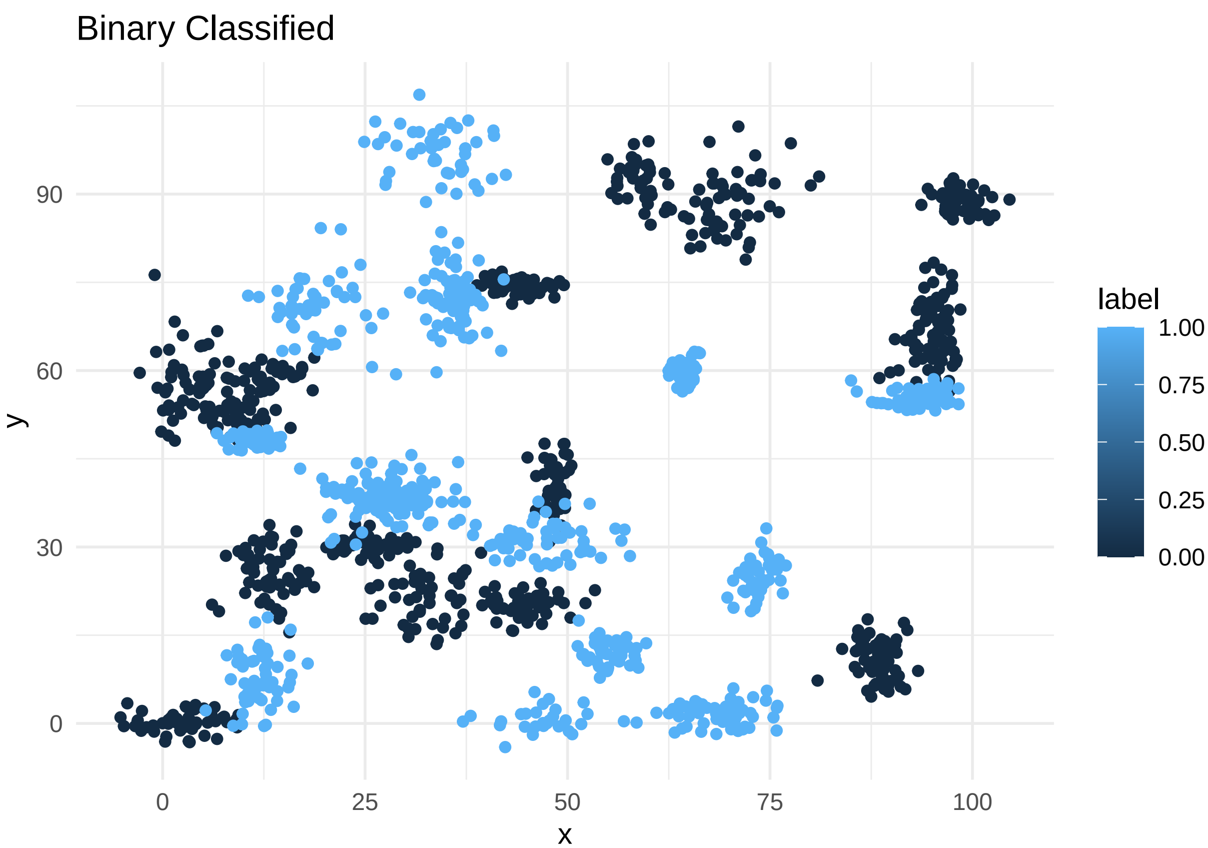
weeks 11 and 12

2022-06-04

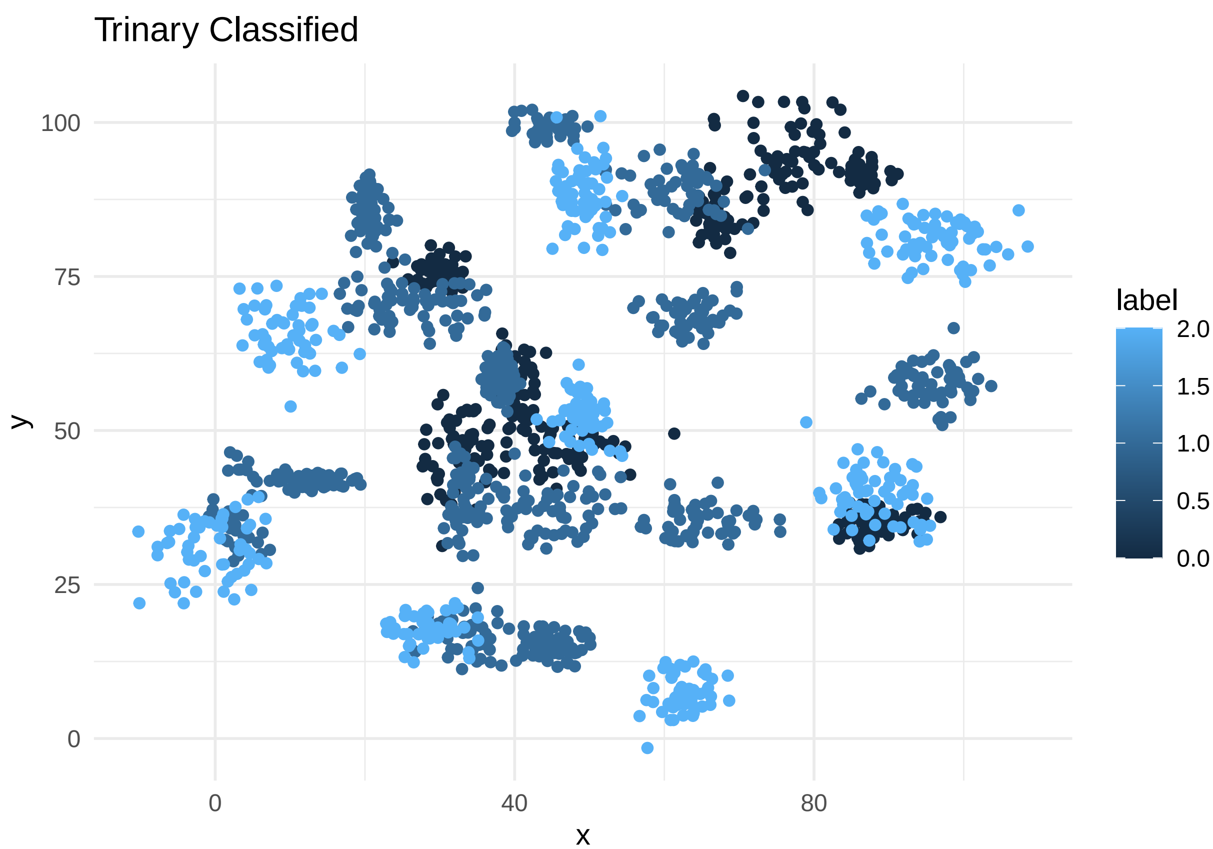
PART 1 - K-Nearest Neighbor

Plot the data from each dataset using a scatter plot.

```
setwd("/Users/marianamacdonald/Documents/DATA SCIENCE/DSC 520/Statistics R/Week 2/dsc520")
BinaryClassifier <- read.csv("data/binary-classifier-data.csv", stringsAsFactors = FALSE)
TrinaryClassifier <- read.csv("data/trinary-classifier-data.csv", stringsAsFactors = FALSE)
library(ggplot2)
theme_set(theme_minimal())
ggplot(BinaryClassifier, aes(x=x, y=y)) +  geom_point(aes(color=label)) + ggtitle("Binary Classified")
```




```
ggplot(TrinaryClassifier, aes(x=x, y=y)) +  geom_point(aes(color=label)) + ggtitle("Trinary Classified")
```



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.

```
library(caret)
```

```
## Loading required package: lattice
```

```
library(class)


#BINARY CLASSIFIER

TrainingIndex=createDataPartition(BinaryClassifier$label, p=0.8)$Resample1
TrainingBinary=BinaryClassifier[TrainingIndex,]
TestBinary=BinaryClassifier[-TrainingIndex,]

kMatrix = c(3, 5, 10, 15, 20, 25)
accBin <- c()
index = 0
for (i in kMatrix) {
        index = index + 1
        kModBin <-  knn(train=TrainingBinary, test=TestBinary, cl=TrainingBinary$label, k=i )
        accBin[index] <- 100 * sum(TestBinary$label == kModBin)/NROW(TestBinary$label)
```

```
}

accuracy_df_binary <- as.data.frame(accBin)
accuracy_df_binary
```

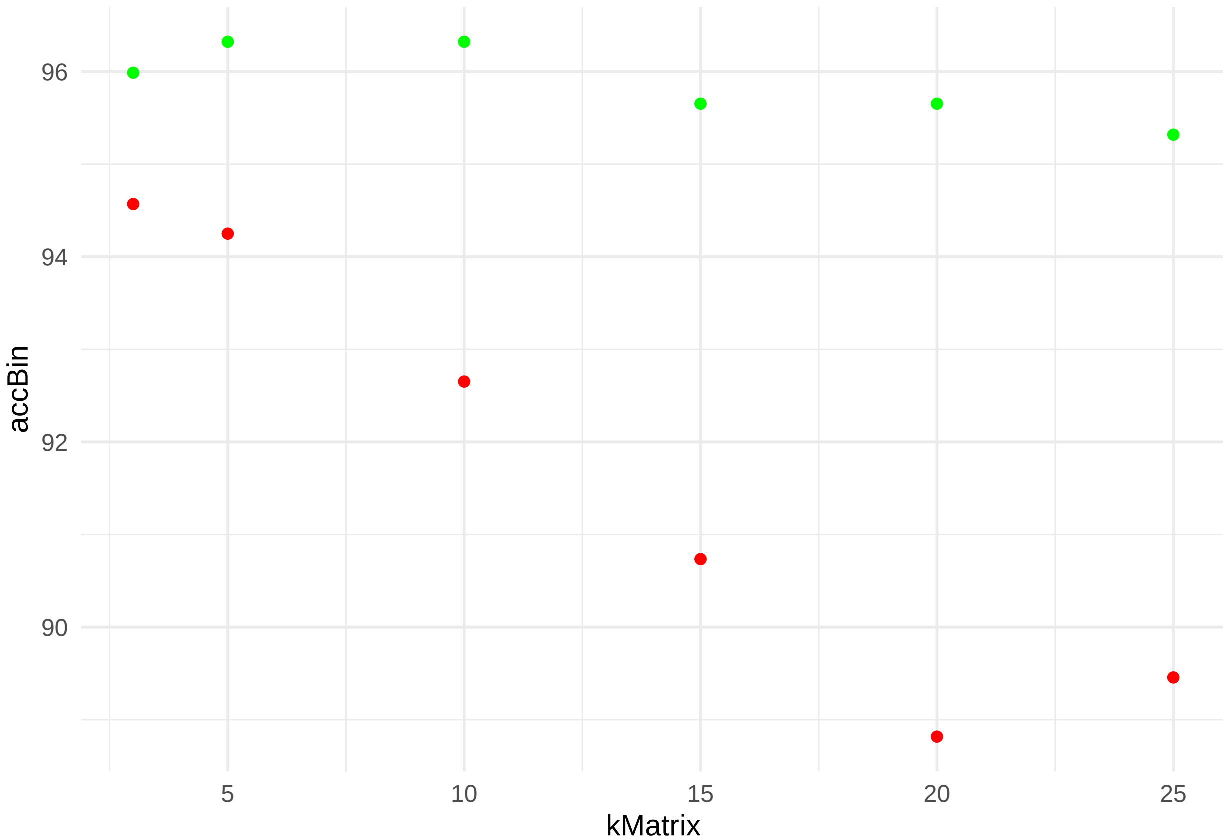
```
##     accBin
## 1 95.98662
## 2 96.32107
## 3 96.32107
## 4 95.65217
## 5 95.65217
## 6 95.31773
```

```
#TRINARY CLASSIFIER

TrainingIndex=createDataPartition(TrinaryClassifier$label, p=0.8)$Resample1
TrainingTrinary=TrinaryClassifier[TrainingIndex,]
TestTrinary=TrinaryClassifier[-TrainingIndex,]

kMatrix = c(3, 5, 10, 15, 20, 25)
accBin <- c()
index = 0
for (i in kMatrix) {
        index = index + 1
        kModBin <-  knn(train=TrainingTrinary, test=TestTrinary, cl=TrainingTrinary$label, k=i )
        accBin[index] <- 100 * sum(TestTrinary$label == kModBin)/NROW(TestTrinary$label)
}

accuracy_df_trinary <- as.data.frame(accBin)
accuracy_df_trinary
```

```
##     accBin
## 1 94.56869
## 2 94.24920
## 3 92.65176
## 4 90.73482
## 5 88.81789
## 6 89.45687
```

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.

```
library(ggplot2)
ggplot() +
geom_point(data=accuracy_df_binary, aes(x=kMatrix, y=accBin), color='green') +
geom_point(data=accuracy_df_trinary, aes(x=kMatrix, y=accBin), color='red')
```



Looking back at the plots of the data, do you think a linear classifier would work well on these datasets?

I don't think so because the plots show the clusters all over the place and not on a straight line.

How does the accuracy of your logistic regression classifier from last week compare? Why is the accuracy different between these two methods?

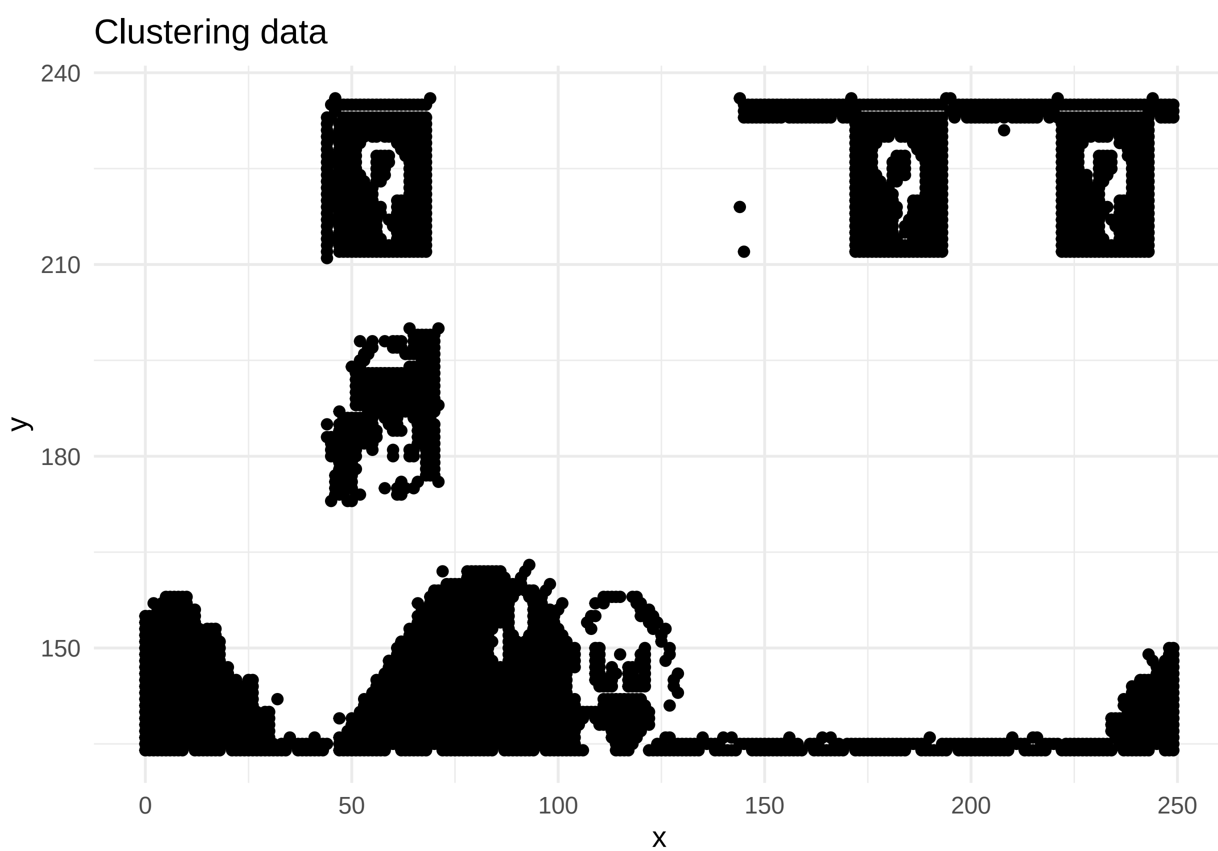
For the binary data set, last week's accuracy was about 58%. This week it is about 97%.

The accuracy is different because according to "Towards Data Science", KNN is a non-parametric model (do not often conform to a normal distribution, as they rely upon continuous data, rather than discrete values), where LR is a parametric model (finite number of parameters). KNN supports non-linear solutions where LR supports only linear solutions.

PART 2 - Clustering

Plot the dataset using a scatter plot.

```
setwd("/Users/marianamacdonald/Documents/DATA SCIENCE/DSC 520/Statistics R/Week 2/dsc520")
clustering_data <- read.csv("data/clustering-data.csv", stringsAsFactors = FALSE)
library(ggplot2)
theme_set(theme_minimal())
ggplot(clustering_data, aes(x=x, y=y)) +  geom_point() + ggtitle("Clustering data")
```



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.

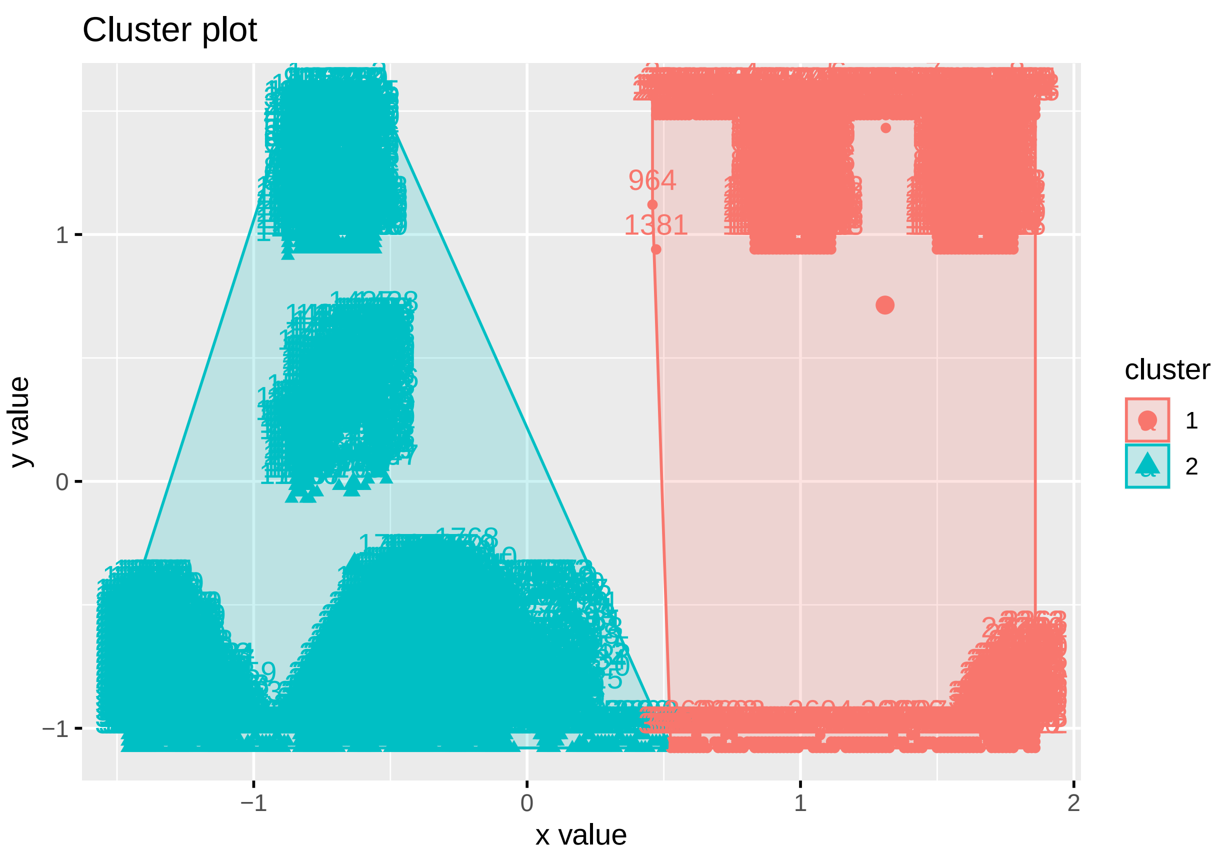
```
dataK2<- kmeans(x=clustering_data, centers=2)
dataK2$size
```

```
## [1] 2714 1308
```

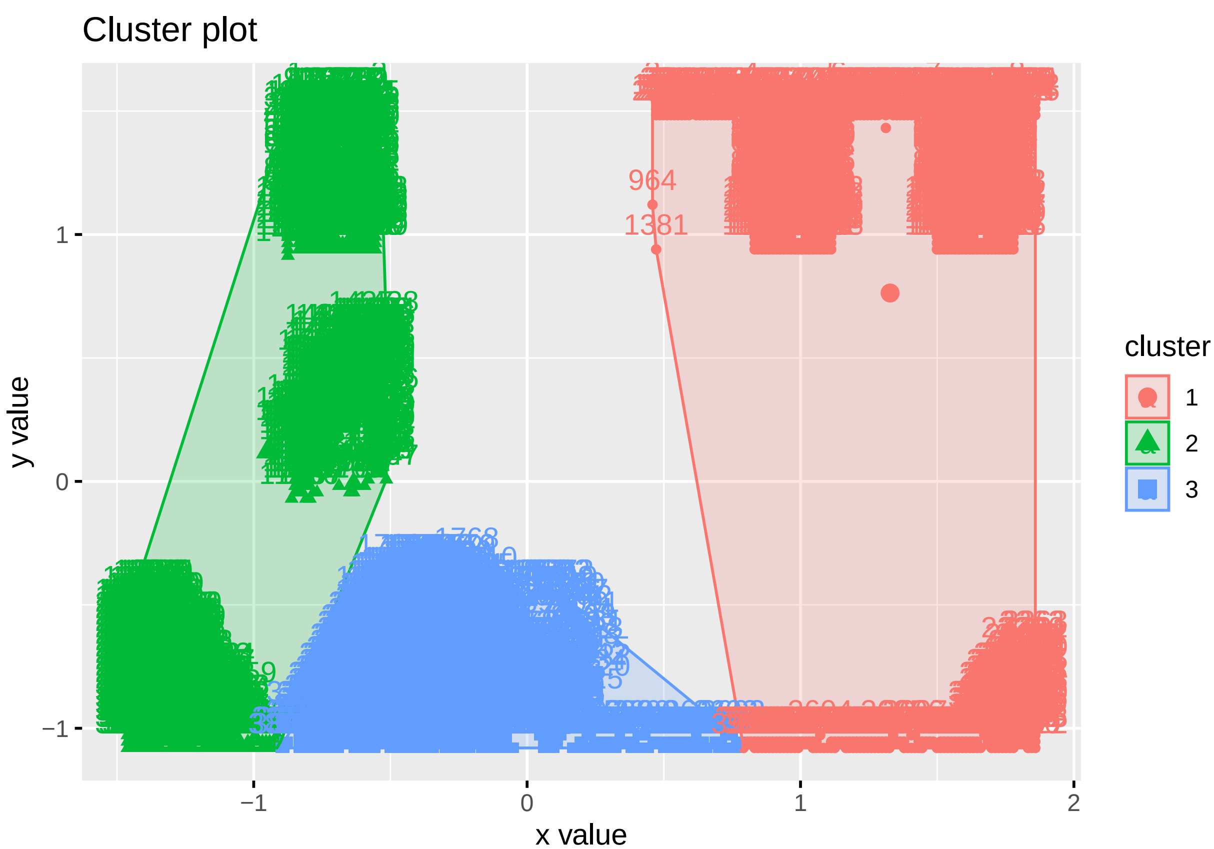
```
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

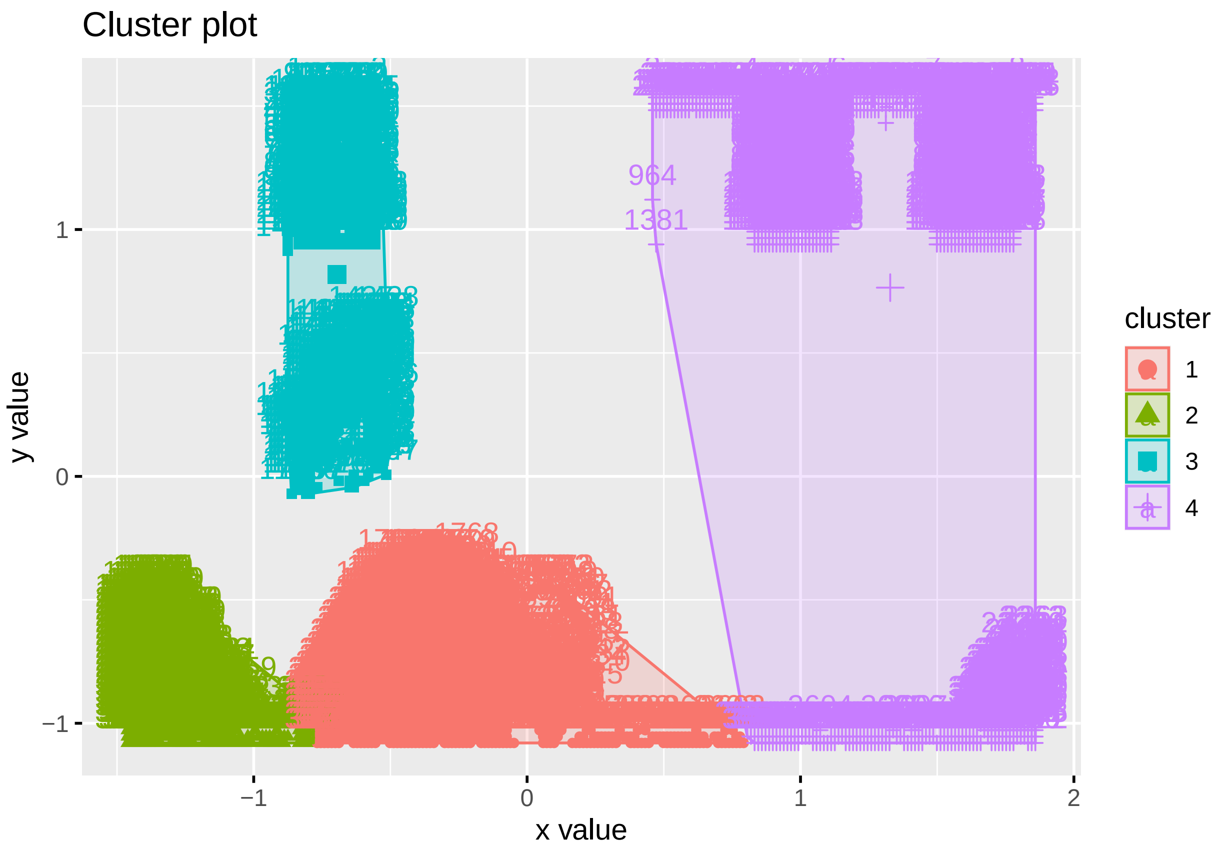
```
set.seed(123)
dataK2N25 <- kmeans(clustering_data, centers=2, nstart=25)
fviz_cluster(dataK2N25, data = clustering_data)
```



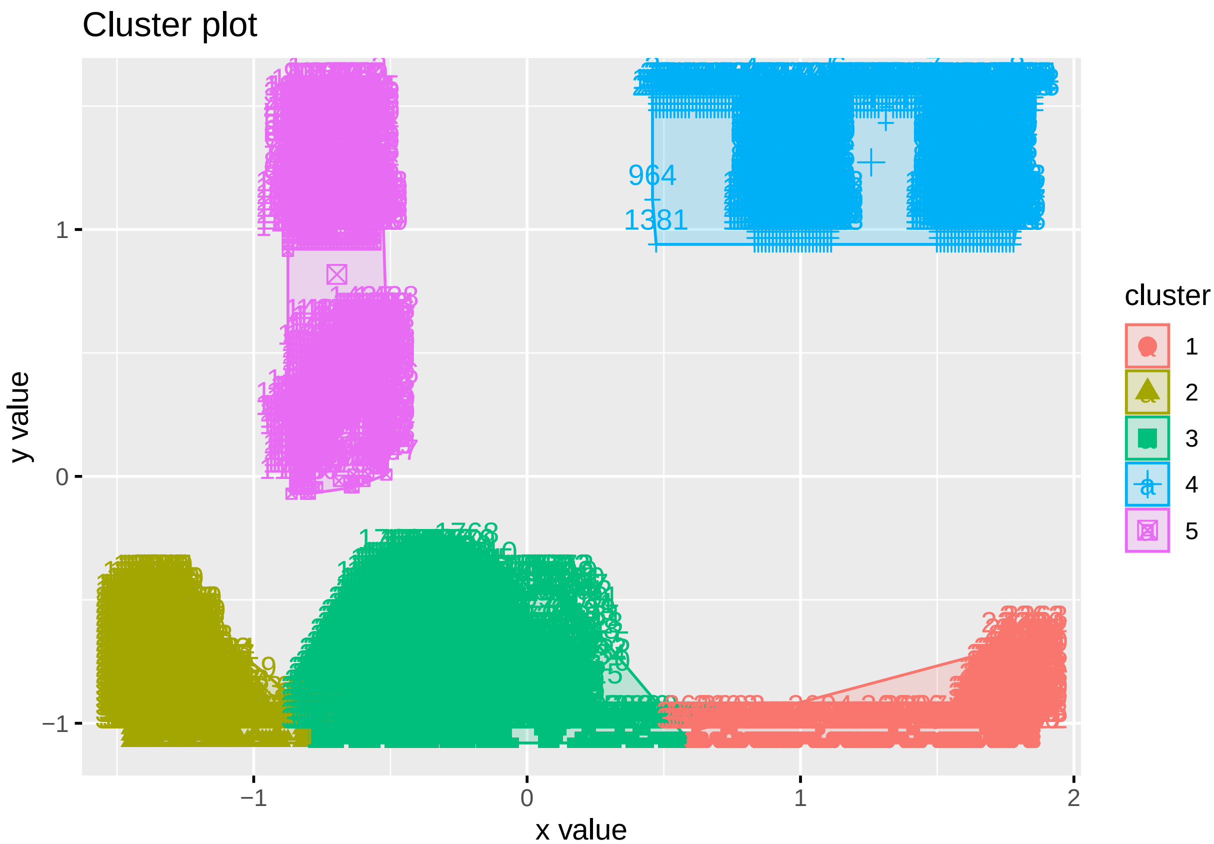

```
library(factoextra)
set.seed(123)
dataK3N25 <- kmeans(clustering_data, centers=3, nstart=25)
fviz_cluster(dataK3N25, data = clustering_data)
```



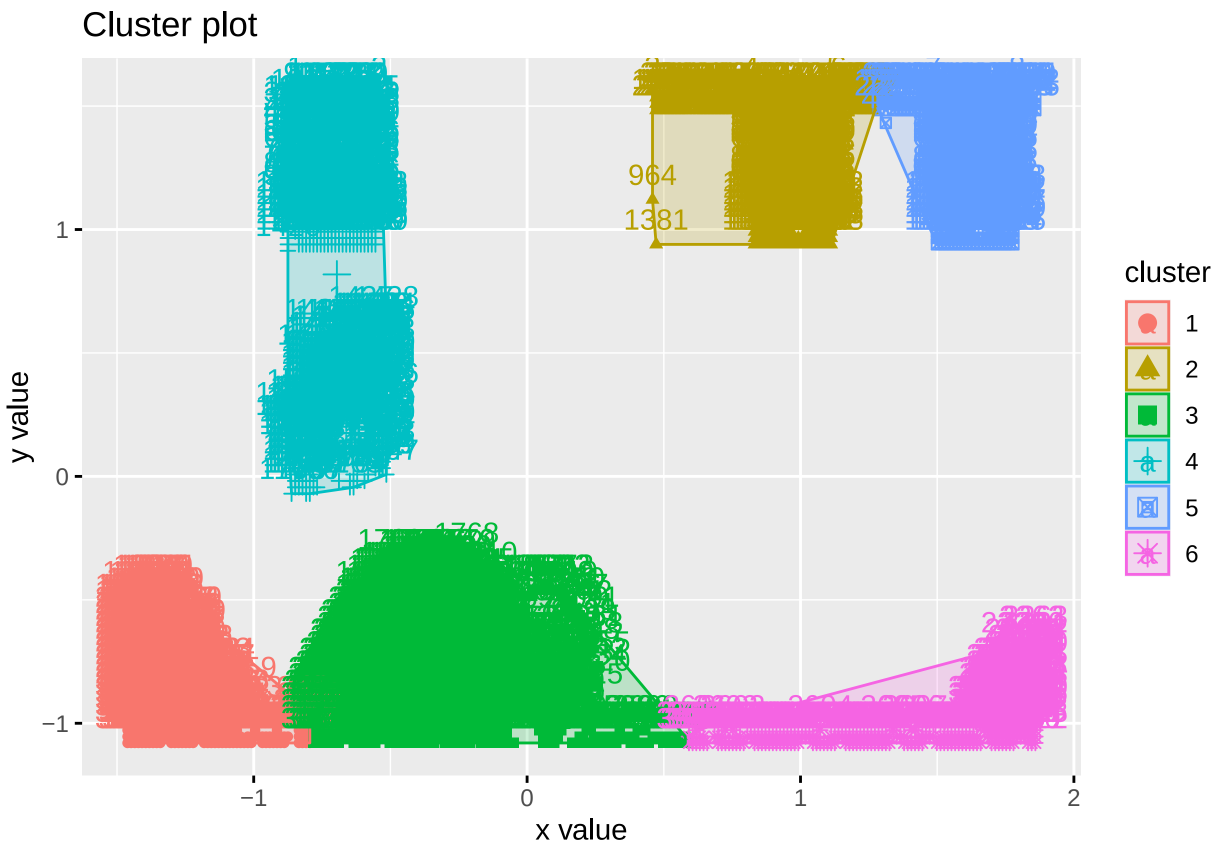

```
set.seed(123)
dataK4N25 <- kmeans(clustering_data, centers=4, nstart=25)
fviz_cluster(dataK4N25, data = clustering_data)
```



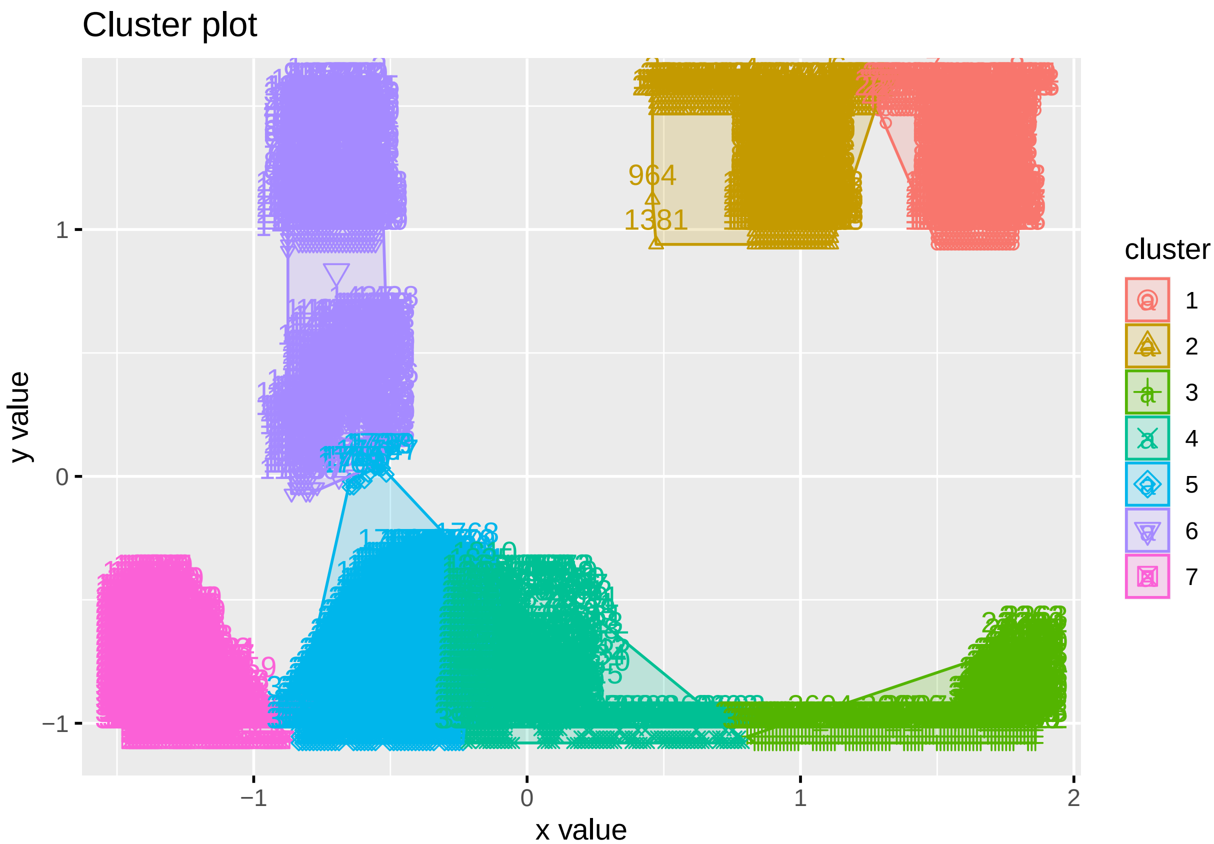

```r
set.seed(123)
dataK5N25 <- kmeans(clustering_data, centers=5, nstart=25)
fviz_cluster(dataK5N25, data = clustering_data)
```



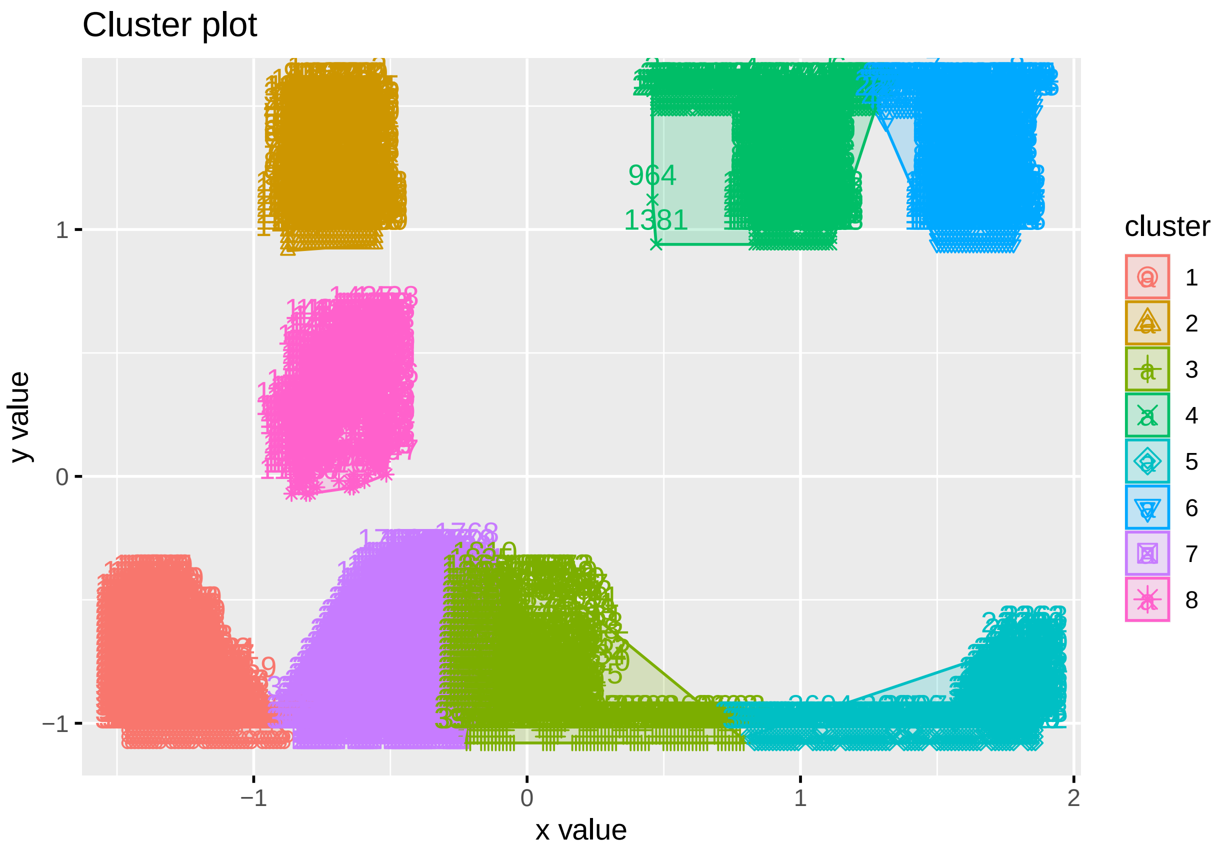

```r
set.seed(123)
dataK6N25 <- kmeans(clustering_data, centers=6, nstart=25)
fviz_cluster(dataK6N25, data = clustering_data)
```



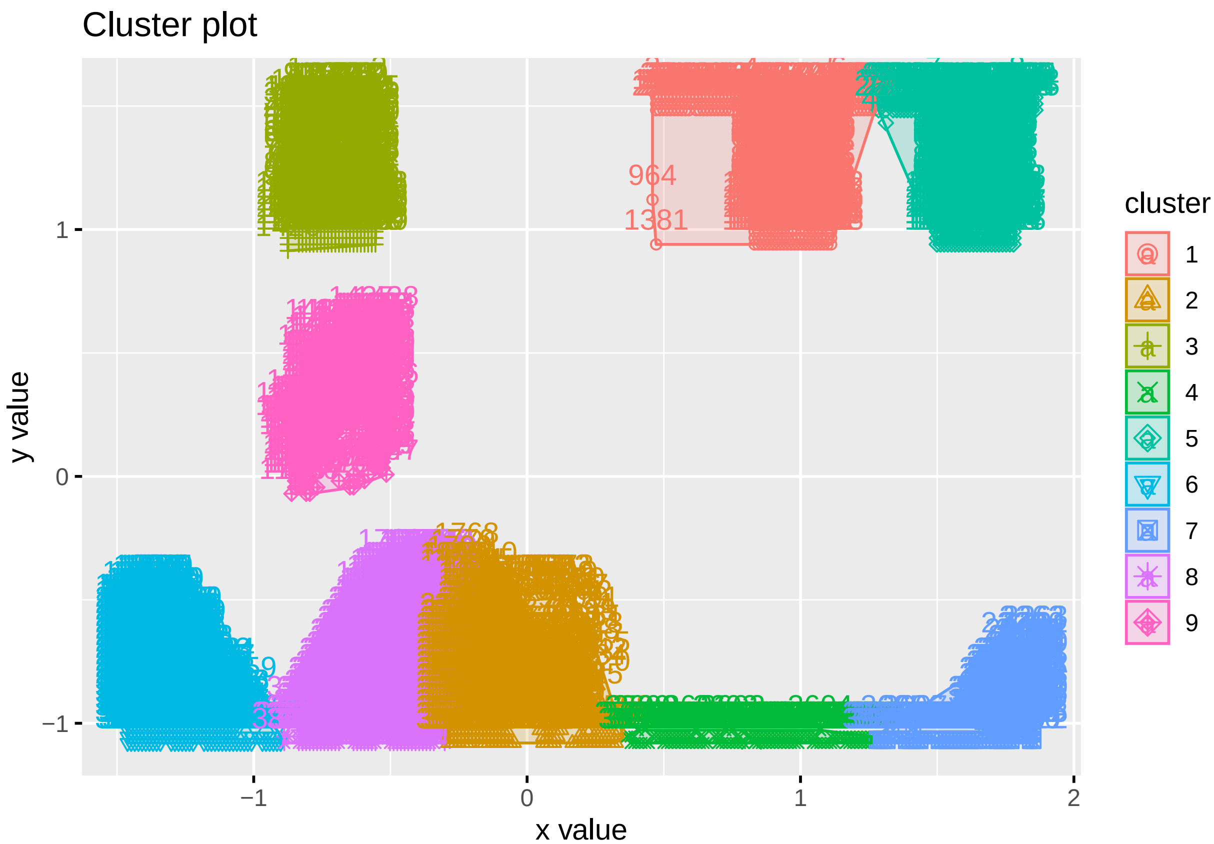

```
set.seed(123)
dataK7N25 <- kmeans(clustering_data, centers=7, nstart=25)
fviz_cluster(dataK7N25, data = clustering_data)
```



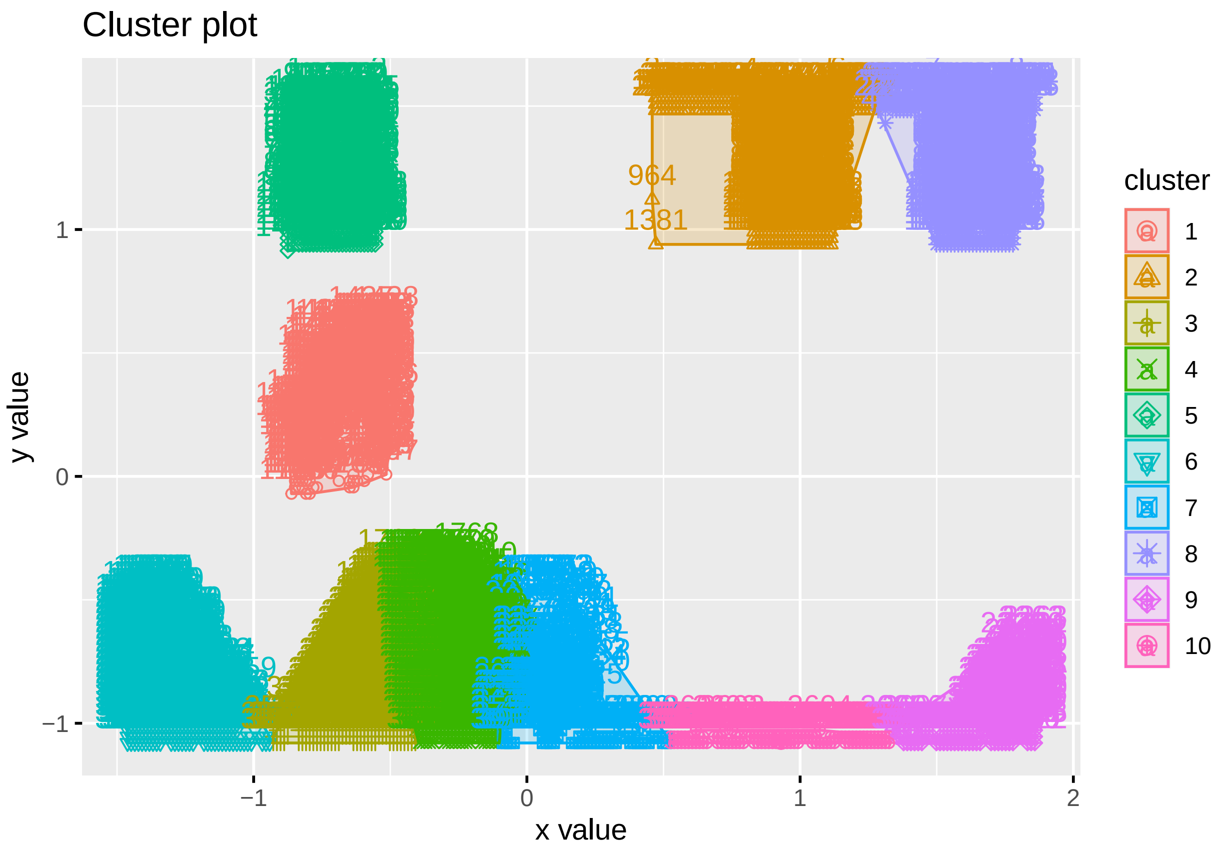

```
set.seed(123)
dataK8N25 <- kmeans(clustering_data, centers=8, nstart=25)
fviz_cluster(dataK8N25, data = clustering_data)
```



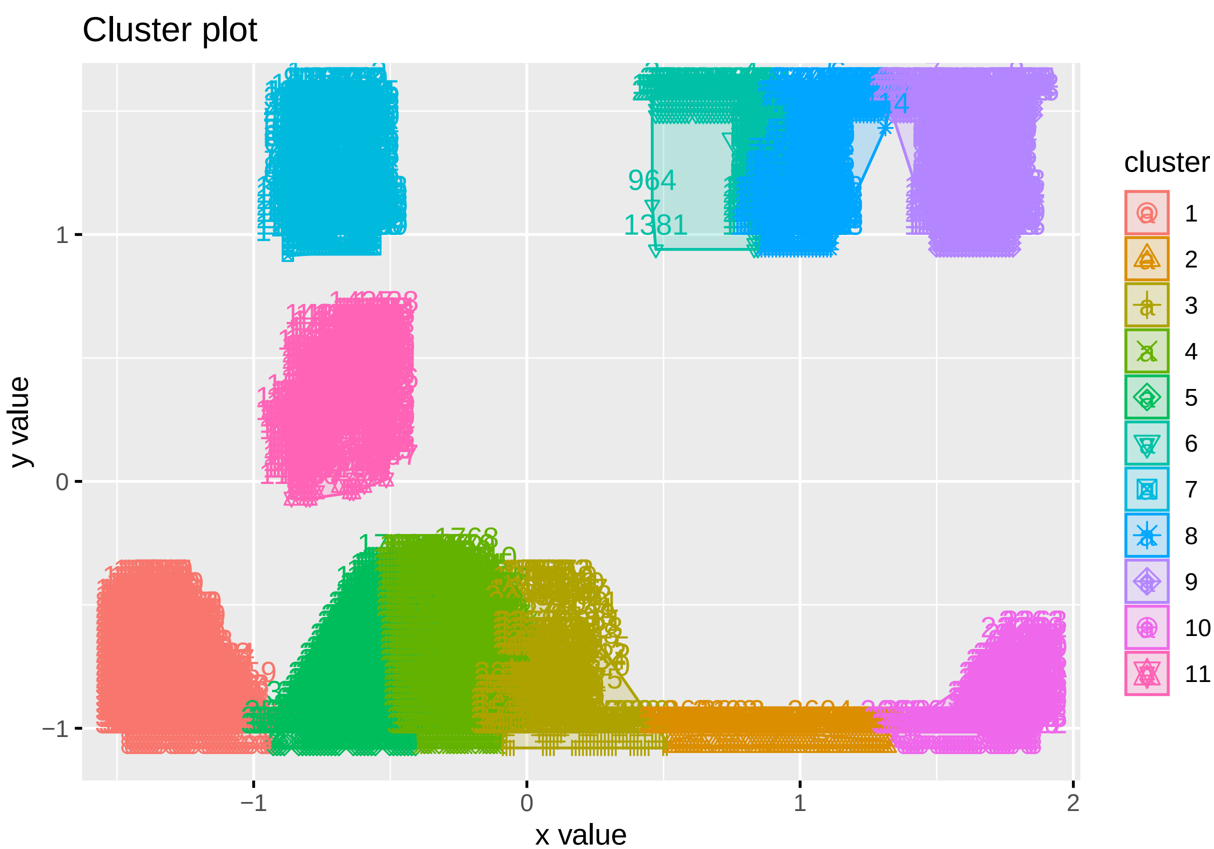

```r
set.seed(123)
dataK9N25 <- kmeans(clustering_data, centers=9, nstart=25)
fviz_cluster(dataK9N25, data = clustering_data)
```



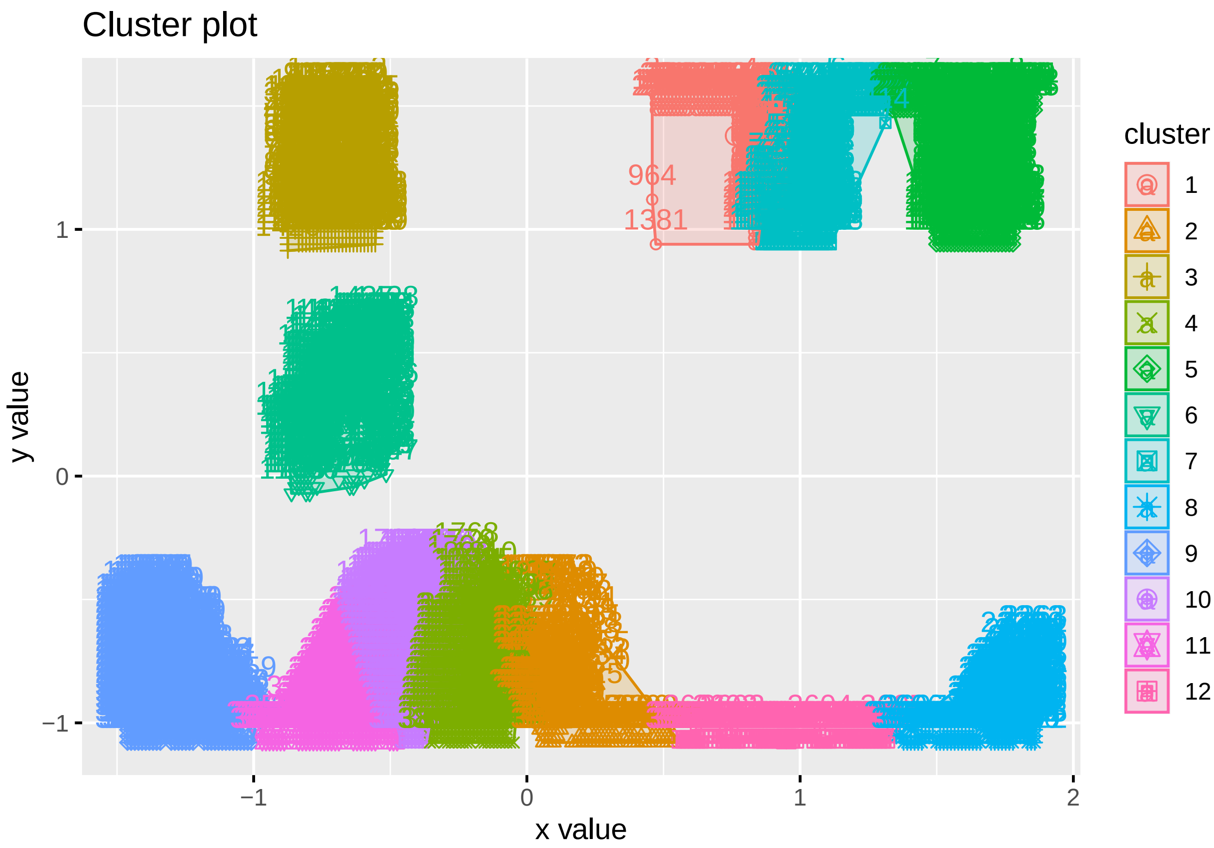

```
set.seed(123)
dataK10N25 <- kmeans(clustering_data, centers=10, nstart=25)
fviz_cluster(dataK10N25, data = clustering_data)
```




```
set.seed(123)
dataK11N25 <- kmeans(clustering_data, centers=11, nstart=25)
fviz_cluster(dataK11N25, data = clustering_data)
```




```
set.seed(123)
dataK12N25 <- kmeans(clustering_data, centers=12, nstart=25)
fviz_cluster(dataK12N25, data = clustering_data)
```



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.

```
library(useful)
dataBest<- FitKMeans(clustering_data, max.clusters=40, nstart=25, seed=123)
```

```
## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations
```

```
dataBest
```

```
##    Clusters     Hartigan AddCluster
## 1         2 9600.6137756       TRUE
## 2         3 1623.3382287       TRUE
## 3         4 2008.8599990       TRUE
## 4         5 3400.0117979       TRUE
```

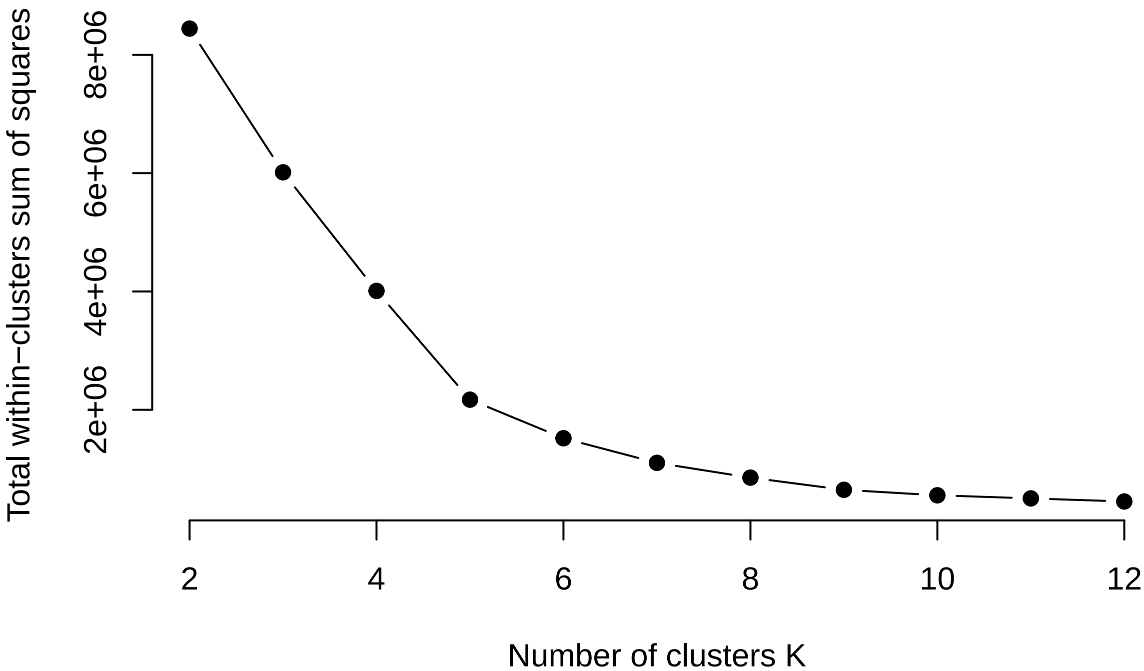
```
## 5          6 1725.2430400        TRUE
## 6          7 1515.0802384        TRUE
## 7          8 1174.8501036        TRUE
## 8          9 1275.9895168        TRUE
## 9         10  670.5540231        TRUE
## 10        11  418.3654420        TRUE
## 11        12  442.2932103        TRUE
## 12        13  327.5829387        TRUE
## 13        14  373.4409782        TRUE
## 14        15  274.9625944        TRUE
## 15        16  318.6367472        TRUE
## 16        17  289.7907577        TRUE
## 17        18  213.6310719        TRUE
## 18        19  249.0605999        TRUE
## 19        20  201.5569046        TRUE
## 20        21  140.0763689        TRUE
## 21        22  408.0147689        TRUE
## 22        23  241.8132830        TRUE
## 23        24  134.2620639        TRUE
## 24        25  154.4612090        TRUE
## 25        26  144.8127500        TRUE
## 26        27  266.5124592        TRUE
## 27        28  147.8748740        TRUE
## 28        29  280.1558945        TRUE
## 29        30   -0.9376062       FALSE
## 30        31  347.9838952        TRUE
## 31        32   58.2544166        TRUE
## 32        33  147.1727404        TRUE
## 33        34  262.1642875        TRUE
## 34        35  110.0333762        TRUE
## 35        36  324.3979718        TRUE
## 36        37   26.5629248        TRUE
## 37        38   30.8233731        TRUE
## 38        39  189.2378838        TRUE
## 39        40   97.1636558        TRUE
```

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.

```
set.seed(123)
k.max <- 12
data <- clustering_data
wss <- sapply(2:k.max,
              function(k){kmeans(data, k, nstart=50,iter.max = 12 )$tot.withinss})
wss
```

```
##  [1] 8443681.1 6014377.9 4009678.4 2171612.8 1519043.4 1102869.9  853160.1
##  [8]  647328.7  554632.2  502329.9  450061.1
```

```
plot(2:k.max, wss,
     type="b", pch = 19, frame = FALSE,
     xlab="Number of clusters K",
     ylab="Total within-clusters sum of squares")
```



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?

k = 6