seems a good choice for the number of clusters as there is reasonable separability between clusters. It also seems like a suitable like for our image of flower. Checking the final image constructed using these k values to decide the final cluster setting.

```
subfinal_hc <- cutree(hc_avg, k = 3)
final_hc <- cutree(hc_avg, k = 6)
```

```
par(mrow=c(1,2))
```

```
## Warning in par(mrow = c(1, 2)): "mrow" is not a graphical parameter
```

```
########## subfinal_hc: k=3, 'average' linkage ###########
# Calculating the cluster centers
cluster_center1 = aggregate(img_expand,list(cluster=subfinal_hc),mean)
cluster_center1 = cluster_center1[,2:4]
# Replace pixels with their corresponding cluster mean
new_img_expand1 = cluster_center1[subfinal_hc,]
# Converting back to the array that can be plotted as an image
new_img1 = img
```

```
new_img1[, , 1] = matrix(new_img_expand1[, 1], 100, 100)
new_img1[, , 2] = matrix(new_img_expand1[, 2], 100, 100)
new_img1[, , 3] = matrix(new_img_expand1[, 3], 100, 100)


########## subfinal_hc: k=6, 'average' linkage ###########
# Calculating the cluster centers
cluster_center = aggregate(img_expand,list(cluster=final_hc),mean)
cluster_center = cluster_center[,2:4]
# Replace pixels with their corresponding cluster mean
new_img_expand = cluster_center[final_hc,]
# Converting back to the array that can be plotted as an image
new_img = img
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)

par(mfrow=c(1,2))
# plot the new image
plot(
  c(0, 100),
  c(0, 100),
  xaxt = 'n',
  yaxt = 'n',
  bty = 'n',
  pch = '',
  ylab = '',
  xlab = ''
)
rasterImage(new_img1, 0, 0, 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)
```

k=6 gives a much better picture hence, it seems a more suitable choice.

Hence, we the final clustering choice: - Hierarchical Clustering method - "average" - Number of clusters - 6

- [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.

```
# Calculating the cluster centers
cluster_center = aggregate(img_expand,list(cluster=final_hc),mean)
cluster_center = cluster_center[,2:4]
cluster_center
```

```
##          V1        V2         V3
## 1 0.2785843 0.4758721 0.33704360
## 2 0.9486363 0.6369565 0.83173612
## 3 0.6842855 0.5899915 0.45765975
## 4 0.4323073 0.1177337 0.08002848
## 5 0.9080254 0.2173631 0.52118875
## 6 0.9843137 0.7725490 0.00000000
```

```
# Replace pixels with their corresponding cluster mean
new_img_expand = cluster_center[final_hc,]

# Converting back to the array that can be plotted as an image
new_img = img
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)
```

- [10 Points] Plot this new image.

```
# 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)
```