**Coursera Data Science Course 4 Week 3**

**1. Hierarchical Clustering**

Clustering organizes things that are close into groups  
-How do we define close?  
-How do we group things?  
-How do we visualize the grouping?  
-How do we interpret the grouping?

**What is hierarchical clustering  
-an agglomerative approach** -find the closest two things  
 -put them together  
 -find next closest  
**-requires** -a defined distance  
 -a merging approach  
**-produces** -a tree showing how close things are to each other

**How do we define close?  
-most important step** - garbage in -> garbage out // need a measure that makes sense  
**-distance or similarity** - continuous – Euclidean distance // the literal, straight line distance between two things  
 - continuous – correlation similarity  
 - binary – manhattan distance   
 ^ pick one that makes sense  
Euclidean distance – can calculate the distance between Baltimore and DC using Pythagorean theorem, easily generalizable to higher dimensions  
Manhattan distance – think about manhattan, have to follow the streets/grids. Absolute sum of all the difference between the little coordinates.

**Example:**  
**set.seed(1234)  
par(mar = c(0, 0, 0, 0))  
x <- rnorm(12, mean = rep(1:3, each = 4), sd = 0.2)  
y <- rnorm(12, mean = rep(c(1, 2, 1), each = 4), sd = 0.2)  
plot(x, y, col = “blue”, pch = 19, cex = 2)  
text(x + 0.05, y + 0.05, labels = as.character(1:12))**

**dist function – (x, method):  
dataFrame <- data.frame(x = x, y = y)  
dist(dataFrame)**

**Algorithm**  
-> take the two points that are closest with each other and merge them together  
Eventually, **hclust:  
dataFrame <- data.frame(x = x, y = y)  
distxy <- dist(dataFrame)  
hClustering <- hclust(distxy)  
plot(hClustering)**

How many clusters there are is entirely dependent on what you want the cutoff to be

**If we want a prettier dendrograms:  
myplclust(hClustering, lab = rep(1:3, each = 4), lab.col = rep(1:3, each = 4))**

**Another question is how do we merge the points together?**-average linkage – takes the average of the x and y coordinates of the two points  
-complete linkage - take the farthest distance between two clusters and that’s the distance between two points

**Heatmap()  
dataFrame <- data.frame(x = x, y = y)  
set.seed(143)  
dataMatrix <- as.matrix(dataFrame)[sample(1:12), ]  
heatmap(dataMatrix)** //reorganizes data by closeness

However, notes:  
-picture may be unstable  
-change a few points  
-missing values   
-pick different distances  
-change merging strategy  
-change the scale of points for one variable  
but it is deterministic

**2. K-Means Clustering**

**A partitioning approach** -fix a number of clusters  
 -get “centroids” of each cluster // the “center of gravity” for each cluster  
 -assign things to closest centroid  
 -recalculate centroids

**Requires** -a defined distance metric  
 -a number of clusters  
 -an initial guess as to cluster centroids

**Produces** -final estimate of cluster centroids  
 -an assignment of each point to clusters

**Example:**

**Set.seed(1234)  
par(mar = c(0, 0, 0, 0))  
x <- rnorm(12, mean = rep(1:3, each = 4), sd = 0.2)  
y <- rnorm(12, mean = rep(c(1, 2, 1) each = 4), sd = 0.2)  
plot(x , y, col = “blue”, pch = 19, cex = 2)  
text(x + 0.05, y + 0.05, labels = as.character(1:12))**

1. Create centroid  
2. Assign data points to closest centroid  
3. Recalculate centroids – take mean of that cluster  
4. Reassign values  
5. Update centroids

**Kmeans()** // important parameters: x, centers, iter.max, nstart  
**dataFrame <- data.frame(x, y)  
kmeansObj <- kmeans(dataFrame, centers = 3)  
name(kmeansObj)**

**Par(mar = rep(0.2, 4))  
plot(x, y, col = kmeansObj$cluster, pch = 19, cex = 2)  
points(kmeansObj$centers, col = 1:3, pch = 3, cex = 3, lwd = 3)**

**Heatmaps (again!)  
set.seed(1234)  
dataMatrix <- as.matrix(dataFrame)[sample(1:12), ]  
kmeansObj2 <- kmeans(dataMatrix, centers = 3)  
par(mfrow = c(1, 2), mar = c(2, 4, 0.1, 0.1))  
image(t(dataMatrix)[, nrow(dataMatrix):1], yaxt = “n”)  
image(t(dataMatrix)[, order(kmeansObj$cluster)], yaxt = “n”)**

K-means requires a number of clusters-Pick clusters by eye/intuition AND/OR cross validation/information theory  
-Determining the number of clusters

Not deterministic- can have different # of clusters

**3. Dimension Reduction**

Example: Random matrix data  
**set.seed(12345)  
par(mar = rep(0.2, 4))  
dataMatrix <- matrix(rnorm(400), nrow = 40)  
image(1:10, 1:40, t(dataMatrix)[, nrow(dataMatrix):1])**Hierarchical cluster analysis  
**par(mar = rep(0.2, 4))  
heatmap(dataMatrix)**

We can add a pattern to the data set:  
**for(i in 1:40) {  
 coinFlip <- rbinom(1, size = 1, prob = 0.5)  
 if (coinFlip) {  
 dataMatrix[i, ] <- dataMatrix[i, ] + rep(c(0, 3), each = 5)  
 }  
}  
image(1:10, 1:40, t(dataMatrix)[, nrow(dataMatrix):1])**

Patterns:

**Hh <- hclust(dist(dataMatrix))  
dataMatrixOrdered <- dataMatrix[hh$order, ]  
par(mfrow = c(1, 3))  
image(t(dataMatrixOrder)[, nrow(dataMatrixOrdered):1])  
plot(rowMeans(dataMatrixOrdered), 40:1, xlab = “Row Mean”, ylab = “Row”, pch = 19)  
plot(colMeans(dataMatrixOrdered), xlab = “Column”, ylab = “Column Mean”, pch = 19)**//plotting the mean shows the pattern

**Related Problems**-Multivariate variables  
-find a new set of multivariate variables that are uncorrelated and explain variance  
-if all variables are together in one matrix, find the best matrix created with fewer variable that explains the original data  
-two goals: STATISTICAL, then DATA COMPRESSION