Machine Learning

Felipe Rodriguez

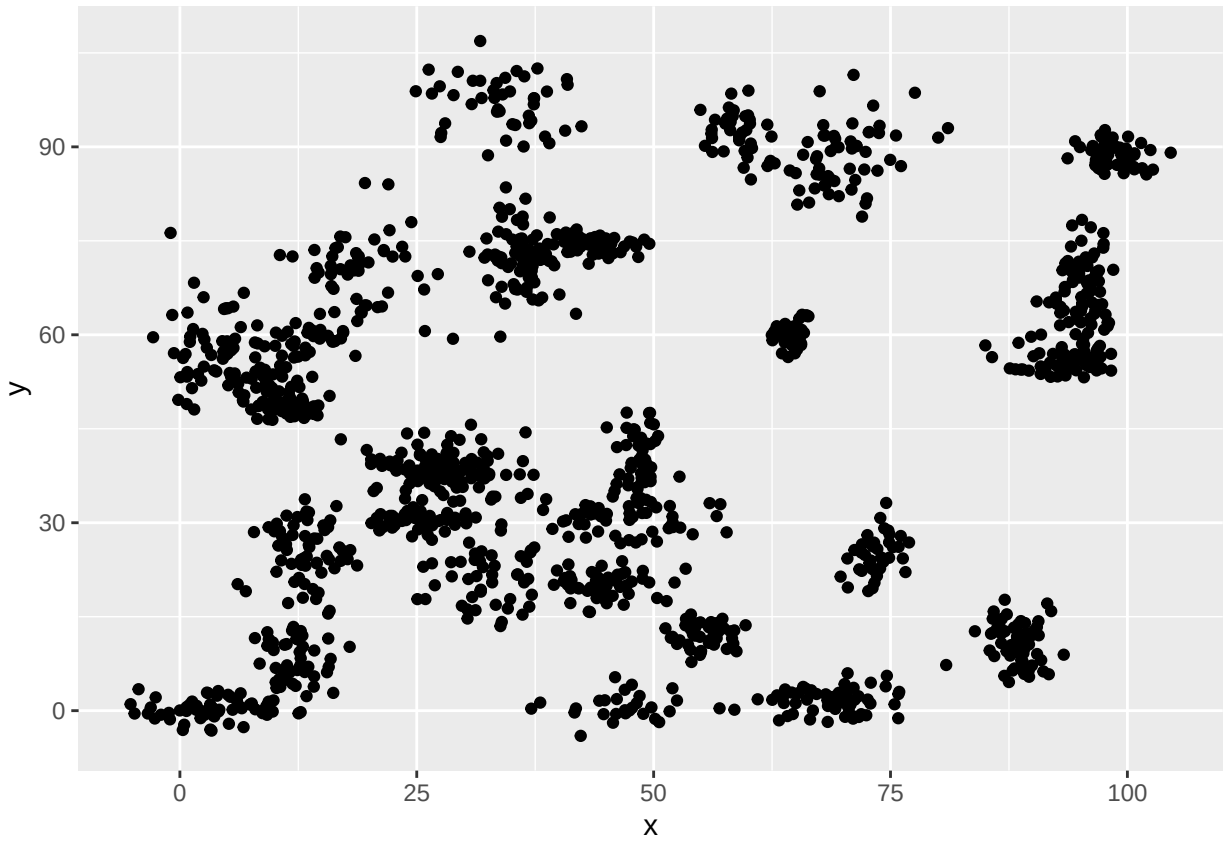
2023-03-03

In this problem, you will use the nearest neighbors algorithm to fit a model on two simplified datasets. The first dataset (found in binary-classifier-data.csv) contains three variables; label, x, and y. The label variable is either 0 or 1 and is the output we want to predict using the x and y variables (You worked with this dataset last week!). The second dataset (found in trinary-classifier-data.csv) is similar to the first dataset except that the label variable can be 0, 1, or 2.

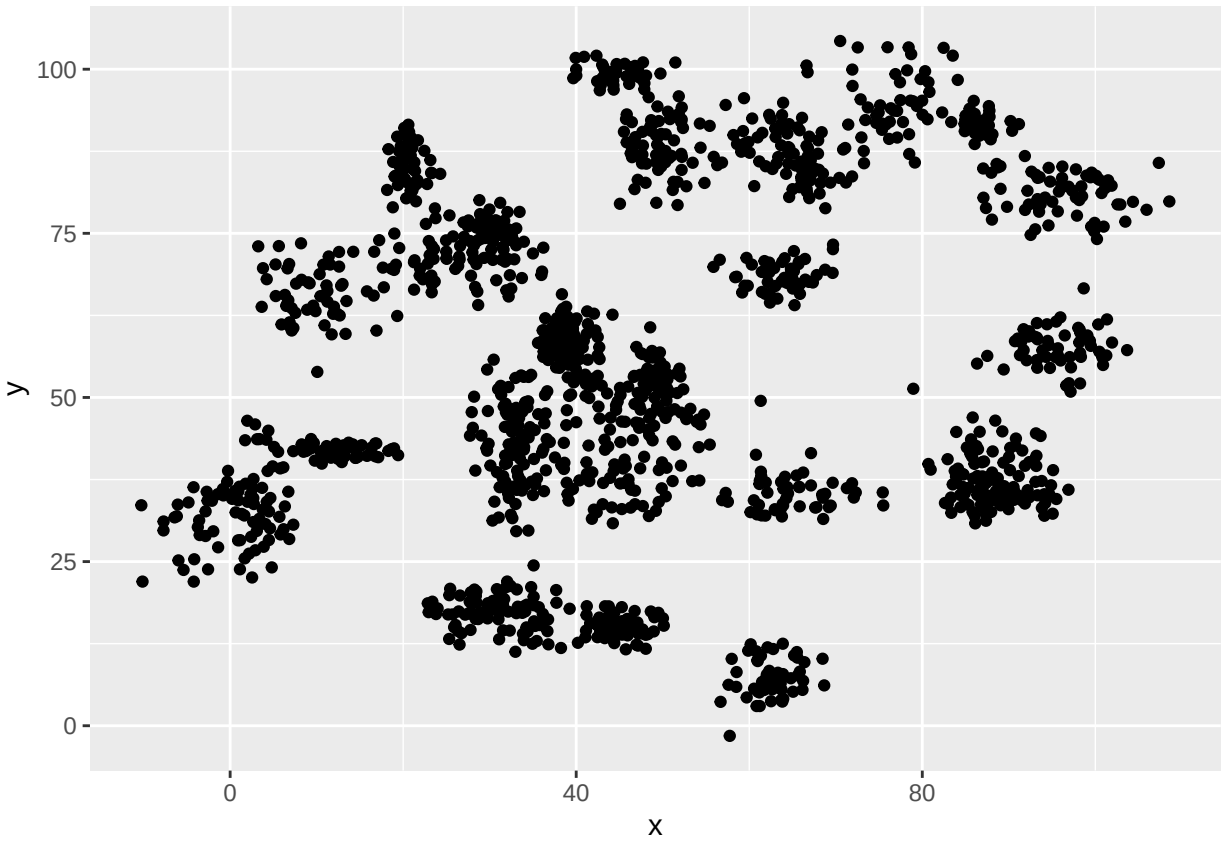
```
setwd("/Users/feliperodriguez/OneDrive - Bellevue University/Github/dsc520/data/")
binary <- read.csv("binary-classifier-data.csv")
trinary <- read.csv("trinary-classifier-data.csv")
```

Plot the data from each dataset using a scatter plot.

```
# Binary Data Scatter Plot
library(ggplot2)
ggplot(binary, aes(x = x, y = y)) + geom_point()
```




```
# Trinary Data Scatter Plot
ggplot(trinary, aes(x = x, y = y)) + geom_point()
```



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. As a refresher, the Euclidean distance between two points:

Fitting a model is when you use the input data to create a predictive model. There are various metrics you can use to determine how well your model fits the data. For this problem, you will focus on a single metric, accuracy. Accuracy is simply the percentage of how often the model predicts the correct result. If the model always predicts the correct result, it is 100% accurate. If the model always predicts the incorrect result, it is 0% accurate.

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.

Binary

```
#Normalization
normalize <- function(x)
```

```
{return ((x - min(x)) / (max(x) - min(x)))}
binary.n <- as.data.frame(lapply(binary[,2:3], normalize))

set.seed(123)
binary.d <- sample(1:nrow(binary.n),size=nrow(binary.n)*0.7,replace = FALSE) #random selection of 70% d

train.binary <- binary.n[binary.d,] # 70% training data
test.binary <- binary.n[-binary.d,] # remaining 30% test data


#Creating seperate dataframe for 'Label feature which is our target.
train.binary_labels <- binary[binary.d,1]
test.binary_labels <-binary[-binary.d,1]

library(class)
NROW(train.binary_labels)
```

```
## [1] 1048
```

```
knn.3 <- knn(train=train.binary, test=test.binary, cl=train.binary_labels, k=3)
knn.5 <- knn(train=train.binary, test=test.binary, cl=train.binary_labels, k=5)
knn.10 <- knn(train=train.binary, test=test.binary, cl=train.binary_labels, k=10)
knn.15 <- knn(train=train.binary, test=test.binary, cl=train.binary_labels, k=15)
knn.20 <- knn(train=train.binary, test=test.binary, cl=train.binary_labels, k=20)
knn.25 <- knn(train=train.binary, test=test.binary, cl=train.binary_labels, k=25)
```

```
ACC.3 <- 100 * sum(test.binary_labels == knn.3)/NROW(test.binary_labels)
ACC.5 <- 100 * sum(test.binary_labels == knn.5)/NROW(test.binary_labels)
ACC.10 <- 100 * sum(test.binary_labels == knn.10)/NROW(test.binary_labels)
ACC.15 <- 100 * sum(test.binary_labels == knn.15)/NROW(test.binary_labels)
ACC.20 <- 100 * sum(test.binary_labels == knn.20)/NROW(test.binary_labels)
ACC.25 <- 100 * sum(test.binary_labels == knn.25)/NROW(test.binary_labels)

ACC.3
```

```
## [1] 97.11111
```

```
ACC.5
```

```
## [1] 97.11111
```

```
ACC.10
```

```
## [1] 98
```

```
ACC.15
```

```
## [1] 97.33333
```

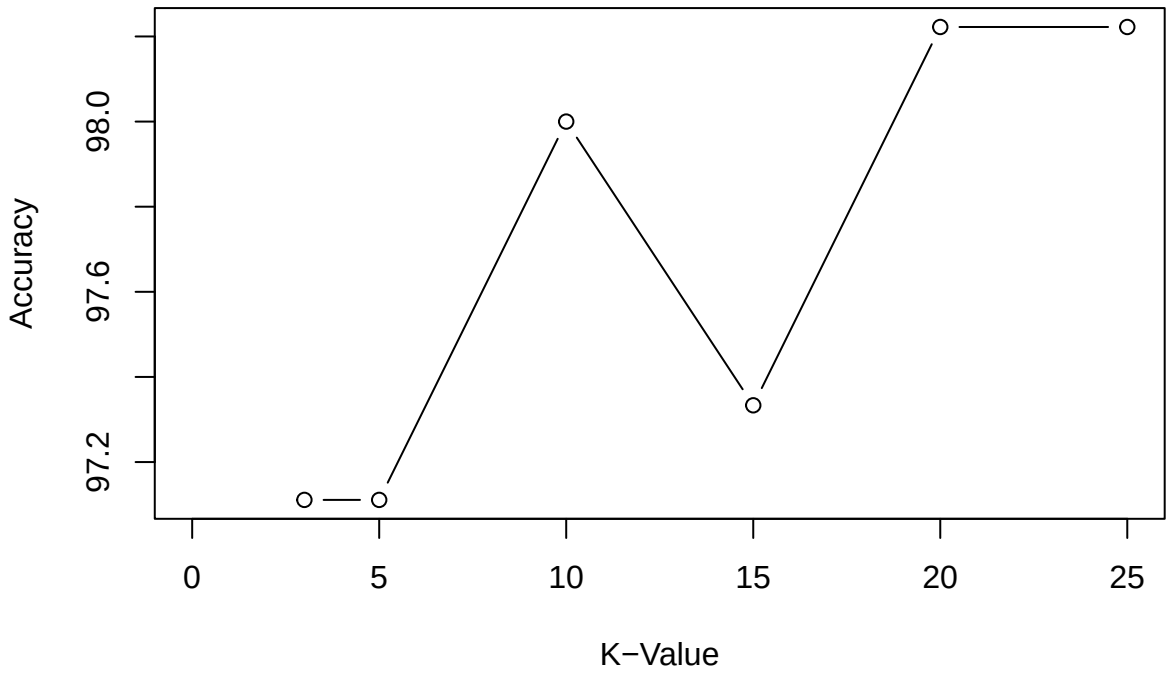
```
ACC.20
```

```
## [1] 98.22222
```

```
ACC.25
```

```
## [1] 98.22222
```

```
accurary = c(ACC.3,
ACC.5,
ACC.10,
ACC.15,
ACC.20,
ACC.25)
k_value =c(3, 5, 10, 15, 20, 25)
data <- data.frame(k_value, accurary)
plot(data, type="b", xlab="K-Value", ylab="Accuracy", xlim=c(0,25))
```



Trinary

```
#Normalization
normalize <- function(x)
{return ((x - min(x)) / (max(x) - min(x)))}
trinary.n <- as.data.frame(lapply(trinary[,2:3], normalize))

set.seed(123)
trinary.d <- sample(1:nrow(trinary.n),size=nrow(trinary.n)*0.7,replace = FALSE) #random selection of 70
```

```
train.trinary <- trinary.n[trinary.d,] # 70% training data
test.trinary <- trinary.n[-trinary.d,] # remaining 30% test data


#Creating seperate dataframe for 'Label feature which is our target.
train.trinary_labels <- trinary[trinary.d,1]
test.trinary_labels <-trinary[-trinary.d,1]

library(class)
NROW(train.trinary_labels)
```

```
## [1] 1097
```

```
knn.3 <- knn(train=train.trinary, test=test.trinary, cl=train.trinary_labels, k=3)
knn.5 <- knn(train=train.trinary, test=test.trinary, cl=train.trinary_labels, k=5)
knn.10 <- knn(train=train.trinary, test=test.trinary, cl=train.trinary_labels, k=10)
knn.15 <- knn(train=train.trinary, test=test.trinary, cl=train.trinary_labels, k=15)
knn.20 <- knn(train=train.trinary, test=test.trinary, cl=train.trinary_labels, k=20)
knn.25 <- knn(train=train.trinary, test=test.trinary, cl=train.trinary_labels, k=25)
```

```
trinary.ACC.3 <- 100 * sum(test.trinary_labels == knn.3)/NROW(test.trinary_labels)
trinary.ACC.5 <- 100 * sum(test.trinary_labels == knn.5)/NROW(test.trinary_labels)
trinary.ACC.10 <- 100 * sum(test.trinary_labels == knn.10)/NROW(test.trinary_labels)
trinary.ACC.15 <- 100 * sum(test.trinary_labels == knn.15)/NROW(test.trinary_labels)
trinary.ACC.20 <- 100 * sum(test.trinary_labels == knn.20)/NROW(test.trinary_labels)
trinary.ACC.25 <- 100 * sum(test.trinary_labels == knn.25)/NROW(test.trinary_labels)

trinary.ACC.3
```

```
## [1] 85.77495
```

```
trinary.ACC.5
```

```
## [1] 87.68577
```

```
trinary.ACC.10
```

```
## [1] 86.6242
```

```
trinary.ACC.15
```

```
## [1] 86.83652
```

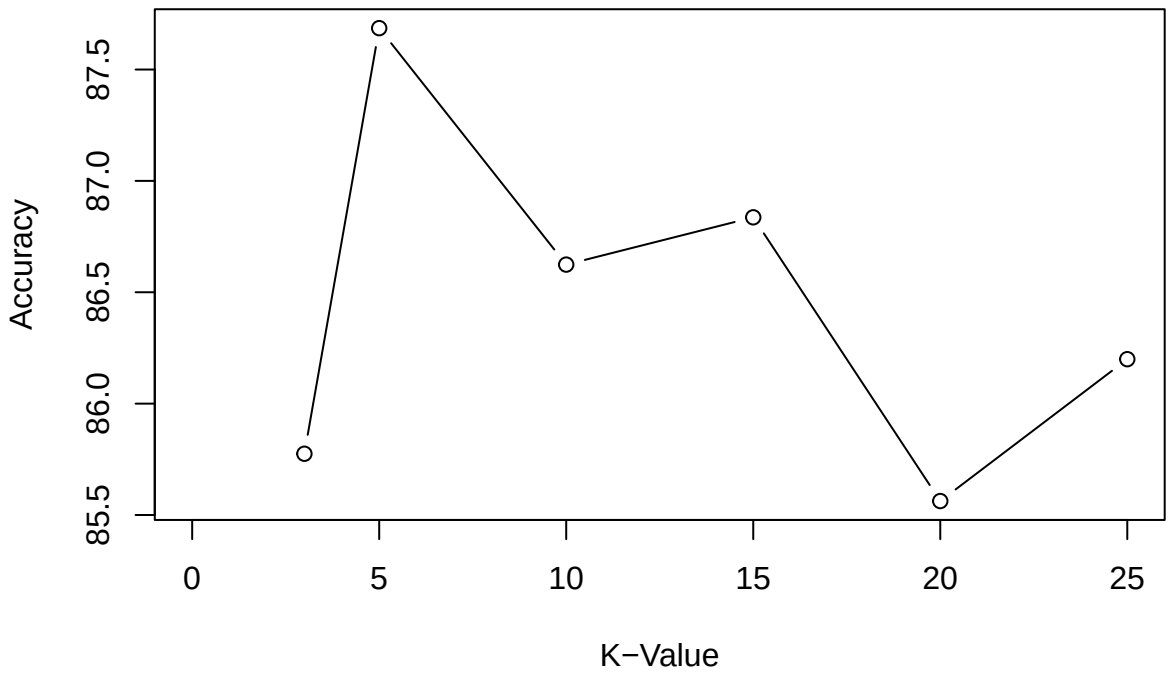
```
trinary.ACC.20
```

```
## [1] 85.56263
```

```
trinary.ACC.25
```

```
## [1] 86.19958
```

```
accurary = c(trinary.ACC.3,
trinary.ACC.5,
trinary.ACC.10,
trinary.ACC.15,
trinary.ACC.20,
trinary.ACC.25)
k_value =c(3, 5, 10, 15, 20, 25)
data <- data.frame(k_value, accurary)
plot(data, type="b", xlab="K-Value", ylab="Accuracy", xlim=c(0,25))
```



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

I do not believe a linear classifier would work well on the scatter plots of the two data sets. Although the data sets only have 2 and 3 labels, the clusters that are formed in each plot have too many clusters to be only categorized into two classes.

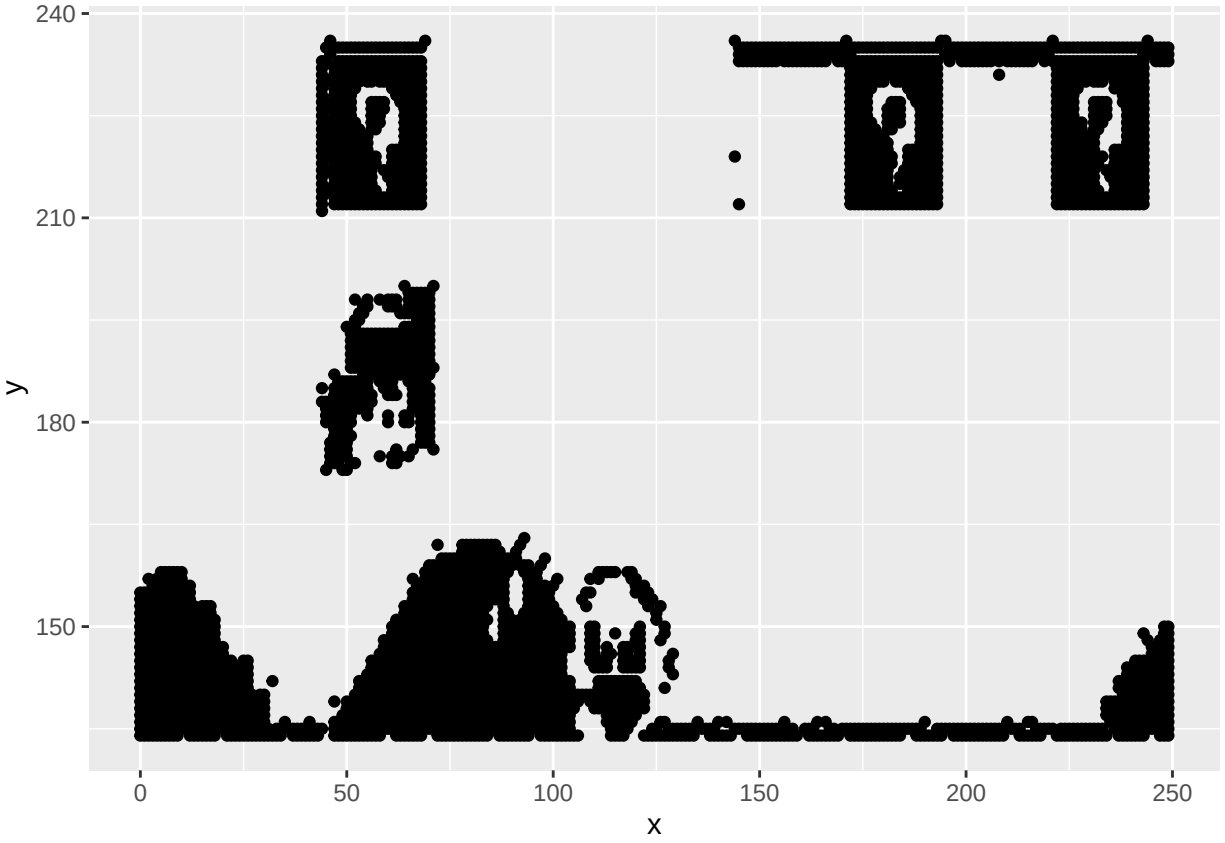
How does the accuracy of your logistic regression classifier from last week compare? Why is the accuracy different between these two methods?

The accuracy for KNN is higher than logistical regression last week. If the target label has no linear correlation with the features, the model will be less accurate. This data seems to be non-linear so KNN fits better against this data.

In this problem, you will use the k-means clustering algorithm to look for patterns in an unlabeled dataset. The dataset for this problem is found at data/clustering-data.csv.

Plot the dataset using a scatter plot.

```
setwd("/Users/feliperodriguez/OneDrive - Bellevue University/Github/dsc520/data/")
data_cluster <- read.csv("clustering-data.csv")
library(ggplot2)
ggplot(data_cluster, aes(x = x, y = y)) + geom_point()
```



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

```
data_cluster_k2 <- kmeans(x=data_cluster, centers = 2)
data_cluster_k2
```

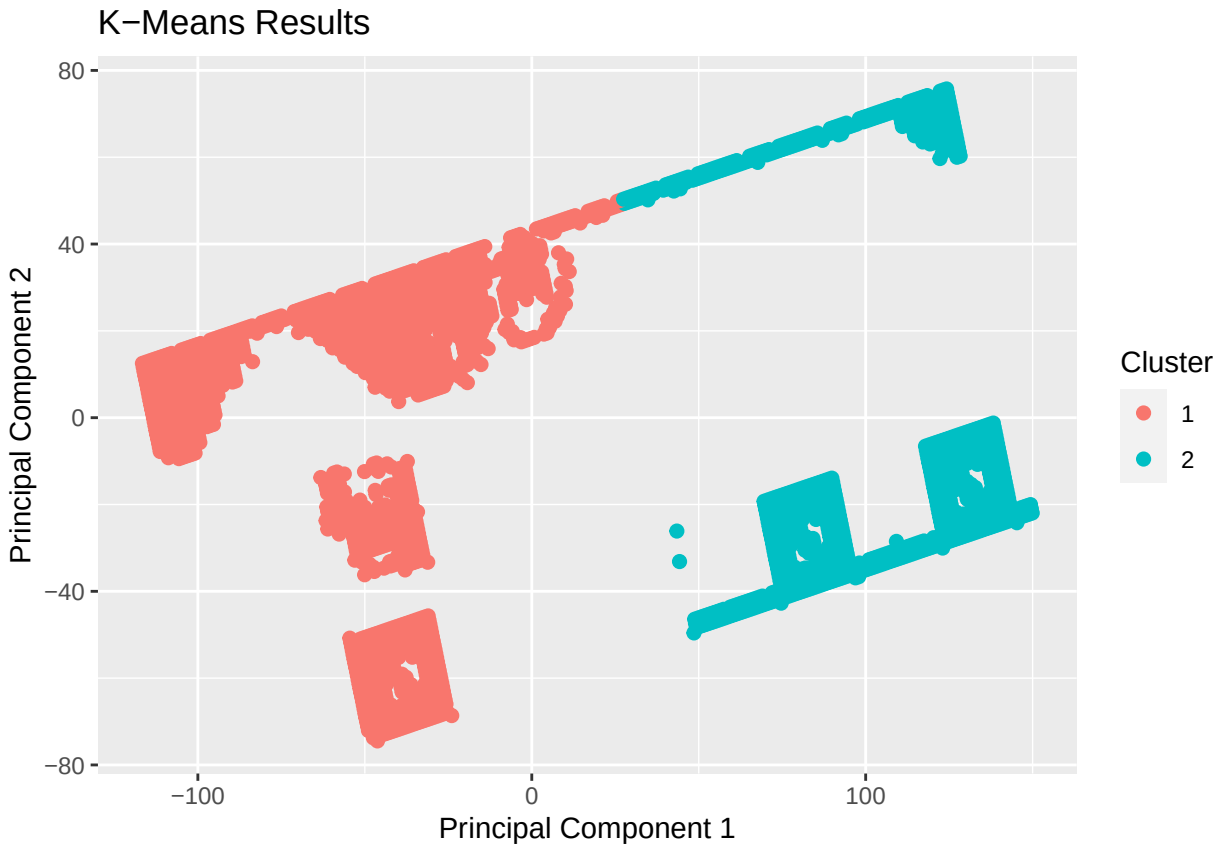
```
## K-means clustering with 2 clusters of sizes 2714, 1308
##
```

```
## Cluster means:
##          x        y
## 1  62.2815 162.4090
## 2 207.8005 203.2898
##
## Clustering vector:
##    [1] 1 1 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2
##   [38] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##   [75] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [112] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [149] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [186] 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2
##  [223] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [260] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [297] 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2
##  [334] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [371] 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2
##  [408] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
##  [445] 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [482] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
##  [519] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2
##  [556] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2
##  [593] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1
##  [630] 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1
##  [667] 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [704] 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [741] 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2
##  [778] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2
##  [815] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1
##  [852] 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1
##  [889] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [926] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [963] 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [1000] 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2
## [1037] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1
## [1074] 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [1111] 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2
## [1148] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1
## [1185] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [1222] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2
## [1259] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1
## [1296] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [1333] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1
## [1370] 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [1407] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1444] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1481] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1518] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1555] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1592] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1629] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1666] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1703] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1740] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
## [1777] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1814] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1851] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1888] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1925] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1962] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [1999] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2036] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2073] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2110] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2147] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2184] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2221] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2258] 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2295] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2
## [2332] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2369] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 1 1 1 1
## [2406] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2443] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2
## [2480] 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2517] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2554] 1 1 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2591] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2628] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2665] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2702] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2
## [2739] 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2776] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2813] 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2850] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2887] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2
## [2924] 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2961] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [2998] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2
## [3035] 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3072] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3109] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
## [3146] 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3183] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3220] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2
## [3257] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3294] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3331] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3368] 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3405] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3442] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3479] 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3516] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3553] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3590] 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1
## [3627] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3664] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3701] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3738] 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```
## [3775] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [3812] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3849] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3886] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3923] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [3960] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [3997] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##
## Within cluster sum of squares by cluster:
## [1] 5385717 3057964
##  (between_SS / total_SS =  70.5 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
library(useful)
plot(data_cluster_k2, data=data_cluster)
```



K=12

```
data_cluster_k12 <- kmeans(x=data_cluster, centers = 12)
data_cluster_k12
```

```
## K-means clustering with 12 clusters of sizes 441, 535, 193, 216, 148, 228, 467, 105, 602, 196, 518, 3
##
```

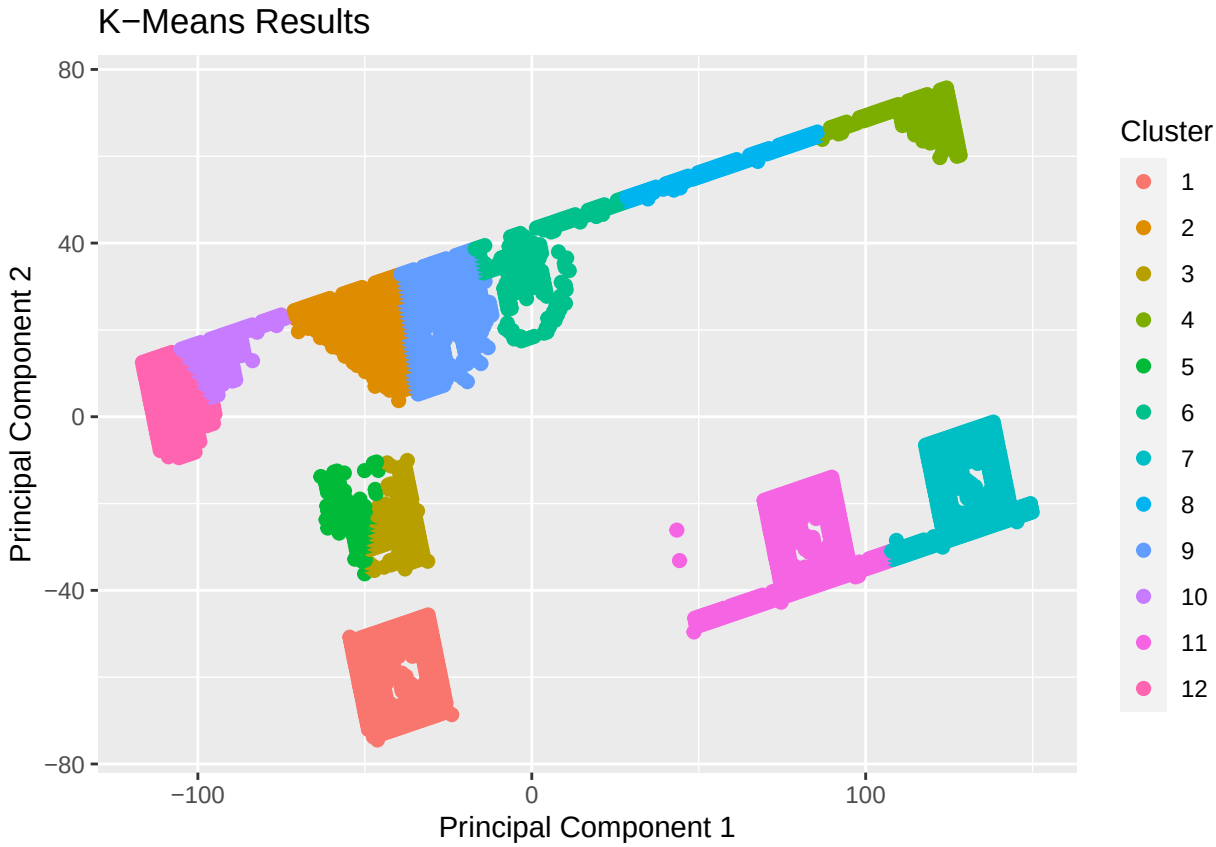
```
## Cluster means:
##             x        y
## 1   56.344671 223.0181
## 2   66.629907 143.7776
## 3   64.580311 189.7358
## 4  239.287037 139.0648
## 5   51.324324 183.3919
## 6  119.298246 141.4693
## 7  231.089936 224.3683
## 8  179.933333 134.5524
## 9   89.928571 146.5980
## 10  23.489796 138.3980
## 11 179.550193 225.2664
## 12   7.597855 145.7936
##
## Clustering vector:
##    [1]  1  1 11 11 11 11  7  7  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
##   [25]  1  1  1  1  1  1  1 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
##   [49] 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
##   [73] 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11  7  7  7  7  7  7
##   [97]  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7
##  [121]  7  7  7  7  7  7  7  7  7  7  7  7  1 11 11 11 11 11 11 11 11 11 11 11
##  [145] 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
##  [169] 11 11  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7
##  [193]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 11
##  [217] 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
##  [241] 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
##  [265] 11 11 11 11  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7
##  [289]  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  1  1  1  1  1  1  1
##  [313]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 11 11 11 11 11 11 11 11
##  [337] 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11  7  7  7  7  7  7  7  7  7
##  [361]  7  7  7  7  7  7  7  7  7  7  7  7  1  1  1  1  1  1  1  1  1  1  1  1
##  [385]  1  1  1  1  1  1  1  1  1 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
##  [409] 11 11 11 11 11  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7
##  [433]  7  7  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
##  [457] 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11  7  7
##  [481]  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  1  1  1  1  1  1  1  1  1
##  [505]  1  1  1  1  1  1 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11  7  7
##  [529]  7  7  7  7  7  7  7  7  7  7  1  1  1  1  1  1  1  1  1  1  1  1  1 11
##  [553] 11 11 11 11 11 11 11 11 11 11 11 11  7  7  7  7  7  7  7  7  7  7  7  1
##  [577]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 11 11 11 11 11 11 11 11
##  [601] 11 11 11 11 11  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  1  1  1
##  [625]  1  1  1  1  1  1  1  1  1  1  1  1 11 11 11 11 11 11 11 11 11 11 11 11
##  [649] 11 11  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  1  1  1  1  1  1  1
##  [673]  1  1  1  1  1  1 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11  7  7
##  [697]  7  7  7  7  7  7  7  7  7  7  1  1  1  1  1  1  1  1  1  1  1  1  1  1
##  [721]  1  1 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11  7  7  7  7  7
##  [745]  7  7  7  7  7  7  7  7  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
##  [769] 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11  7  7  7  7  7  7  7  7  7
##  [793]  7  7  7  7  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 11 11 11
##  [817] 11 11 11 11 11 11 11 11 11 11  7  7  7  7  7  7  7  7  7  7  7  7  7  7
##  [841]  7  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 11 11 11 11 11 11
##  [865] 11 11 11 11 11 11 11 11  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  1
##  [889]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 11 11 11 11 11 11
```

```
##  [913] 11 11 11 11 11 11 11 11 11 11 11 11 11  7  7  7  7  7  7  7  7  7  7  7
##  [937]  7  7  7  7  7  7  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
##  [961]  1  1  1 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11  7
##  [985]  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  1  1  1  1  1
## [1009]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 11 11 11 11 11 11 11 11 11
## [1033] 11 11 11 11 11 11 11 11 11 11 11  7  7  7  7  7  7  7  7  7  7  7  7  7
## [1057]  7  7  7  7  7  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
## [1081]  1  1  1 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11  7  7
## [1105]  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  1  1  1  1  1  1
## [1129]  1  1  1  1  1  1  1  1  1  1  1  1  1  1 11 11 11 11 11 11 11 11 11 11
## [1153] 11 11 11 11 11 11 11 11 11 11  7  7  7  7  7  7  7  7  7  7  7  7  7  7
## [1177]  7  7  7  7  7  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
## [1201] 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11  7  7  7  7
## [1225]  7  7  7  7  7  7  7  7  7  7  7  7  7  1  1  1  1  1  1  1  1  1  1  1
## [1249]  1  1  1  1  1  1  1  1 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
## [1273] 11  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  1  1  1  1  1  1
## [1297]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 11 11 11 11 11 11 11
## [1321] 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11  7  7  7  7  7  7  7  7
## [1345]  7  7  7  7  7  7  7  7  7  7  7  7  7  1  1  1  1  1  1  1  1  1  1  1
## [1369]  1  1  1  1  1  1  1  1  1  1  1  1 11 11 11 11 11 11 11 11 11 11 11 11
## [1393] 11 11 11 11 11 11 11 11 11 11 11 11  7  7  7  7  7  7  7  7  7  7  7  7
## [1417]  7  7  7  7  7  7  7  7  7  1  3  3  3  3  3  3  3  3  5  3  3  3  3  3
## [1441]  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  5  3  3  3  3  3
## [1465]  3  3  3  5  5  3  3  3  3  5  5  3  3  3  3  3  3  3  5  5  5  5  3  3
## [1489]  3  3  3  3  3  3  3  3  3  3  3  3  3  5  5  5  5  5  3  3  3  3  3  3
## [1513]  3  3  3  3  3  3  3  5  5  5  5  5  3  3  3  3  3  3  3  3  3  3  3  3
## [1537]  3  3  3  5  5  5  5  5  5  3  3  3  3  3  3  3  3  3  3  3  3  3  3  5
## [1561]  5  5  5  5  5  3  3  3  3  3  3  3  3  3  3  3  3  3  3  5  5  5  5  5
## [1585]  3  3  3  3  3  3  3  3  3  3  3  3  5  5  5  5  5  3  3  3  3  3  3  3
## [1609]  3  3  3  3  3  5  5  5  5  5  5  5  5  5  5  3  3  3  3  3  5  5  5  5
## [1633]  5  5  5  5  5  3  3  3  3  3  3  3  3  5  5  5  5  5  5  5  5  5  3  3
## [1657]  3  3  3  3  5  5  5  5  5  5  5  5  5  5  5  5  5  5  3  3  3  3  5  5
## [1681]  5  5  5  5  5  5  3  3  3  3  5  5  5  5  5  5  5  5  5  3  3  3  3  3
## [1705]  5  5  5  5  5  5  5  5  3  3  3  3  3  5  5  5  3  3  3  5  5  5  5  5
## [1729]  3  3  3  5  5  5  5  5  3  3  3  5  5  5  5  5  5  3  3  5  5  5  5  5
## [1753]  5  5  5  5  3  5  5  5  5  5  5  5  5  5  5  9  2  9  9  9  9  9  9  9
## [1777]  9  9  9  9  9  9  9  9  9  9  9  9  9  9  2  2  2  2  9  9  9  9  9  9
## [1801]  9  9  9  9  9  9  9  9  9  9  2  2  2  2  2  2  2  9  9  9  9  9  9  9
## [1825]  9  9  9  9  9  9  9  9  9  9  9 12 12 12 12 12 12  2  2  2  2  2  2  2
## [1849]  2  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  6  6  6  6  6  6
## [1873]  6 12 12 12 12 12 12 12 12  2  2  2  2  2  2  2  2  2  2  9  9  9  9  9  9
## [1897]  9  9  9  9  9  9  9  9  9  9  6  6  6  6 12 12 12 12 12 12 12 12 12 12
## [1921]  2  2  2  2  2  2  2  2  2  2  9  9  9  9  9  9  9  9  9  9  9  9  9  9
## [1945]  9  9  9  9  6  6 12 12 12 12 12 12 12 12 12 12 12 12 12 12  2  2  2  2  2
## [1969]  2  2  2  2  2  2  2  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9
## [1993]  6  6  6  6  6 12 12 12 12 12 12 12 12 12 12 12 12 12  2  2  2  2  2  2
## [2017]  2  2  2  2  2  2  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  6
## [2041]  6  6 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12  2  2  2  2  2
## [2065]  2  2  2  2  2  2  2  2  2  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9
## [2089]  6  6  6 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12  2  2  2
## [2113]  2  2  2  2  2  2  2  2  2  2  2  9  9  9  9  9  9  9  9  9  9  9  9  9
## [2137]  9  9  9  9  6 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
## [2161]  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  9  9  9  9  9  9  9  9
## [2185]  9  9  9  9  9  9  9  9  9  9  9  9  9  6 12 12 12 12 12 12 12 12 12 12
```

```
## [2209] 12 12 12 12 12 12 12 12 12  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
## [2233]  2  2  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9
## [2257]  9  6  6  6  6  4  4 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
## [2281] 12 12  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  9  9  9  9  9
## [2305]  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  6  6  6  6  6  6
## [2329]  4  4  4 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12  2  2
## [2353]  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  9  9  9  9  9  9  9
## [2377]  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  6  6  6  6  6  4  4
## [2401]  4  4 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 10 10 10  2
## [2425]  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  9  9  9  9  9
## [2449]  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  6  6  6
## [2473]  6  6  6  6  4  4  4  4  4 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
## [2497] 12 12 12 10 10 10  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
## [2521]  2  2  2  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9
## [2545]  9  9  9  6  6  6  6  6  6  6  6  4  4  4  4  4 12 12 12 12 12 12 12 12
## [2569] 12 12 12 12 12 12 12 12 12 10 10 10 10 10 10 10  2  2  2  2  2  2  2  2
## [2593]  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  9  9  9  9  9  9  9  9  9
## [2617]  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  6  6  6  6  6  6  6  6  6
## [2641]  6  6  4  4  4  4  4  4  4  4  4 12 12 12 12 12 12 12 12 12 12 12 12 12
## [2665] 12 12 12 12 10 10 10 10 10 10 10 10 10  2  2  2  2  2  2  2  2  2  2  2
## [2689]  2  2  2  2  2  2  2  2  2  2  2  2  9  9  9  9  9  9  9  9  9  9  9  9
## [2713]  9  9  9  9  9  9  9  9  9  9  9  9  6  6  6  6  6  6  6  6  6  6  4  4
## [2737]  4  4  4  4  4  4  4  4  4 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
## [2761] 12 10 10 10 10 10 10 10 10 10 10 10  2  2  2  2  2  2  2  2  2  2  2  2  2
## [2785]  2  2  2  2  2  2  2  2  2  2  2  9  9  9  9  9  9  9  9  9  9  9  9  9
## [2809]  9  9  9  9  9  9  9  9  9  9  9  6  4  4  4  4  4  4  4  4  4  4  4 12
## [2833] 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 10 10 10 10 10 10 10 10 10
## [2857] 10 10 10  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
## [2881]  2  2  2  2  2  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9
## [2905]  9  9  9  9  9  9  9  6  6  6  6  6  6  6  6  6  6  4  4  4  4  4  4  4
## [2929]  4  4  4  4  4  4 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 10 10
## [2953] 10 10 10 10 10 10 10 10 10  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
## [2977]  2  2  2  2  2  2  2  2  2  2  2  9  9  9  9  9  9  9  9  9  9  9  9  9
## [3001]  9  9  9  9  9  9  9  9  9  9  9  9  9  6  6  6  6  6  6  6  6  6  6  6
## [3025]  6  4  4  4  4  4  4  4  4  4  4  4  4  4 12 12 12 12 12 12 12 12 12 12
## [3049] 12 12 12 12 12 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10  2  2  2  2
## [3073]  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  9
## [3097]  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9
## [3121]  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  4  4  4  4  4
## [3145]  4  4  4  4  4  4  4 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 10 10
## [3169] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10  2  2  2  2  2  2  2  2  2  2
## [3193]  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  9  9  9
## [3217]  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  6  6  6
## [3241]  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  4  4  4  4  4  4  4  4  4
## [3265]  4  4  4  4  4  4 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 10 10 10 10
## [3289] 10 10 10 10 10 10 10 10 10 10 10 10 10 10  2  2  2  2  2  2  2  2  2  2  2
## [3313]  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  9  9  9  9  9
## [3337]  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  6  6  6  6  6
## [3361]  6  6  6  6  6  6  6  6  6  6  4  4  4  4  4  4  4  4  4  4  4  4  4  4
## [3385]  4  4 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 10 10 10 10 10 10 10 10
## [3409] 10 10 10 10 10 10 10 10 10 10  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
## [3433]  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  9  9  9  9  9  9  9  9
## [3457]  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  6  6  6  6  6  6  6  6
## [3481]  6  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4 12 12 12 12 12 12 12
```

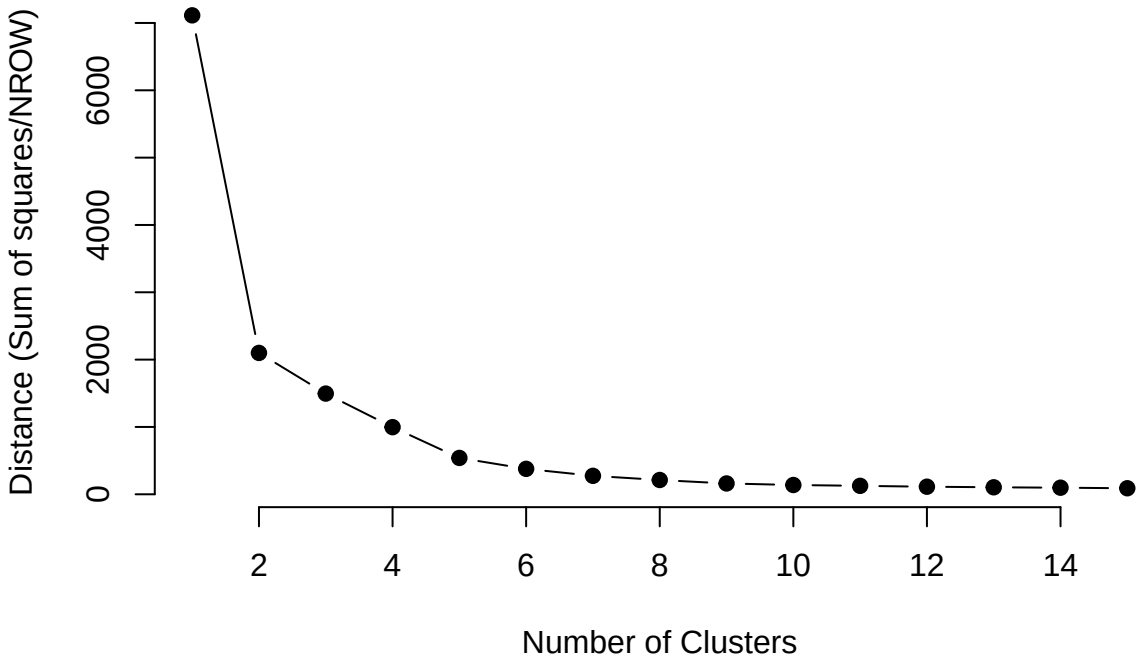
```
## [3505] 12 12 12 12 12 12 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
## [3529] 10 10  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
## [3553]  2  2  2  2  2  2  2  2  2  2  2  9  9  9  9  9  9  9  9  9  9  9  9  9
## [3577]  9  9  9  9  9  9  9  9  9  9  9  6  6  6  6  6  6  6  6  6  6  6  6  6
## [3601]  8  8  8  8  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4 12 12
## [3625] 12 12 12 12 12 12 12 12 12 12 12 10 10 10 10 10 10 10 10 10 10 10 10 10
## [3649] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10  2  2  2  2  2  2  2  2  2
## [3673]  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
## [3697]  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9
## [3721]  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6
## [3745]  6  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8
## [3769]  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8
## [3793]  8  8  8  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
## [3817]  4  4  4  4  4  4  4  4  4  4  4  4 12 12 12 12 12 12 12 12 12 12 10 10
## [3841] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
## [3865] 10 10  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
## [3889]  2  2  2  2  2  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9
## [3913]  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6
## [3937]  6  6  6  6  6  6  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8
## [3961]  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8
## [3985]  8  8  8  8  8  8  8  8  8  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
## [4009]  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
##
## Within cluster sum of squares by cluster:
##  [1]  47345.465  58919.252  10198.528  22577.296   7485.703  33696.504
##  [7]  65846.874  33068.495  70966.646  11925.939 112959.431  24216.783
##  (between_SS / total_SS =  98.3 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
library(useful)
plot(data_cluster_k12, data=data_cluster)
```



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)
library(purrr)
k.values <- 1:15
wws <- function(k){
  kmeans(data_cluster, k, nstart = 10)$tot.withinss/NROW(data_cluster)
}
wws.values <- map_dbl(k.values, wws)
plot(k.values, wws.values, type="b", pch=19, frame=FALSE, xlab="Number of Clusters", ylab="Distance (Su
```

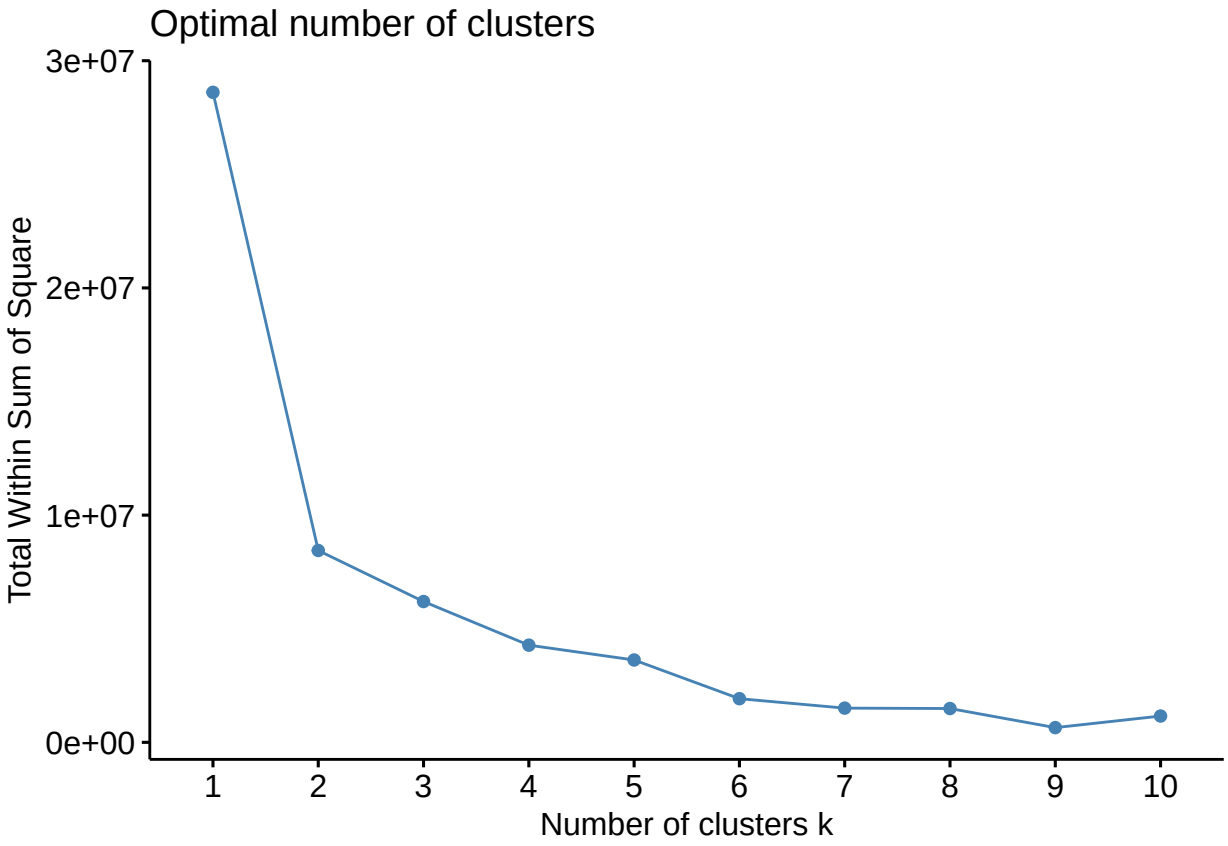


One way of determining the "right" number of clusters is to look at the graph of k versus average distance and finding the "elbow point". Looking at the graph you generated in the previous example, what is the elbow point for this dataset?

```
set.seed(123)
library(factoextra)
```

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

```
fviz_nbclust(data_cluster, kmeans, method = "wss")
```



The elbow point for this data set is k = 4.