DSC 520 Week 9

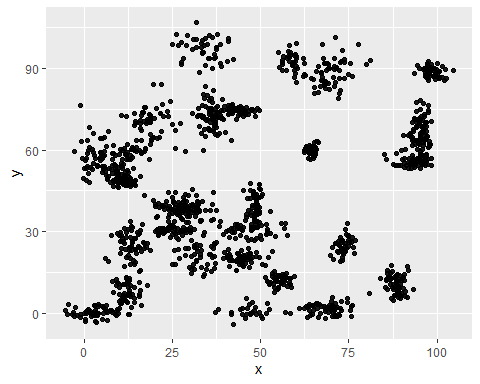
Dan Wiltse

Feb 9th 2020

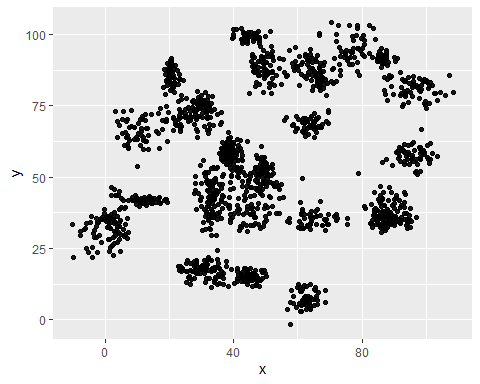
##9.2 - Intro to Machine Learning

## a. Plot the data from each dataset using a scatter plot.

ggplot(data = data9.2binary, aes(x = x, y = y)) +  
 geom\_point(position = "jitter", alpha = 0.9)



ggplot(data = data9.2trinary, aes(x = x, y = y)) +  
 geom\_point(position = "jitter", alpha = 0.9)

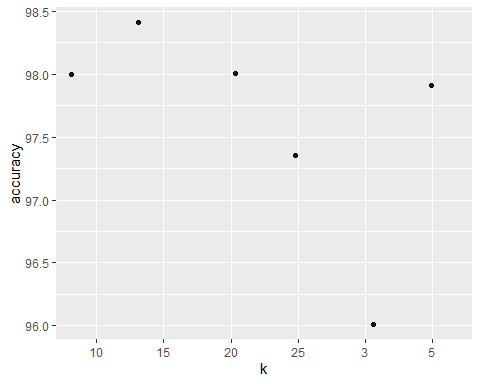


## b. The k nearest neighbors algorithm categorizes an input value by looking at the labels for the k nearest points and assigning a category based on the most common label. In this problem, you will determine which points are nearest by calculating the Euclidean distance between two points. Fit a k nearest neighbors model for each dataset for k=3, k=5, k=10, k=15, k=20, and k=25. Compute the accuracy of the resulting models for each value of k. Plot the results in a graph where the x-axis is the different values of k and the y-axis is the accuracy of the model.

k <- c('3','5', '10', '15', '20', '25')  
accuracy <- c(96,98, 98, 98.667,98,97.33)  
plotdata <- data.frame(k, accuracy, stringsAsFactors=FALSE)  
head(plotdata)

## k accuracy  
## 1 3 96.000  
## 2 5 98.000  
## 3 10 98.000  
## 4 15 98.667  
## 5 20 98.000  
## 6 25 97.330

ggplot(data = plotdata, aes(x = k, y = accuracy)) +  
 geom\_point(position = "jitter", alpha = 0.9)



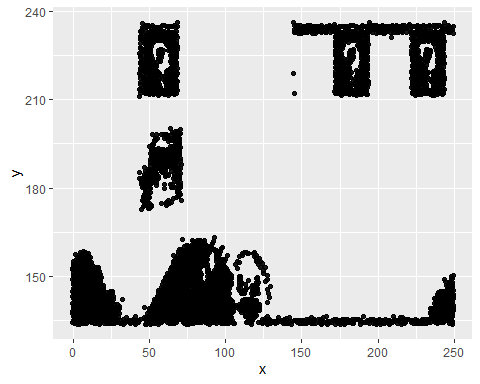
## c. In later lessons, you will learn about linear classifiers. These algorithms work by defining a decision boundary that separates the different categories. Looking back at the plots of the data, do you think a linear classifier would work well on these datasets?

No

##9.3 - Clustering

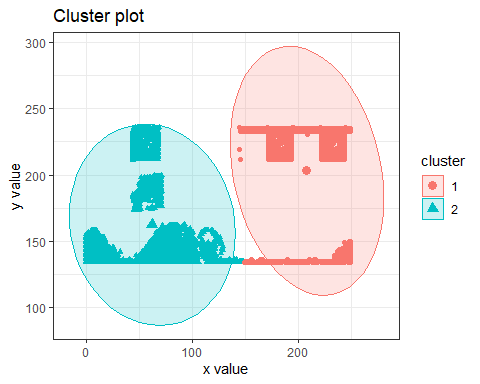
## a. Plot the dataset using a scatter plot.

ggplot(data = data9.3, aes(x = x, y = y)) +  
 geom\_point(position = "jitter", alpha = 0.9)

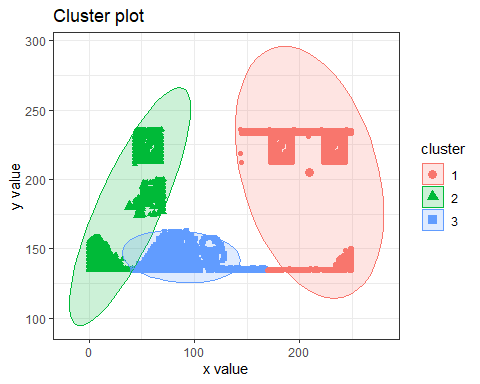


## b. Fit the dataset using the k-means algorithm from k=2 to k=12. Create a scatter plot of the resultant clusters for each value of k.

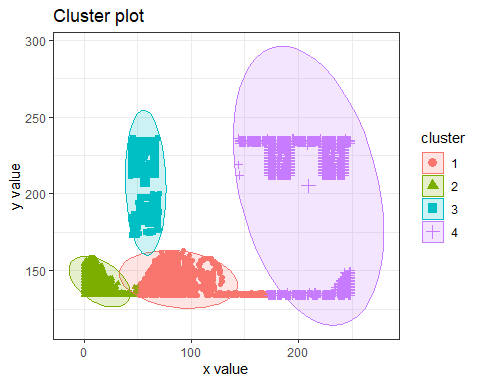
#k=2  
cluster2 <- kmeans(data9.3[,1:2], 2)  
set.seed(123)  
km.res <- kmeans(x = data9.3, centers = 2, nstart = 25)  
  
fviz\_cluster(object = km.res, data = data9.3, geom = "point",  
 choose.vars = c("x", "y"), stand = FALSE,   
 ellipse.type = "norm") + theme\_bw()



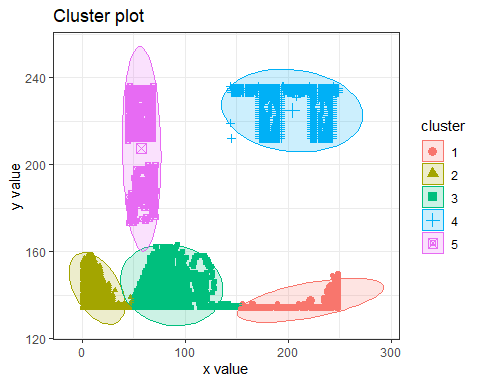
#k=3  
cluster2 <- kmeans(data9.3[,1:2], 3)  
set.seed(123)  
km.res <- kmeans(x = data9.3, centers = 3, nstart = 25)  
  
fviz\_cluster(object = km.res, data = data9.3, geom = "point",  
 choose.vars = c("x", "y"), stand = FALSE,   
 ellipse.type = "norm") + theme\_bw()



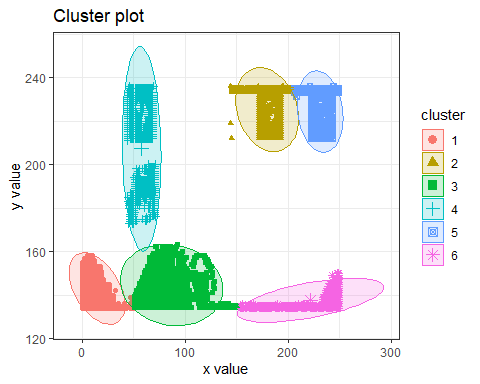
#k=4  
cluster2 <- kmeans(data9.3[,1:2], 4)  
set.seed(123)  
km.res <- kmeans(x = data9.3, centers = 4, nstart = 25)  
  
fviz\_cluster(object = km.res, data = data9.3, geom = "point",  
 choose.vars = c("x", "y"), stand = FALSE,   
 ellipse.type = "norm") + theme\_bw()



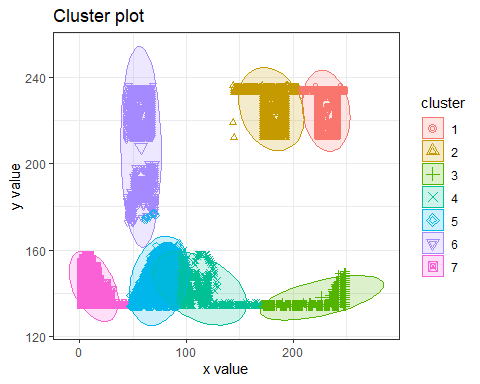
#k=5  
cluster2 <- kmeans(data9.3[,1:2], 5)  
set.seed(123)  
km.res <- kmeans(x = data9.3, centers = 5, nstart = 25)  
  
fviz\_cluster(object = km.res, data = data9.3, geom = "point",  
 choose.vars = c("x", "y"), stand = FALSE,   
 ellipse.type = "norm") + theme\_bw()



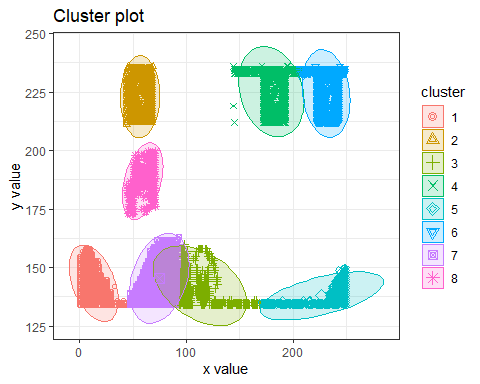
#k=6  
cluster2 <- kmeans(data9.3[,1:2], 6)  
set.seed(123)  
km.res <- kmeans(x = data9.3, centers = 6, nstart = 25)  
  
fviz\_cluster(object = km.res, data = data9.3, geom = "point",  
 choose.vars = c("x", "y"), stand = FALSE,   
 ellipse.type = "norm") + theme\_bw()



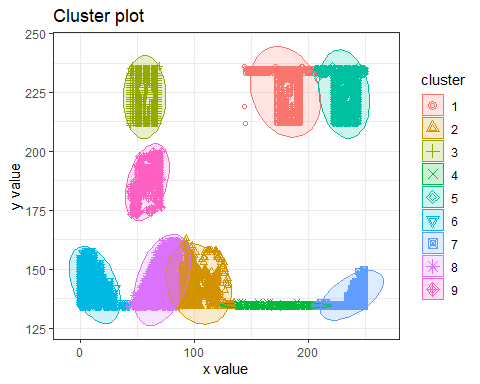
#k=7  
cluster2 <- kmeans(data9.3[,1:2], 7)  
set.seed(123)  
km.res <- kmeans(x = data9.3, centers = 7, nstart = 25)  
  
fviz\_cluster(object = km.res, data = data9.3, geom = "point",  
 choose.vars = c("x", "y"), stand = FALSE,   
 ellipse.type = "norm") + theme\_bw()



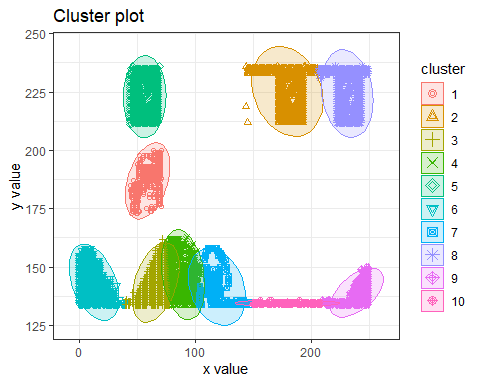
#k=8  
cluster2 <- kmeans(data9.3[,1:2], 8)  
set.seed(123)  
km.res <- kmeans(x = data9.3, centers = 8, nstart = 25)  
  
fviz\_cluster(object = km.res, data = data9.3, geom = "point",  
 choose.vars = c("x", "y"), stand = FALSE,   
 ellipse.type = "norm") + theme\_bw()



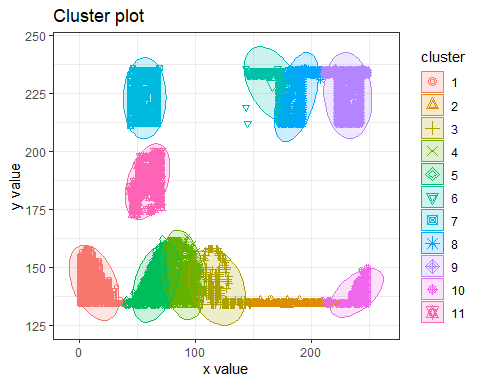
#k=9  
cluster2 <- kmeans(data9.3[,1:2], 9)  
set.seed(123)  
km.res <- kmeans(x = data9.3, centers = 9, nstart = 25)  
  
fviz\_cluster(object = km.res, data = data9.3, geom = "point",  
 choose.vars = c("x", "y"), stand = FALSE,   
 ellipse.type = "norm") + theme\_bw()



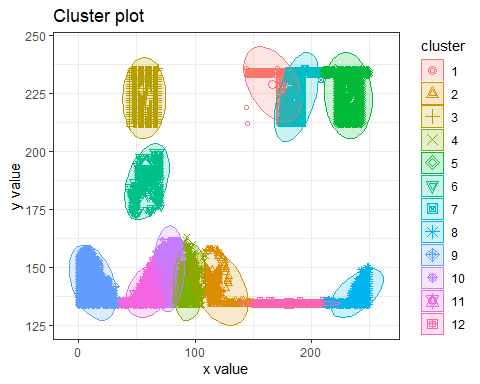
#k=10  
cluster2 <- kmeans(data9.3[,1:2], 10)  
set.seed(123)  
km.res <- kmeans(x = data9.3, centers = 10, nstart = 25)  
  
fviz\_cluster(object = km.res, data = data9.3, geom = "point",  
 choose.vars = c("x", "y"), stand = FALSE,   
 ellipse.type = "norm") + theme\_bw()



#k=11  
cluster2 <- kmeans(data9.3[,1:2], 11)  
set.seed(123)  
km.res <- kmeans(x = data9.3, centers = 11, nstart = 25)  
  
fviz\_cluster(object = km.res, data = data9.3, geom = "point",  
 choose.vars = c("x", "y"), stand = FALSE,   
 ellipse.type = "norm") + theme\_bw()



#k=12  
cluster2 <- kmeans(data9.3[,1:2], 12)  
set.seed(123)  
km.res <- kmeans(x = data9.3, centers = 12, nstart = 25)  
  
fviz\_cluster(object = km.res, data = data9.3, geom = "point",  
 choose.vars = c("x", "y"), stand = FALSE,   
 ellipse.type = "norm") + theme\_bw()



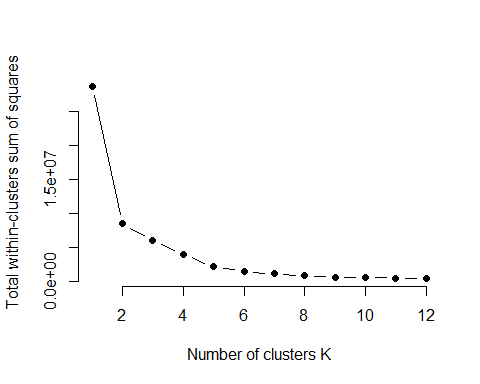
## c1. As k-means is an unsupervised algorithm, you cannot compute the accuracy as there are no correct values to compare the output to. Instead, you will use the average distance from the center of each cluster as a measure of how well the model fits the data. To calculate this metric, simply compute the distance of each data point to the center of the cluster it is assigned to and take the average value of all of those distances.

##c2. Calculate this average distance from the center of each cluster for each value of k and plot it as a line chart where k is the x-axis and the average distance is the y-axis.

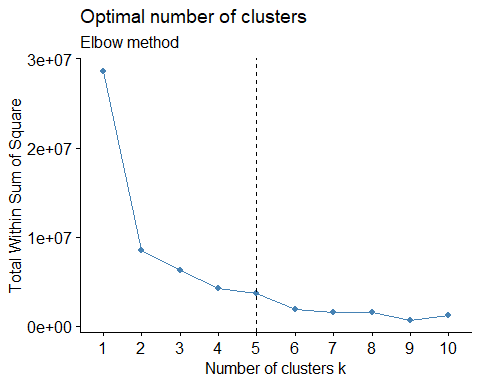
#Elbow Method for finding the optimal number of clusters  
set.seed(123)  
# Compute and plot wss for k = 2 to k = 12  
k.max <- 12  
data <- data9.3  
wss <- sapply(1:k.max,   
 function(k){kmeans(data9.3, k, nstart=50,iter.max = 12 )$tot.withinss})  
wss

## [1] 28608984.7 8443681.1 6014377.9 4009678.4 2171612.8 1519043.4  
## [7] 1102869.9 853160.1 647331.9 554632.2 500038.6 450089.2

plot(1:k.max, wss,  
 type="b", pch = 19, frame = FALSE,   
 xlab="Number of clusters K",  
 ylab="Total within-clusters sum of squares")



##Alternative package  
  
# Elbow method  
fviz\_nbclust(data9.3, kmeans, method = "wss") +  
 geom\_vline(xintercept = 5, linetype = 2)+  
 labs(subtitle = "Elbow method")



## d. One way of determining the “right” number of clusters is to look at the graph of k versus average distance and finding the “elbow point”. Looking at the graph you generated in the previous example, what is the elbow point for this dataset?

I believe the elbow point for the optimal number of clusters is 5 or possibly 6, as that is where the bend of the “elbow” appears to level out.