DIMENSIONLESS
TECHNOLOGY

Segmenting Images to Create Data

INTRODUCTION

- We will see how to apply clustering techniques to segment images, with the main application being geared towards medical image segmentation.

- At the end of this lesson, you will get a head start on how to cluster an MRI brain image by tissue substances and locate pathological anatomies.

Image Segmentation

- Divide up digital images to salient regions/clusters corresponding to individual surfaces, objects, or natural parts of objects
- Clusters should be uniform and homogenous with respect to certain characteristics (color, intensity, texture)
- Goal: Useful and analyzable image representation

Wide Applications

- Medical Imaging
  - Locate tissue classes, organs, pathologies and tumors
  - Measure tissue/tumor volume
- Object Detection
  - Detect facial features in photos
  - Detect pedestrians in footages of surveillance videos
- Recognition tasks
  - Fingerprint/Iris recognition

Various Methods

- Clustering methods
  - Partition image to clusters based on differences in pixel colors, intensity or texture
- Edge detection
  - Based on the detection of discontinuity, such as an abrupt change in the gray level in gray-scale images
- Region-growing methods
  - Divides image into regions, then sequentially merges sufficiently similar regions

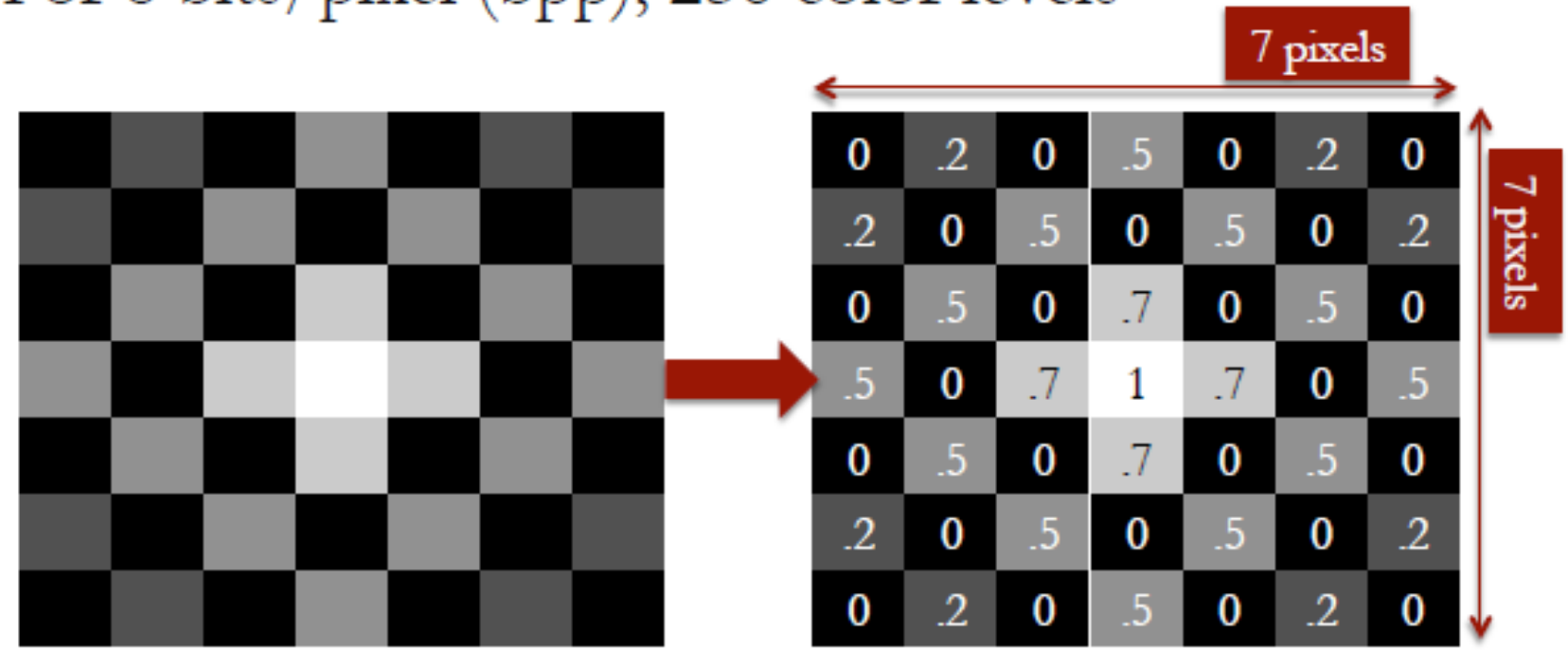
In this Lecture

- Review hierarchical and *k*-means clustering in R
- Restrict ourselves to gray-scale images
  - Simple example of a flower image (flower.csv)
  - Medical imaging application with examples of transverse MRI images of the brain (healthy.csv and tumor.csv)
- Compare the use, pros and cons of all analytics methods we have seen so far

Clustering Pixels

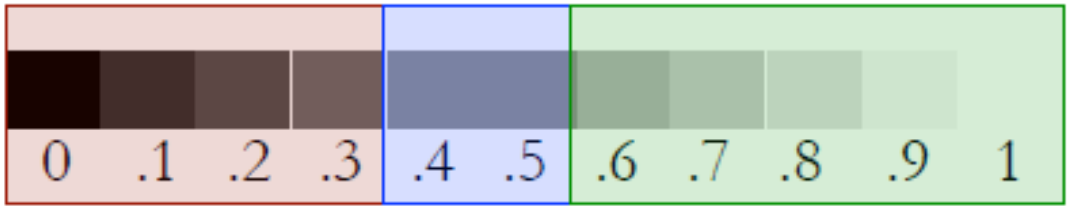
Grayscale Images

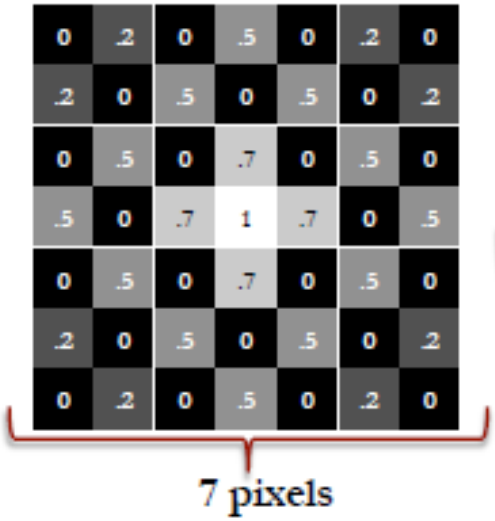
- Image is represented as a matrix of pixel intensity values ranging from 0 (black) to 1 (white)
- For 8 bits/pixel (bpp), 256 color levels

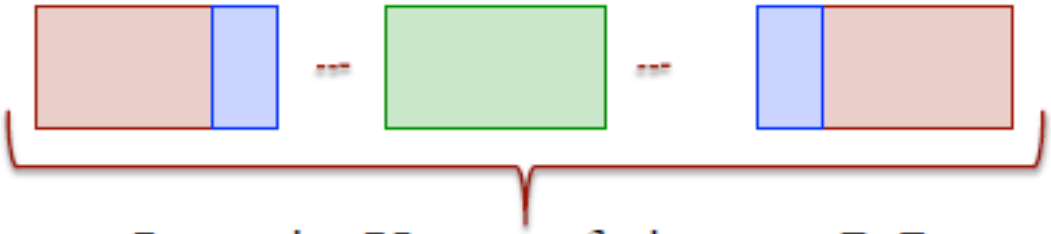


Grayscale Image Segmentation

- Cluster pixels according to their intensity values







Intensity Vector of size n = 7x7

→ Pairwise distances n(n-1)/2

Quick Question

- What should the input be to the clustering algorithm?
- The input to the clustering algorithm should be a vector containing 49 elements, or intensity values.

Implementing in R

Data

- Load the data from the file https://storage.googleapis.com/dimensionless/Analytics/flower.csv .
- We will call our data frame “flower”.
- PS:- There is no header in the data.
- Convert the dataframe into matrix “flowerMatrix”.
- Look at the structure of the matrix
- Further convert the matrix to vector “flowerVector”

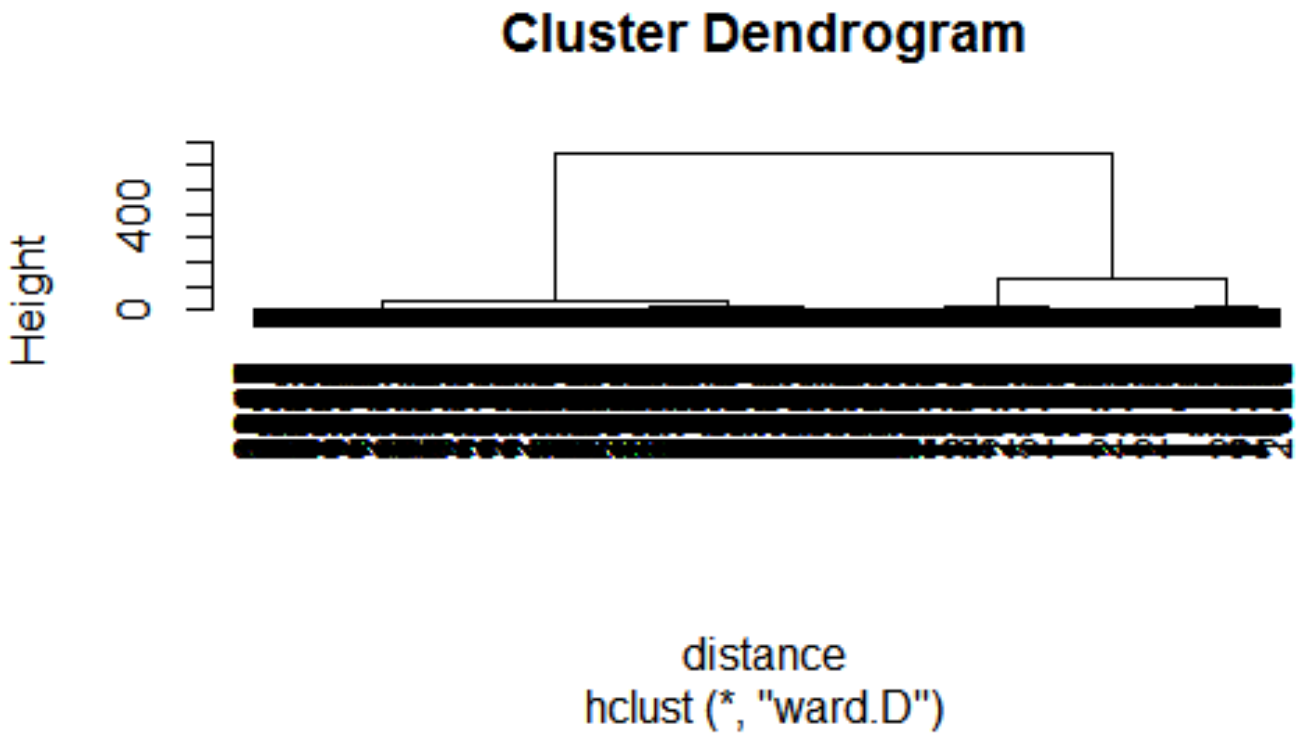
Distance

- The first step is to create the distance matrix
  - distance=dist(flowerVector,method="euclidean")

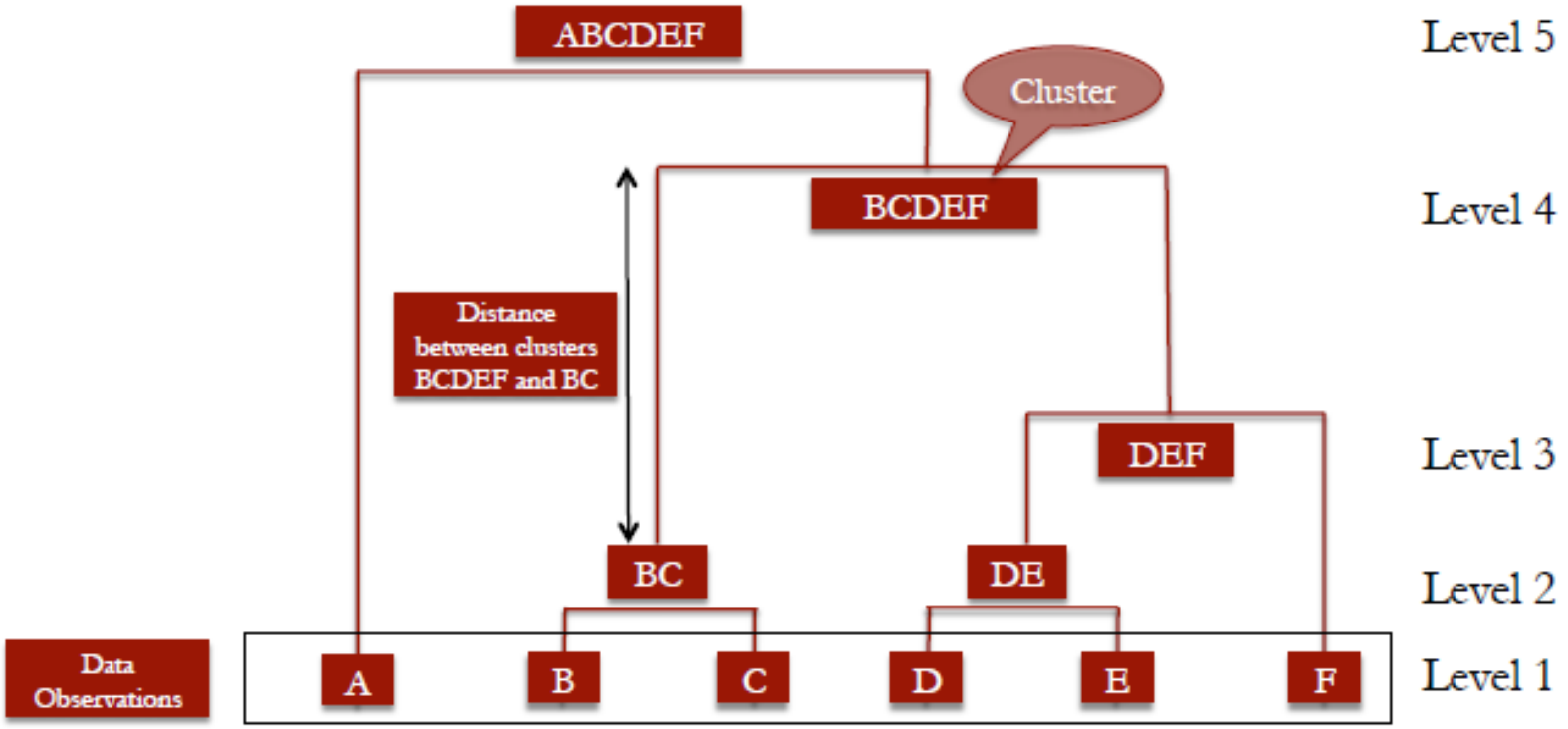
The hierarchical clusters

- Now we can cluster the intensity values using hierarchical clustering.
- We'll call it clusterIntensity
  - `clusterIntensity<-hclust(distance, method="ward.D")`
- Plot the cluster dendrogram
  - `plot(clusterIntensity)`

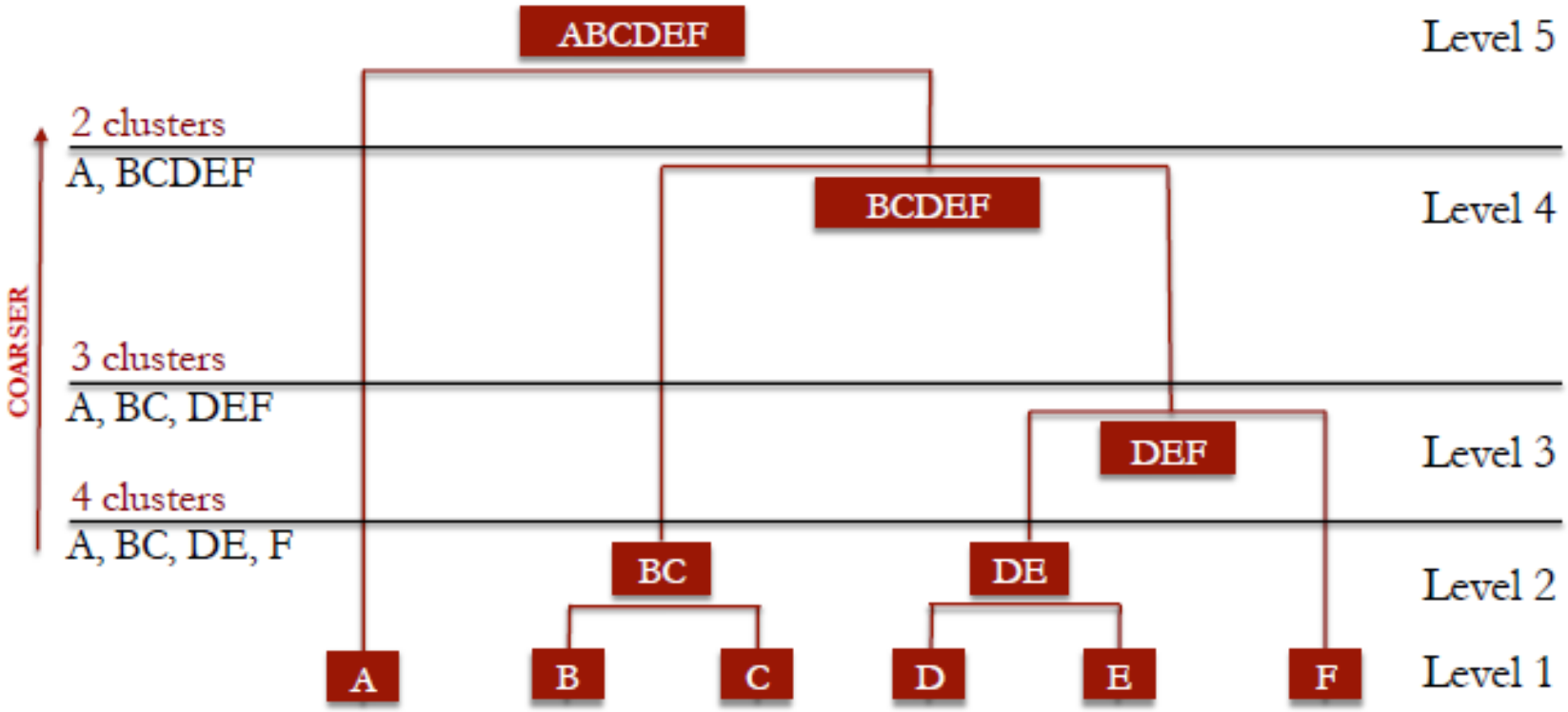
Dendrogram



Dendrogram Example



Dendrogram Example



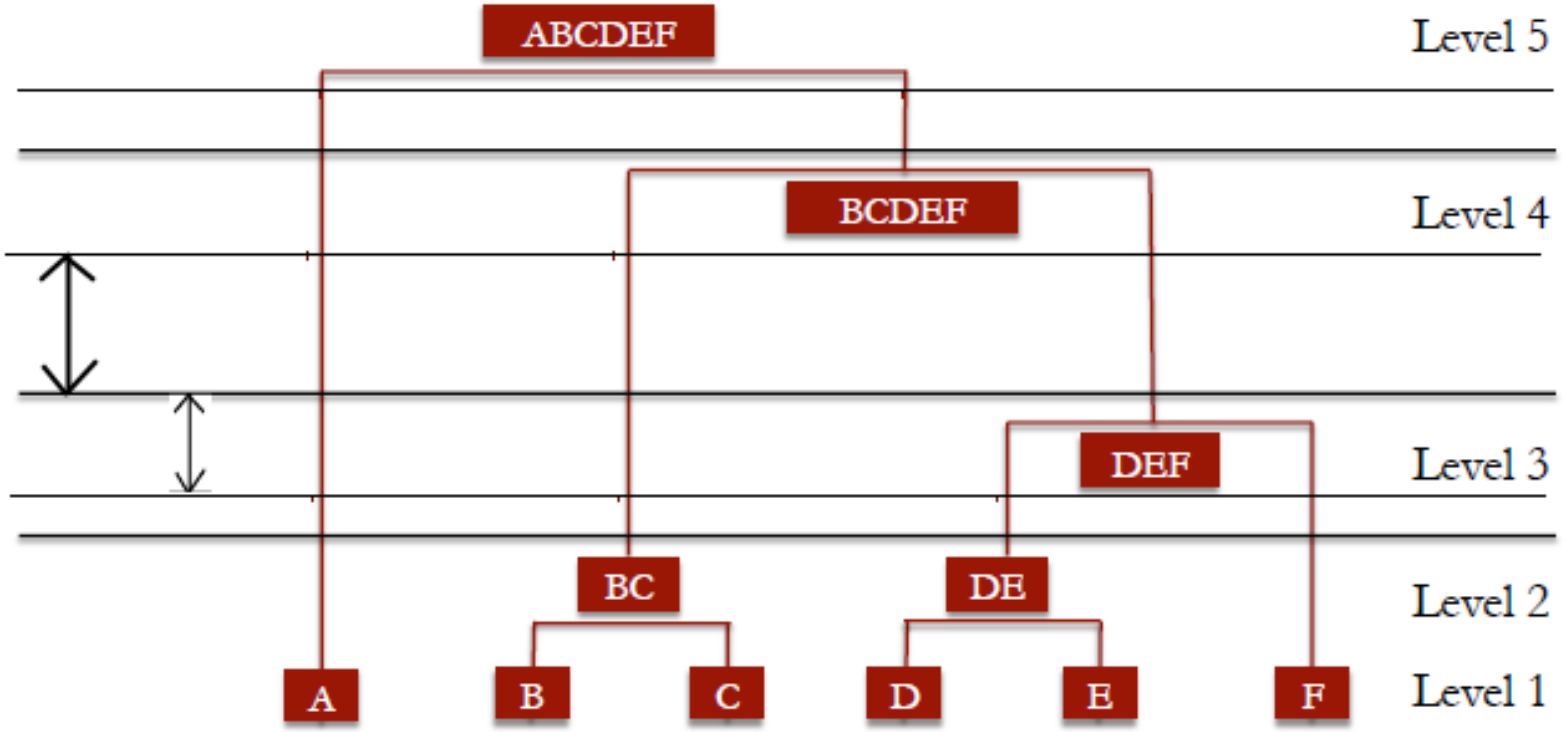
Number of Clusters

- What to choose, two, three, or four clusters?
- The smaller the number of clusters, the coarser the clustering is
- Having many clusters may be too much of a stretch
- We should always have this trade-off in mind.

Number of Clusters

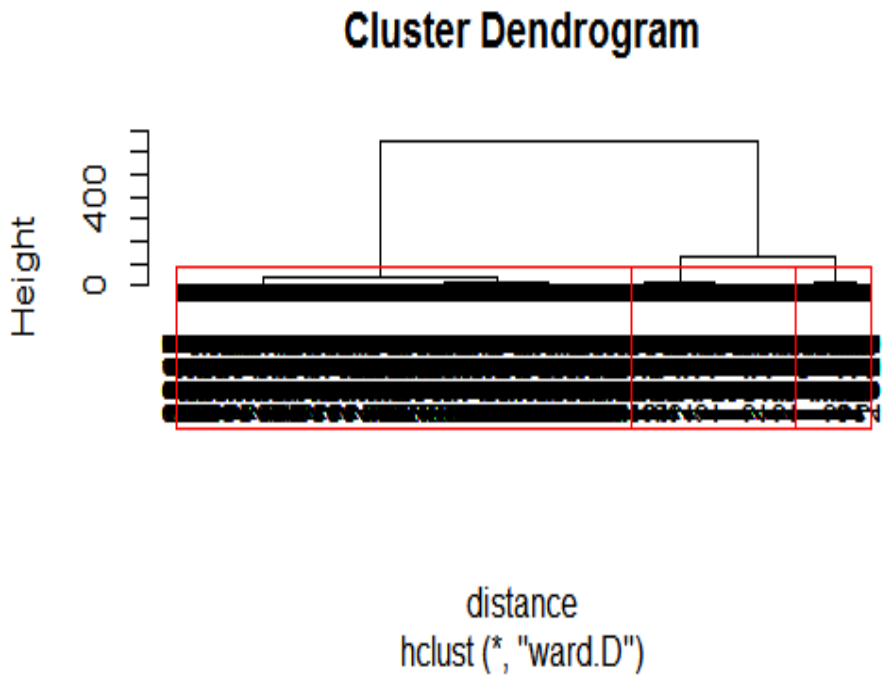
- Distance information between clusters can guide our choice of the number of clusters.
- A good partition belongs to a cut that has a good enough room to move up and down

Moving Space



Visualizing the cluster

- We can actually visualize the cuts by plotting rectangles around the clusters on our tree
  - rect.hclust(clusterIntensity,3)



Splitting the Data

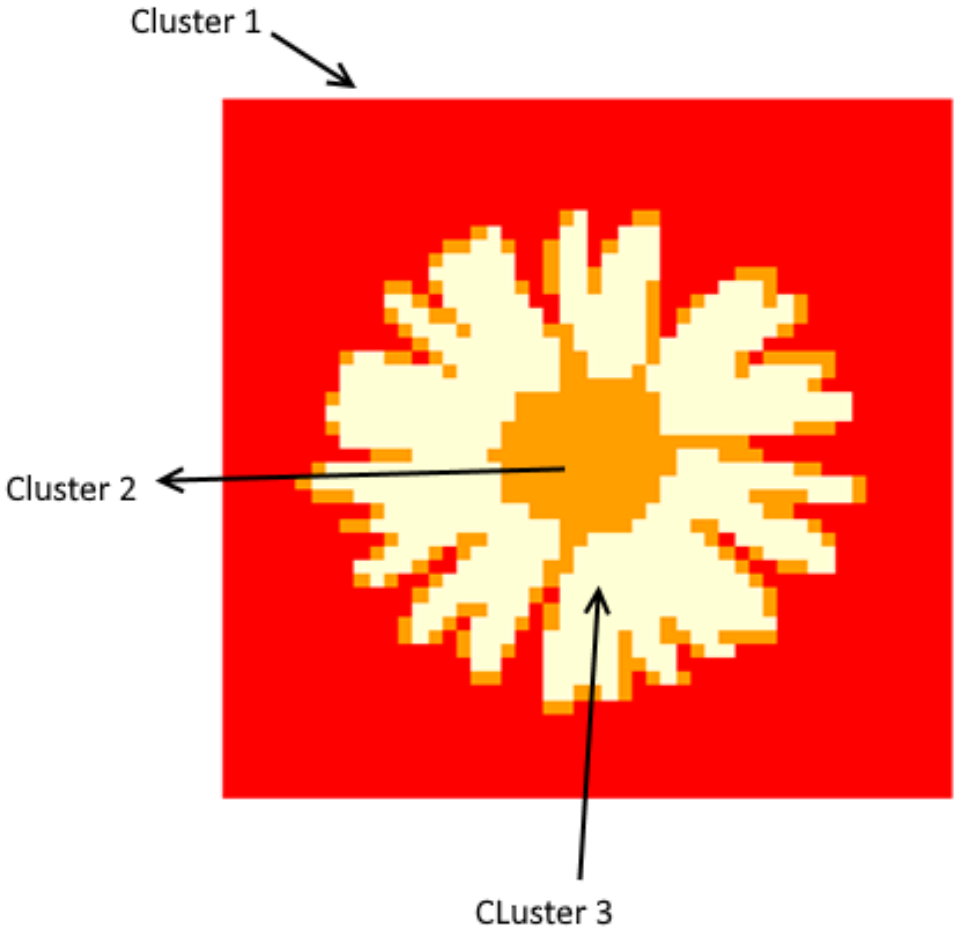
- Split the data into these 3 clusters.
- We'll call it flowerClusters.
  - flowerClusters<-
    cutree(clusterIntensity,k=3)
- flowerClusters is actually a vector that assigns each intensity value in the flower vector to a cluster.
- It has the same length, which is 2,500, and has values 1, 2, and 3, which correspond to each cluster.

Quick Question

- Find out the mean intensity in each cluster.

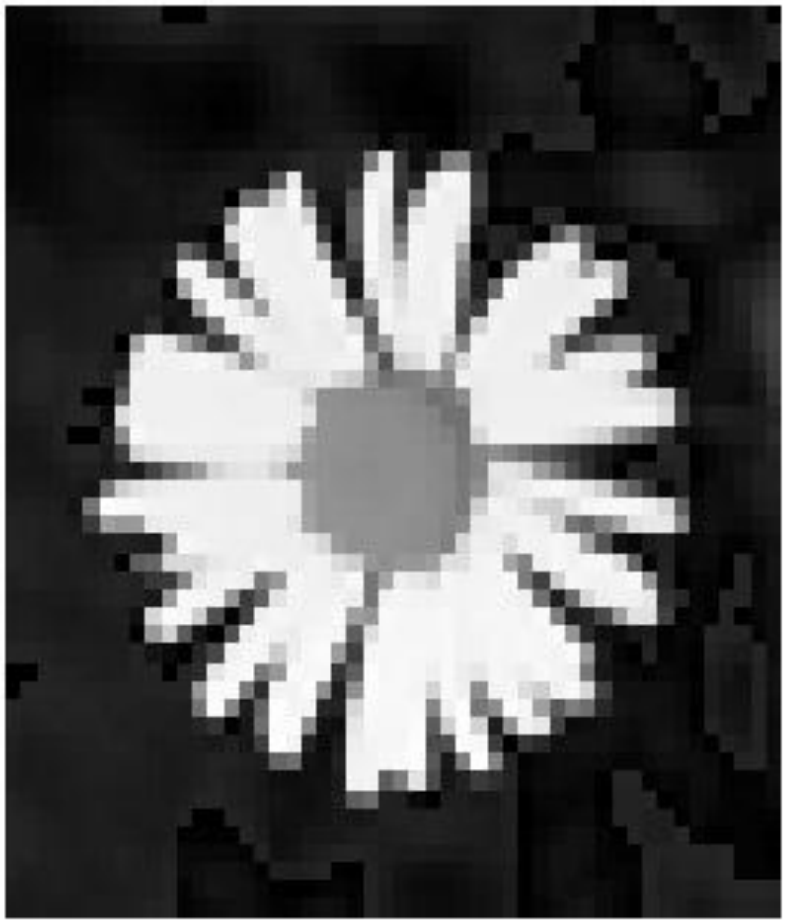
Image segments

- To output an image, we can use the image function which takes a matrix as an input.
- We need to convert flowerClusters into a matrix.
- We can do this by setting the dimension of this variable using dim function
  - dim(flowerClusters)= c(50,50)
- Now we can use the image function
  - image(flowerClusters,axes=FALSE)


Cluster 1
Cluster 2
CLuster 3

Original Image

- Let us now check how the original image looked.
- We'll use the image function on flowerMatrix
- We will add an additional argument regarding the color scheme
- We'll use the grey scale function
- The input to this function is a sequence of values that goes from 0 to 1, which actually is from black to white.
  - image(flowerMatrix,axes=FALSE,col = grey(seq(0,1,length=256)))



MRI Image

MRI

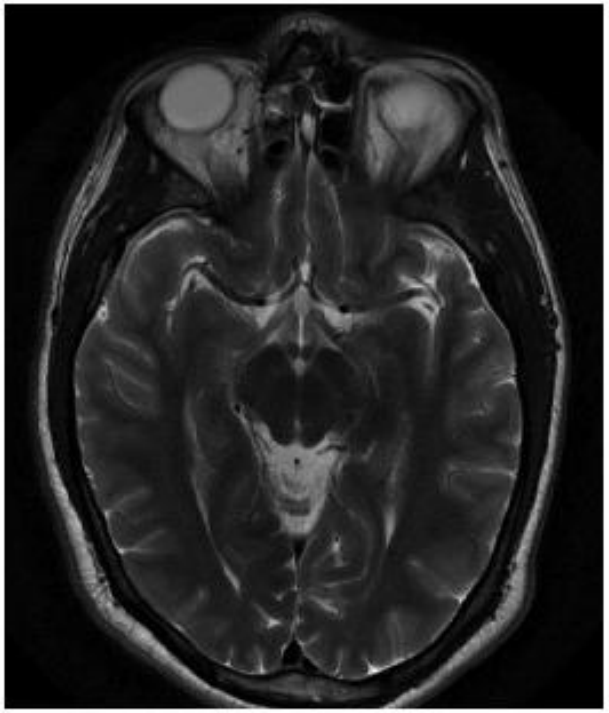
- We will try to segment an MRI brain image of a healthy patient using hierarchical clustering.
- We will be following the exact same steps we did in the previous lecture.
- Use the data from https://storage.googleapis.com/dimensionless/Analytics/healthy.csv
- We'll call the dataset "healthy

Data

- Read the data.
- Convert it into a matrix
- Look at the structure, we have 566 by 646 pixel resolution for our image.
- This MRI image is considerably larger than the little flower image that we saw
- Look at the image using the grey color scheme.

Image

- What we have is the T2-weighted MRI imaging of a top section of the brain.
- It shows different substances, such as the gray matter, the white matter, and the cerebrospinal fluid



Let’s see if we can isolate these substances via hierarchical clustering

Clustering

- Convert the healthy matrix to a vector, and let's call it healthyVector.
- Compute the distance
- You have got an error !!
- See the size of the vector.
- Find the number of distances that we have to compute.

K-means clustering

- The *k*-means clustering aims at partitioning the data into *k* clusters in which each data point belongs to the cluster whose mean is the nearest

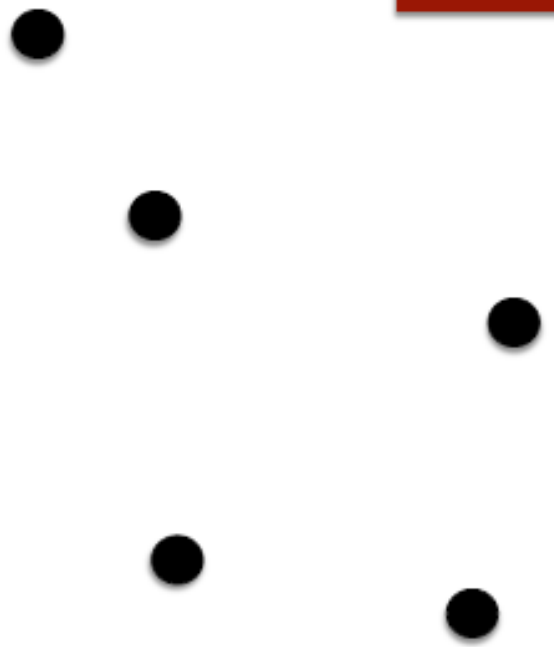
***k*-Means Clustering Algorithm**
1. Specify desired number of clusters *k*
2. Randomly assign each data point to a cluster
3. Compute cluster centroids
4. Re-assign each point to the closest cluster centroid
5. Re-compute cluster centroids
6. Repeat 4 and 5 until no improvement is made

K-means clustering

k-Means Clustering Algorithm

1. Specify desired number of clusters k

$k = 2$

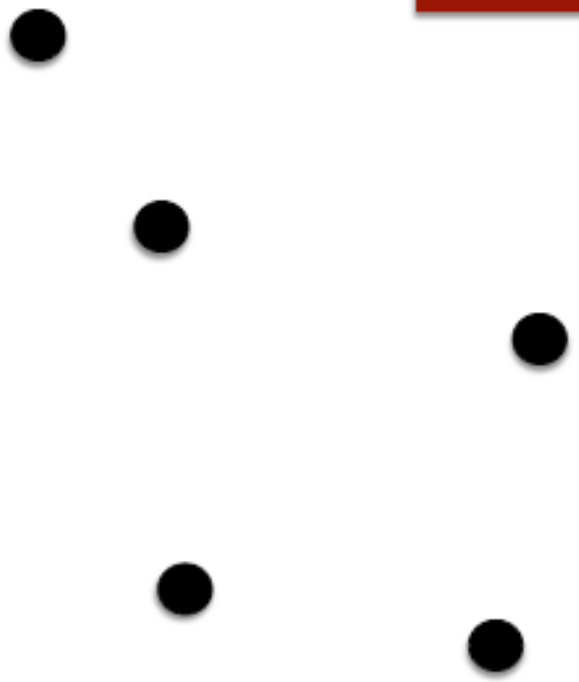


K-means clustering

k-Means Clustering Algorithm

1. Specify desired number of clusters k
2. Randomly assign each data point to a cluster

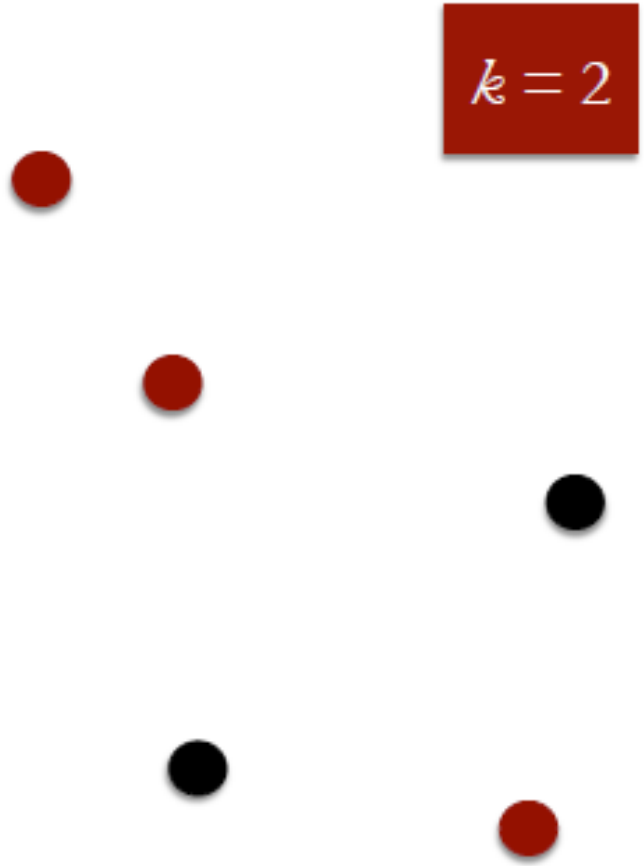
$k = 2$



K-means clustering

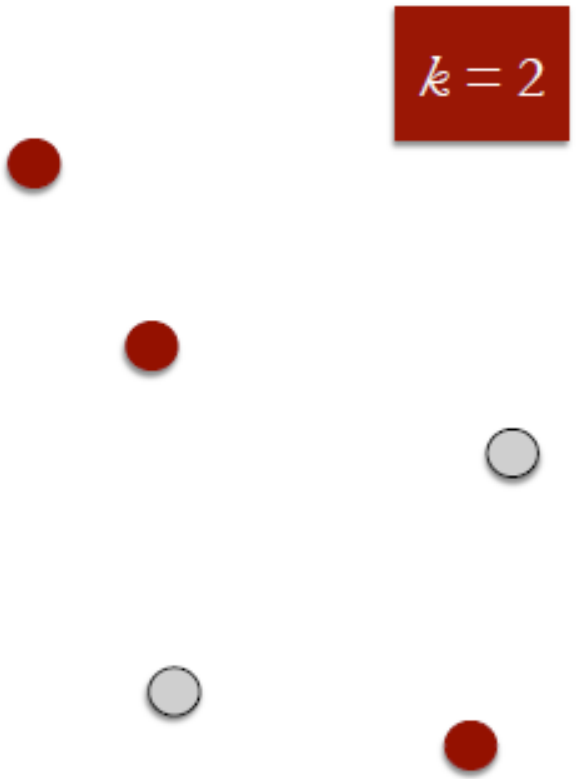
k-Means Clustering Algorithm

1. Specify desired number of clusters k
2. Randomly assign each data point to a cluster



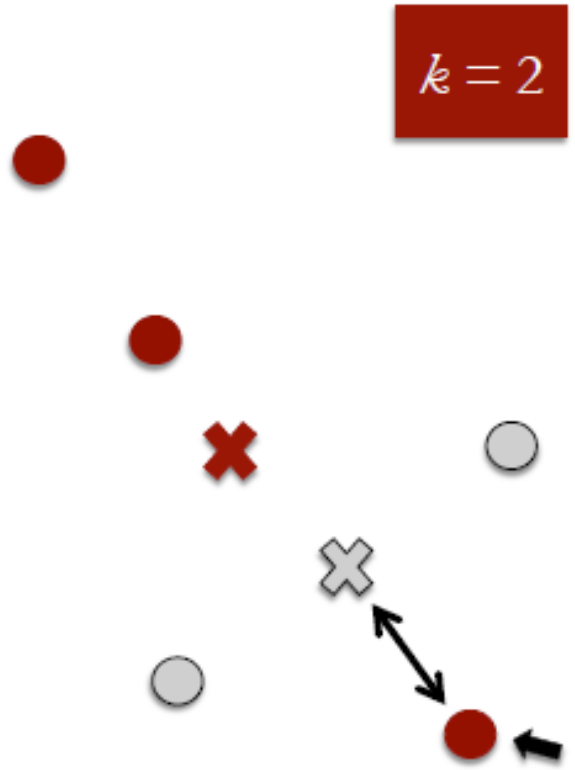
K-means clustering

k-Means Clustering Algorithm
1. Specify desired number of clusters *k*
2. Randomly assign each data point to a cluster



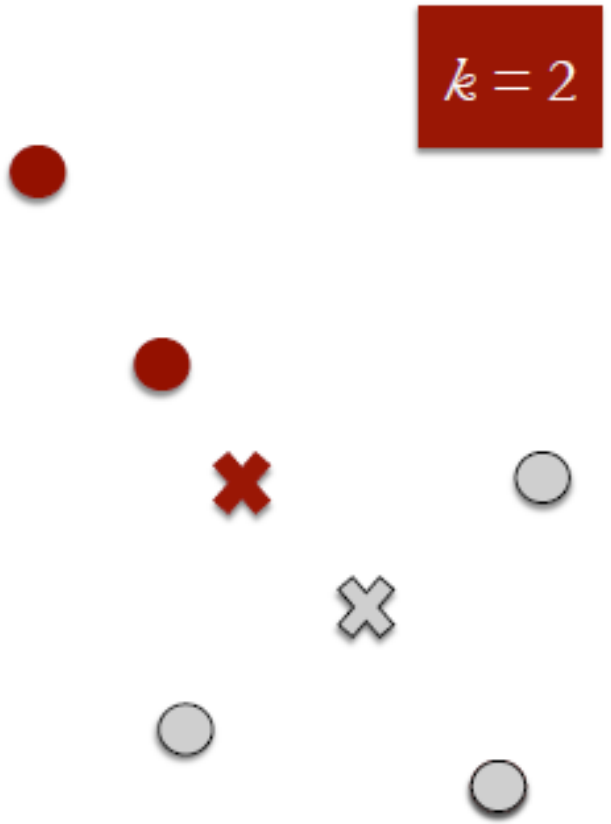
K-means clustering

k-Means Clustering Algorithm
1. Specify desired number of clusters *k*
2. Randomly assign each data point to a cluster
3. Compute cluster centroids
4. Re-assign each point to the closest cluster centroid



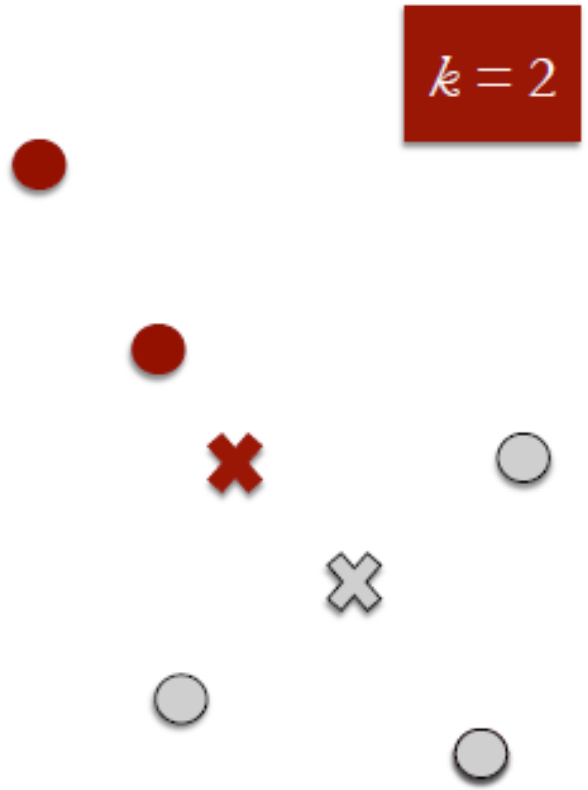
K-means clustering

k-Means Clustering Algorithm
1. Specify desired number of clusters *k*
2. Randomly assign each data point to a cluster
3. Compute cluster centroids
4. Re-assign each point to the closest cluster centroid



K-means clustering

k-Means Clustering Algorithm
1. Specify desired number of clusters *k*
2. Randomly assign each data point to a cluster
3. Compute cluster centroids
4. Re-assign each point to the closest cluster centroid
5. Re-compute cluster centroids
6. Repeat 4 and 5 until no improvement is made



K-Means Clustering in R

No. Of Clusters

- The first step in k-means clustering involves specifying the number of clusters, k.
- But how do we select k?
- Our clusters would ideally assign each point in the image to a tissue class or a particular substance, for instance, grey matter or white matter, and so on.
- These substances are known to the medical community.
- So setting the number of clusters depends on exactly what you're trying to extract from the image.

Clustering

- We'll start with k = 5.
- The k-means clustering algorithm starts by randomly assigning points to clusters, we should set the seed, so that we all obtain the same clusters.
  - Set.seed(1)
- We'll use the kmeans(), and save the output in KMC

Kmeans()

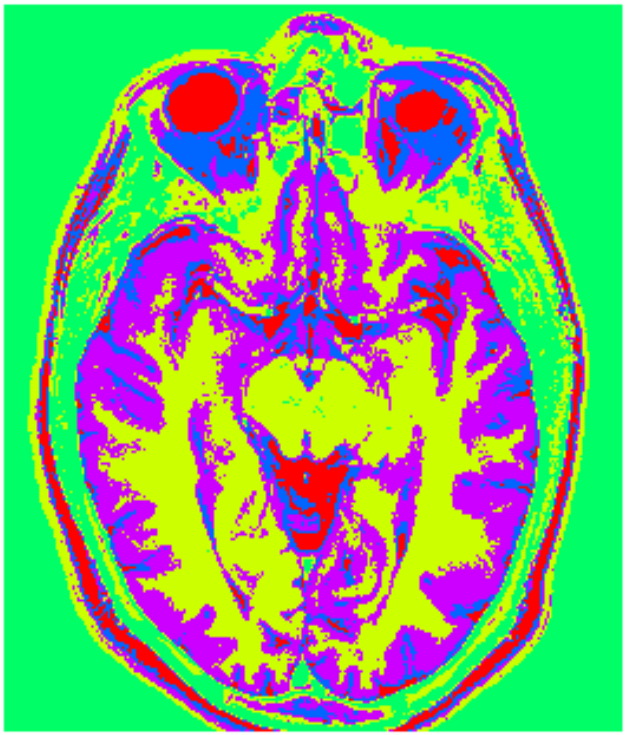
- The first input is whatever we are trying to cluster. In this case it is the healthy vector.
- The second argument is the number of clusters, and we can specify it using the argument centers, and that would be equal to k.
- And then, since the k-means is an iterative method that could take very long to converge, we need to set a maximum number of iterations.
  - `KMC = kmeans(healthyVector, centers = k, iter.max = 1000)`

Output

- Output the structure of the KMC variable.
- The first thing is the cluster vector which assigns each intensity value in the healthy vector to a cluster.
- To output the segmented image, we need to extract this vector.
  - healthyClusters<-KMC$cluster
- The mean intensity value within each of our 5 clusters can be extracted from KMC$centers.
- We can also see the size of each cluster KMC$size.

Segmented Image

- Convert the vector to a matrix
- Create the image using rainbow color palette.



Can we use the clusters, or the classes, found by our k-means algorithm on the healthy MRI image to identify tumors in another MRI image of a sick patient?

Scree Plots

- We can't use dendrograms for k-means clustering
- One common way to select the number of clusters is by using a ***scree plot***, which works for any clustering algorithm.
- A standard scree plot has the number of clusters on the x-axis, and the sum of the *within-cluster sum of squares* on the y-axis.
- The within-cluster sum of squares for a cluster is the sum, across all points in the cluster, of the squared distance between each point and the centroid of the cluster

Plotting Scree Plot

- To create the scree plot, the clustering algorithm is run with a range of values for the number of clusters.
- For each number of clusters, the within-cluster sum of squares can easily be extracted when using k-means clustering
  - KMC2 = kmeans(healthyVector, centers = 2, iter.max = 1000)
  - KMC2$withinss
- We would first repeat the kmeans function call above with centers = 3, centers = 4, etc. to create KMC3, KMC4, and so on.

Plotting Scree Plot

- We'll create a vector of "number of clusters" for our x-axis
  - NumClusters = seq(2,10,1)
- Our y-axis data will be

```
SumWithinss = c(sum(KMC2$withinss),
  sum(KMC3$withinss), sum(KMC4$withinss),
  sum(KMC5$withinss), sum(KMC6$withinss),
  sum(KMC7$withinss), sum(KMC8$withinss),
  sum(KMC9$withinss),
  sum(KMC10$withinss))
```

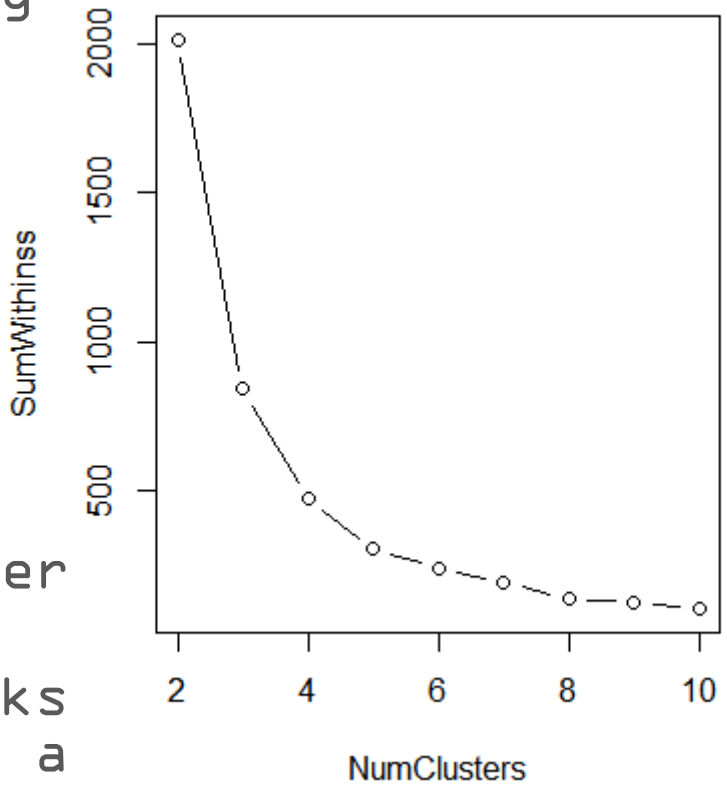
- We can then plot
  - plot(NumClusters, SumWithinss, type="b")

Plotting Scree Plot

- Alternately we can make the plot using sapply function also
  - SumWithinss = sapply(2:10, function(x) sum(kmeans(healthyVector, centers=x, iter.max=1000)$withinss))
  - NumClusters = seq(2,10,1)
  - plot(NumClusters, SumWithinss, type="b")

Interpreting the Plot

- To determine the best number of clusters using this plot, we want to look for a bend, or elbow, in the plot.
- We want to find the number of clusters for which increasing the number of clusters further does not significantly help to reduce the within-cluster sum of squares.
- For our dataset, it looks like 4 or 5 clusters is a good choice.



Detecting Tumors

Objective

- Previously, we identified clusters, or tissue substances, in a healthy brain image.
- It would be really helpful if we can use these clusters to automatically detect tumors in MRI images of sick patients.
- The tumor.csv file corresponds to an MRI brain image of a patient with oligodendroglioma, a tumor that commonly occurs in the front lobe of the brain.
- Since brain biopsy is the only definite diagnosis of this tumor, MRI guidance is key in determining its location and geometry.

Dataset

- Read the data from https://storage.googleapis.com/dimensionless/Analytics/tumor.csv
- We'll call the dataframe tumor.
- Convert it into a matrix tumorMatrix
- Convert it to a vector tumorVector
- We will not run the k-means algorithm again on the tumor vector.
- Instead, we will apply the k-means clustering results that we found using the healthy brain image on the tumor vector.

Implementing in R

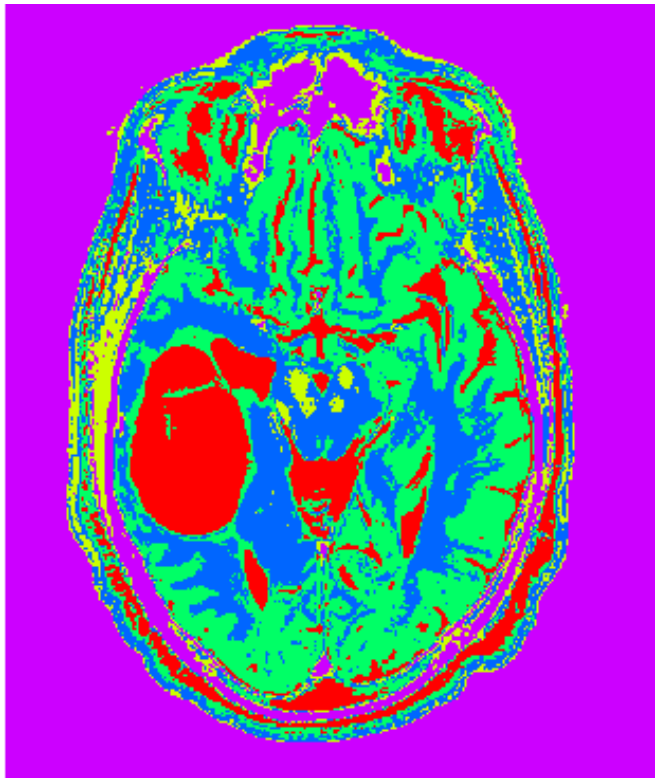
- We treat the healthy vector as training set and the tumor vector as a testing set.
- Install the package "flexclust"
- The flexclust package contains the object class KCCA, which stands for K-Centroids Cluster Analysis.
- We need to convert the information from the clustering algorithm to an object of the class KCCA. This conversion is needed before we can use the predict function on the test set tumorVector
    - KMC.kcca = as.kcca(KMC, healthyVector)
- This data conversion will take some time to run.

Making Predictions

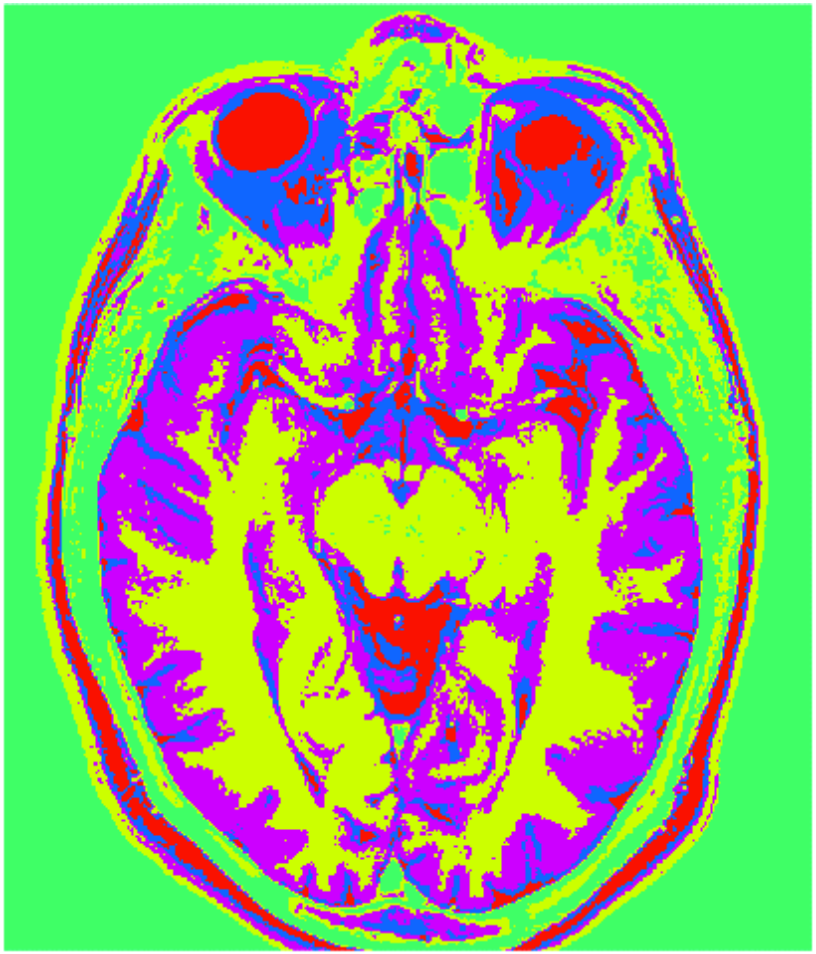
- Let us call the cluster vector tumorClusters
  - tumorClusters <-predict(KMC.kcca, newdata=tumorVector)
- And now, the tumorClusters is a vector that assigns a value 1 through 5 to each of the intensity values in the tumorVector, as predicted by the k-means algorithm.
- We will convert tumorClusters vector to matrix for visualizing the segmented image.
  - dim(tumorClusters)<-c(nrow(tumorMatrix),ncol(tumorMatrix))

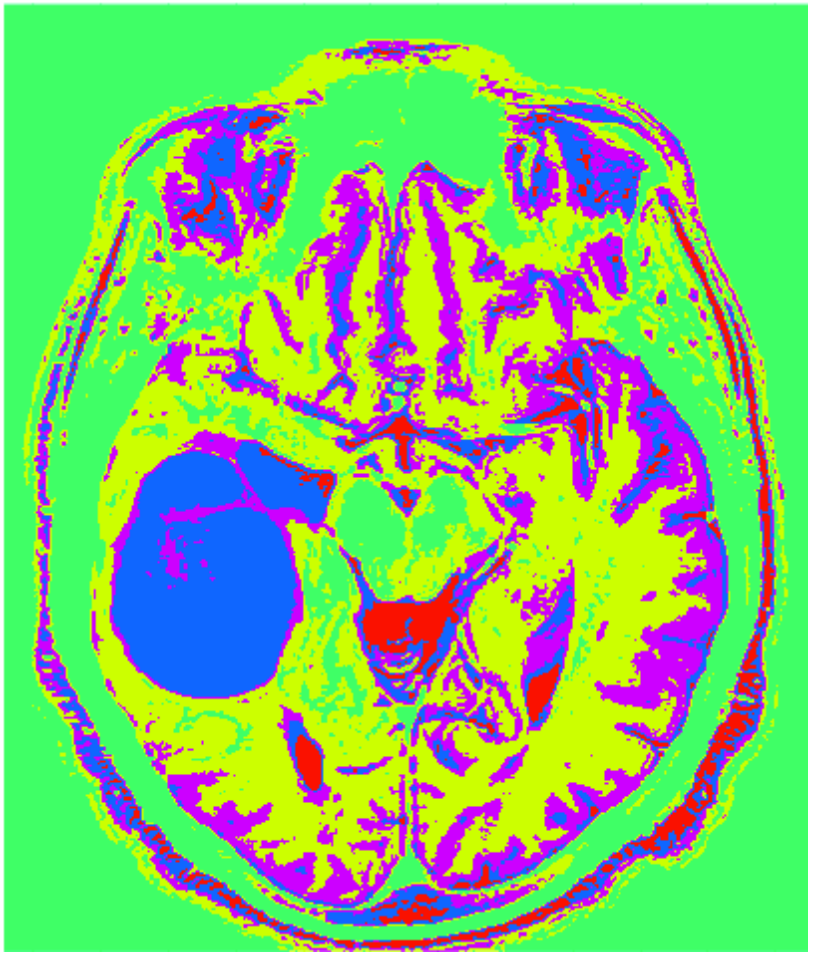
Segmented Image

- We can visualize the clusters by using the image function with the input tumorClusters matrix
- Can we detect the tumor.
- Yes, It is this abnormal substance highlighted in red that was no present in the healthy MRI image



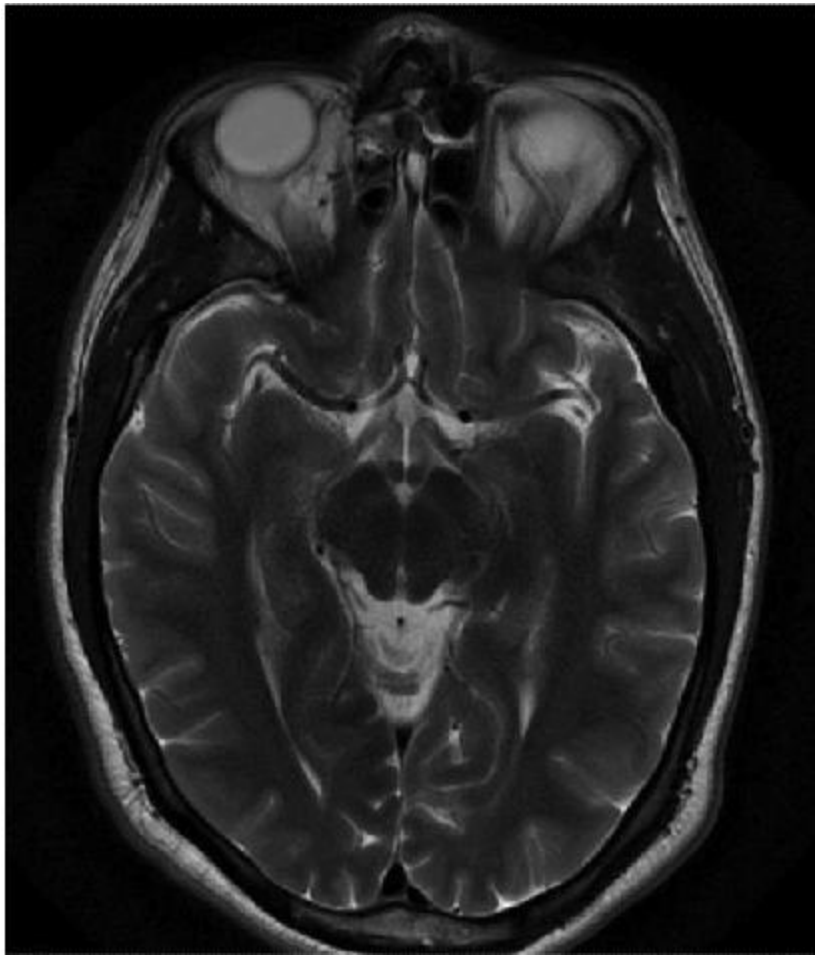
Segmented MRI Image

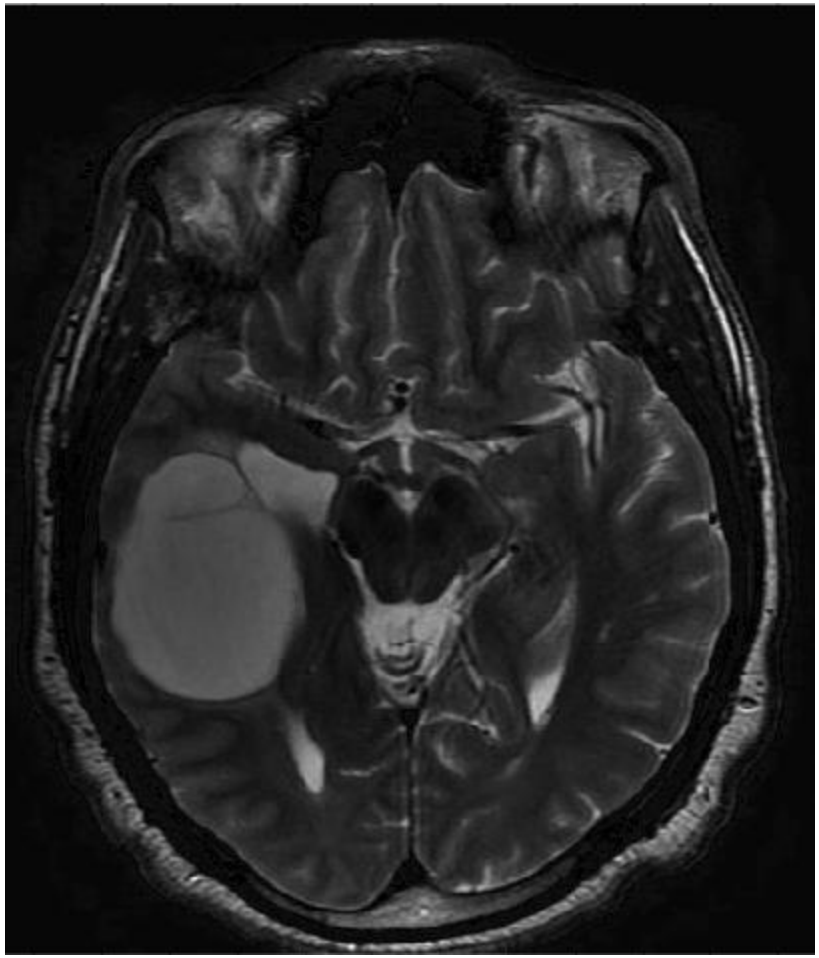




- For the sick patient, the substance highlighted in blue is the oligodendroglioma tumor.
- We do not see substantial blue regions in the healthy brain image, apart from the region around the eyes
- Looking at the eyes regions, we notice that the two images were not taken precisely at the same section of the brain
- This might explain some differences in shapes between the two images
- Let's see how the images look like originally.

T2 Weighted MRI Images





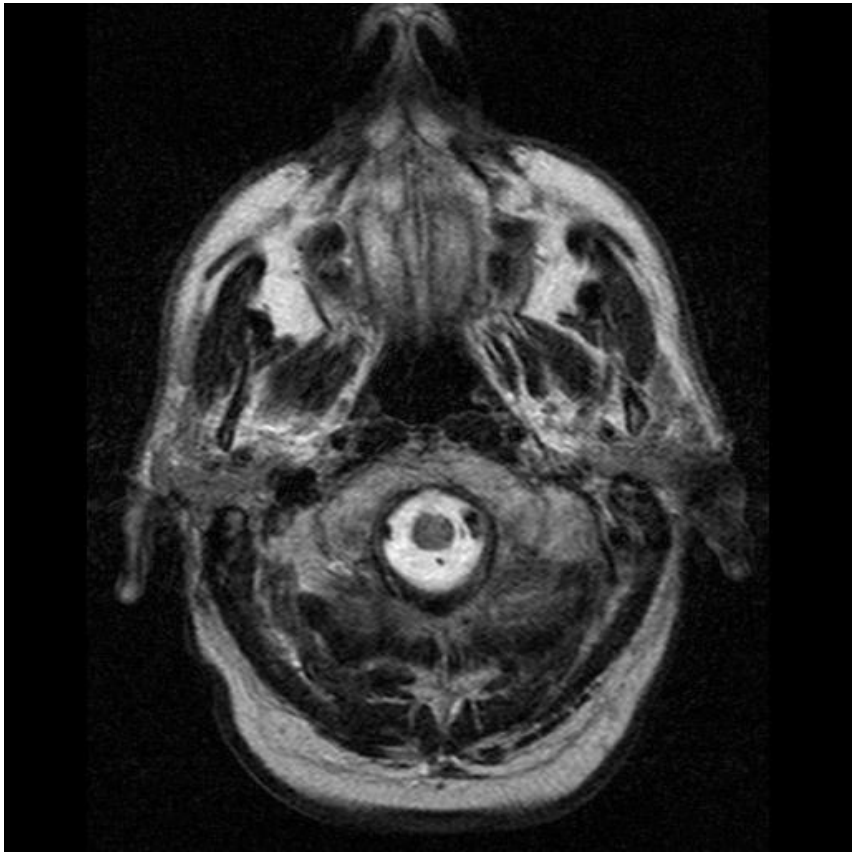
First Taste of a Fascinating Field

- MRI image segmentation is subject of ongoing research

- *k*-means is a good starting point, but not enough
  - Advanced clustering techniques such as the modified fuzzy k-means (MFCM) clustering technique
  - Packages in R specialized for medical image analysis

    http://cran.r-project.org/web/views/MedicalImaging.html

3D Reconstruction



- 3D reconstruction from 2D cross sectional MRI images
- Important in the medical field for diagnosis, surgical planning and biological research

Comparing Methods

Comparison of Methods

	Used For	Pros	Cons
Linear Regression	Predicting a continuous outcome (salary, price, number of votes, etc.)	• Simple, well recognized • Works on small and large datasets	• Assumes a linear relationship $Y = a\log(X)+b$
Logistic Regression	Predicting a categorical outcome (Yes/No, Sell/Buy, Accept/Reject, etc.)	• Computes probabilities that can be used to assess confidence of the prediction	• Assumes a linear relationship

Comparison of Methods

	Used For	Pros	Cons
CART	Predicting a categorical outcome (quality rating 1--5, Buy/Sell/Hold) or a continuous outcome (salary, price, etc.)	• Can handle datasets without a linear relationship • Easy to explain and interpret	• May not work well with small datasets
Random Forests	Same as CART	• Can improve accuracy over CART	• Many parameters to adjust • Not as easy to explain as CART

DIMENSIONLESS
TECHNOLOGY

THANK YOU

Dimensionless Tech. Pvt. Ltd.
B/303, Silicon Valley, Balewadi
Pune- 45, India

+918108094992
+919923170071
sales@dimensionless.in