#clustering with movies data<Hclust>

## Find which cluster Men in Black is in.

#Clustering with flowers dataset<Hclust>

### Select 3 clusters <Choosing clusters>

# Let's try this with an MRI image of the brain<KMM clust>

##The clustering is done for a healthy man

# Apply to a test image

##Clusters found in the above example will be applied to a test image<person show has tumors in brain>

# Unit 6 - Introduction to Clustering

# Video 6

#Data can be download from 🡪  <http://files.grouplens.org/datasets/movielens/ml-100k/u.item>

E:\A N A L Y T I C S\Algorithmic Thinking\The Analytics edge

# After following the steps in the video, load the data into R

movies = read.table("movieLens.txt", header=FALSE, sep="|",quote="\"")

str(movies)

# Add column names

colnames(movies) = c("ID", "Title", "ReleaseDate", "VideoReleaseDate", "IMDB", "Unknown", "Action", "Adventure", "Animation", "Childrens", "Comedy", "Crime", "Documentary", "Drama", "Fantasy", "FilmNoir", "Horror", "Musical", "Mystery", "Romance", "SciFi", "Thriller", "War", "Western")

str(movies)

# Remove unnecessary variables

movies$ID = NULL

movies$ReleaseDate = NULL

movies$VideoReleaseDate = NULL

movies$IMDB = NULL

# Remove duplicates

movies = unique(movies)

# Take a look at our data again:

str(movies)

# Video 7

# Compute distances

distances = dist(movies[2:20], method = "euclidean")

# Hierarchical clustering

clusterMovies = hclust(distances, method = "ward.d")

# Plot the dendrogram

plot(clusterMovies)

# Assign points to clusters

clusterGroups = cutree(clusterMovies, k = 10)

#Now let's figure out what the clusters are like.

# Let's use the tapply function to compute the percentage of movies in each genre and cluster

tapply(movies$Action, clusterGroups, mean)

tapply(movies$Romance, clusterGroups, mean)

# We can repeat this for each genre. If you do, you get the results in ClusterMeans.ods

# Find which cluster Men in Black is in.

subset(movies, Title=="Men in Black (1997)")

clusterGroups[257]

# Create a new data set with just the movies from cluster 2

cluster2 = subset(movies, clusterGroups==2)

# Look at the first 10 titles in this cluster:

cluster2$Title[1:10]

#Clustering with flowers dataset

# Unit 6 - Recitation

# Video 2

flower = read.csv("flower.csv", header=FALSE)

str(flower)

# Change the data type to matrix

flowerMatrix = as.matrix(flower)

str(flowerMatrix)

# Turn matrix into a vector

flowerVector = as.vector(flowerMatrix)

str(flowerVector)

flowerVector2 = as.vector(flower)

str(flowerVector2)

# Compute distances

distance = dist(flowerVector, method = "euclidean")

# Video 3

# Hierarchical clustering

clusterIntensity = hclust(distance, method="ward.D")

# Plot the dendrogram

plot(clusterIntensity)

# Select 3 clusters

rect.hclust(clusterIntensity, k = 3, border = "red")

flowerClusters = cutree(clusterIntensity, k = 3)

flowerClusters

# Find mean intensity values

tapply(flowerVector, flowerClusters, mean)

# Plot the image and the clusters

dim(flowerClusters) = c(50,50)

image(flowerClusters, axes = FALSE)

# Original image

image(flowerMatrix,axes=FALSE,col=grey(seq(0,1,length=256)))

# Video 4

# Let's try this with an MRI image of the brain

healthy = read.csv("healthy.csv", header=FALSE)

healthyMatrix = as.matrix(healthy)

str(healthyMatrix)

# Plot image

image(healthyMatrix,axes=FALSE,col=grey(seq(0,1,length=256)))

# Hierarchial clustering

healthyVector = as.vector(healthyMatrix)

distance = dist(healthyVector, method = "euclidean")

# We have an error - why?

str(healthyVector)

# Video 5

# Specify number of clusters

k = 5

# Run k-means

set.seed(1)

KMC = kmeans(healthyVector, centers = k, iter.max = 1000)

str(KMC)

# Extract clusters

healthyClusters = KMC$cluster

KMC$centers[2]

# Plot the image with the clusters

dim(healthyClusters) = c(nrow(healthyMatrix), ncol(healthyMatrix))

image(healthyClusters, axes = FALSE, col=rainbow(k))

# Video 6

# Apply to a test image

tumor = read.csv("tumor.csv", header=FALSE)

tumorMatrix = as.matrix(tumor)

tumorVector = as.vector(tumorMatrix)

# Apply clusters from before to new image, using the flexclust package

install.packages("flexclust")

library(flexclust)

KMC.kcca = as.kcca(KMC, healthyVector)

tumorClusters = predict(KMC.kcca, newdata = tumorVector)

# Visualize the clusters

dim(tumorClusters) = c(nrow(tumorMatrix), ncol(tumorMatrix))

image(tumorClusters, axes = FALSE, col=rainbow(k))