Exercise 11.2

Jahedur Rahman

3/4/2022

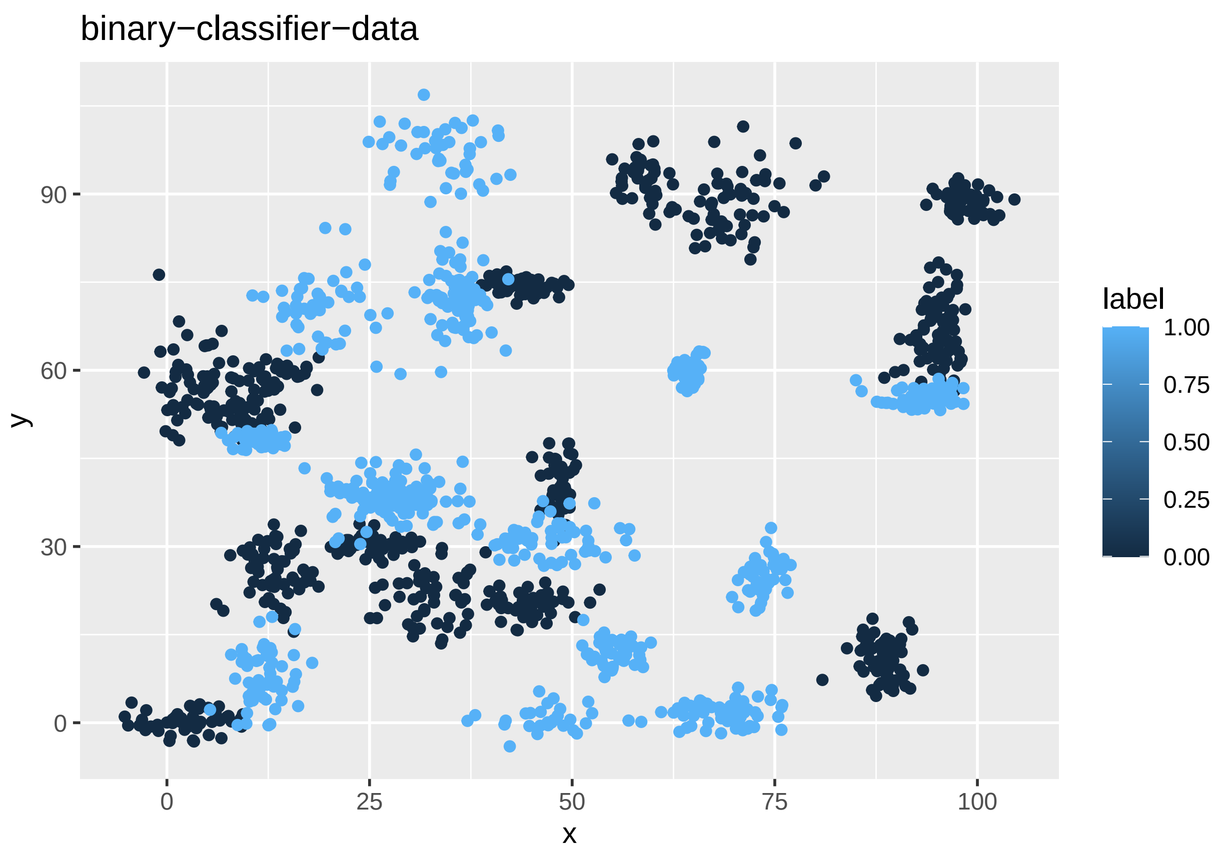
1ei

Plot the data from each dataset using a scatter plot.

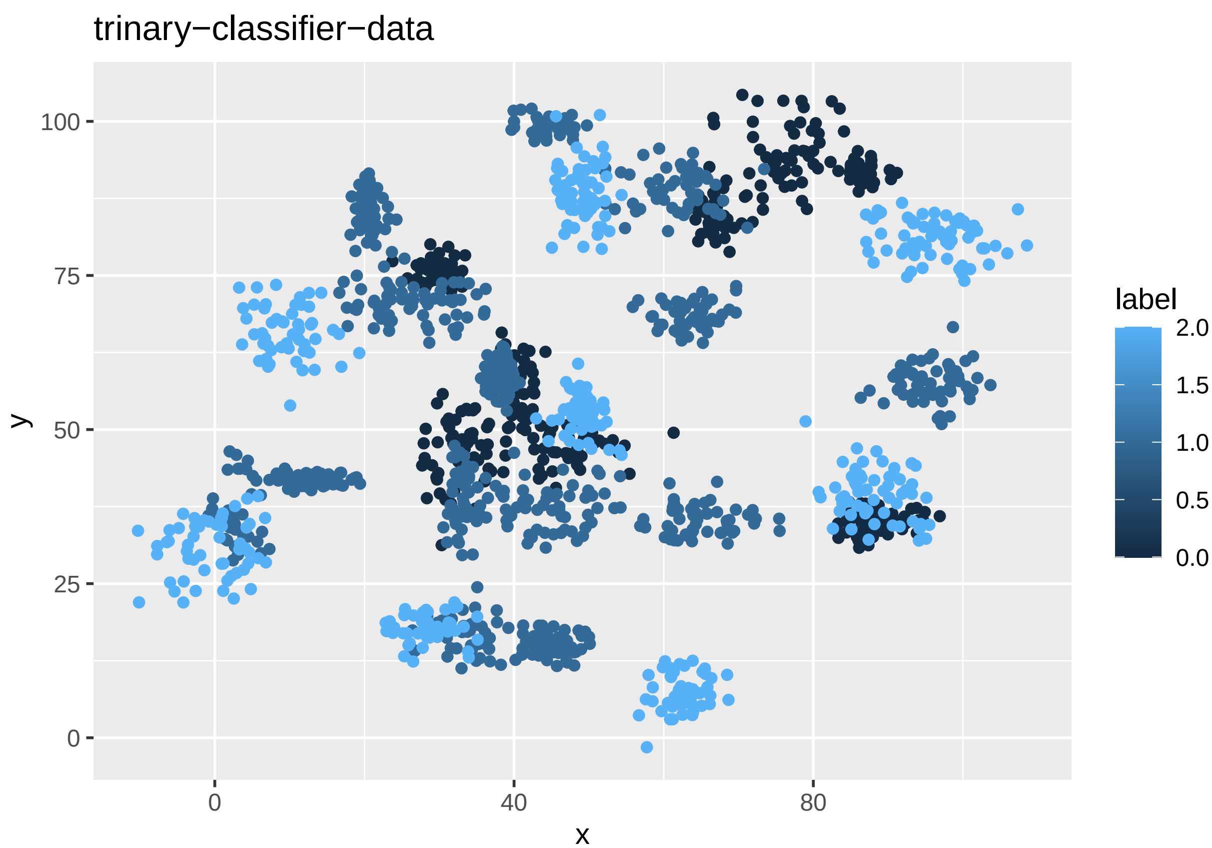
```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(stats)
library(ggplot2)
library(caTools)
# Load binary-classifier-data.csv and trinary-classifier-data.csv
binary_classifier_df=read.csv('binary-classifier-data.csv')
trinary_classifier_df=read.csv('trinary-classifier-data.csv')

ggplot(binary_classifier_df, aes(x=x, y=y, col=label)) + geom_point() + ggtitle("binary-classifier-data
```




```
ggplot(trinary_classifier_df, aes(x=x, y=y, col=label)) + geom_point() + ggtitle("trinary-classifier-dat
```



1eii

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:

```
euclidean_binary <- nls(label ~ sqrt((x1 - x)^2 + (y1 - y)^2), data=binary_classifier_df, start=list(x1=
summary(euclidean_binary)
```

```
##
## Formula: label ~ sqrt((x1 - x)^2 + (y1 - y)^2)
##
## Parameters:
##    Estimate Std. Error t value Pr(>|t|)
## x1   45.130      1.503   30.03   <2e-16 ***
## y1   45.068      1.478   30.49   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 40.55 on 1496 degrees of freedom
##
## Number of iterations to convergence: 11
## Achieved convergence tolerance: 1.347e-06
```

```
euclidean_trinary <- nls(label ~ sqrt((x1 - x)^2 + (y1 - y)^2), data=trinary_classifier_df, start=list(
summary(euclidean_trinary)
```

```
##
## Formula: label ~ sqrt((x1 - x)^2 + (y1 - y)^2)
##
## Parameters:
##    Estimate Std. Error t value Pr(>|t|)
## x1   48.949      1.295   37.80   <2e-16 ***
## y1   55.383      1.299   42.65   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 36.19 on 1566 degrees of freedom
##
## Number of iterations to convergence: 9
## Achieved convergence tolerance: 9.636e-06
```

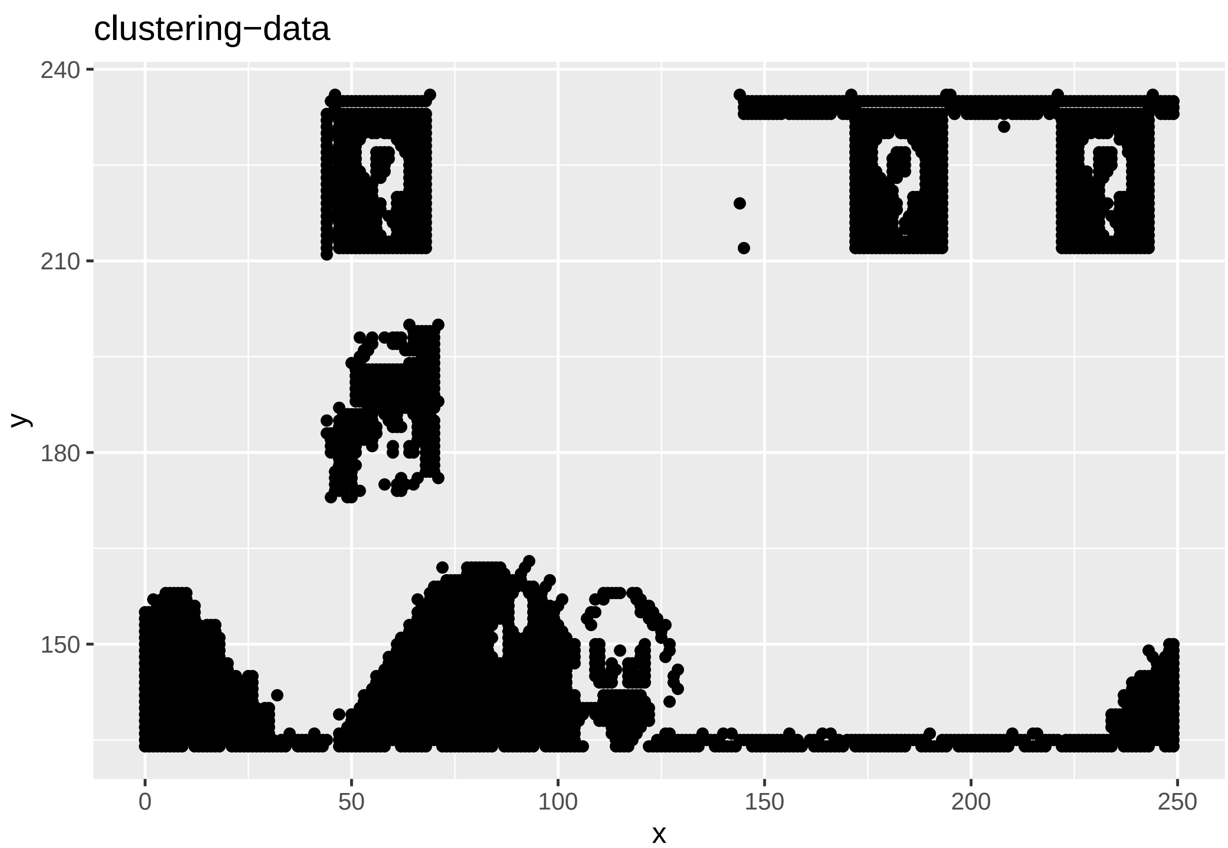
1eiii

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.

2di

Plot the dataset using a scatter plot.

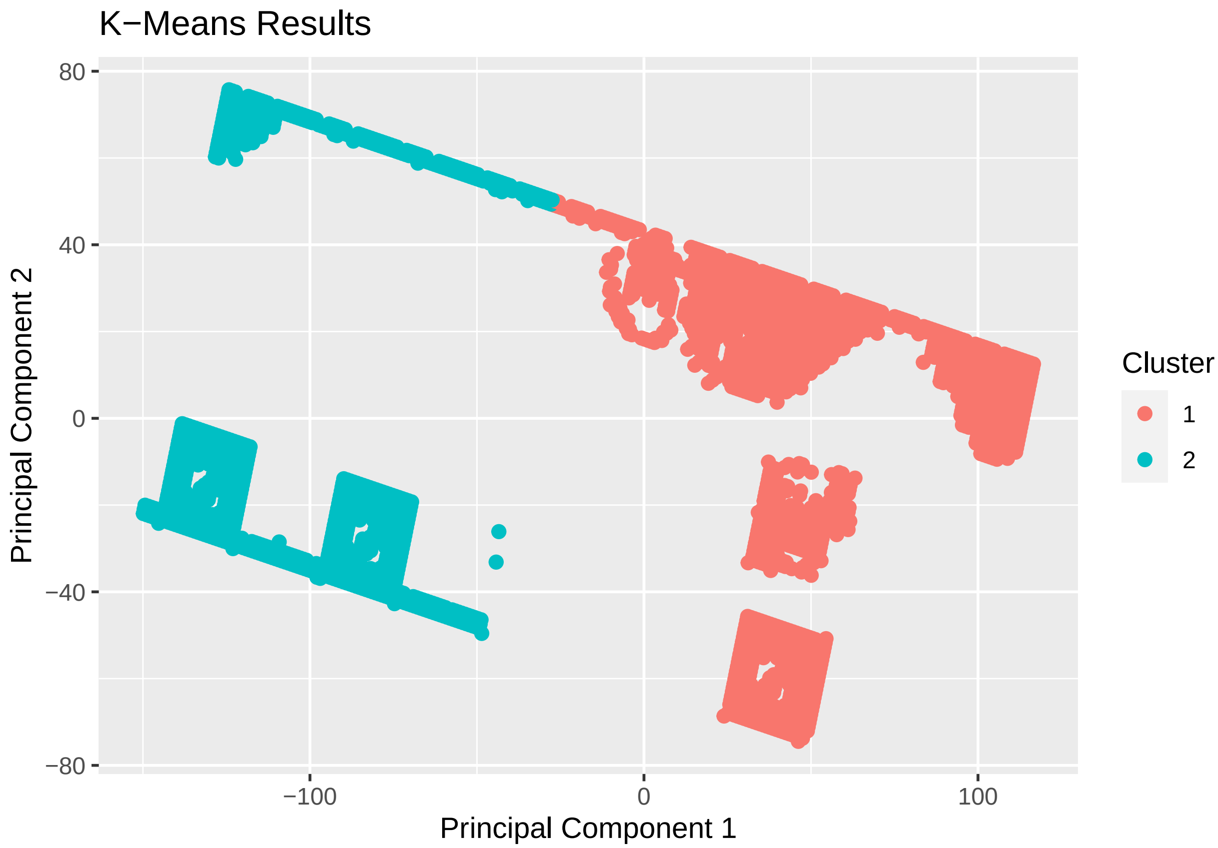
```
# Load clustering-data.csv
clustering_df=read.csv('clustering-data.csv')

ggplot(clustering_df, aes(x=x, y=y)) + geom_point() + ggtitle("clustering-data")
```



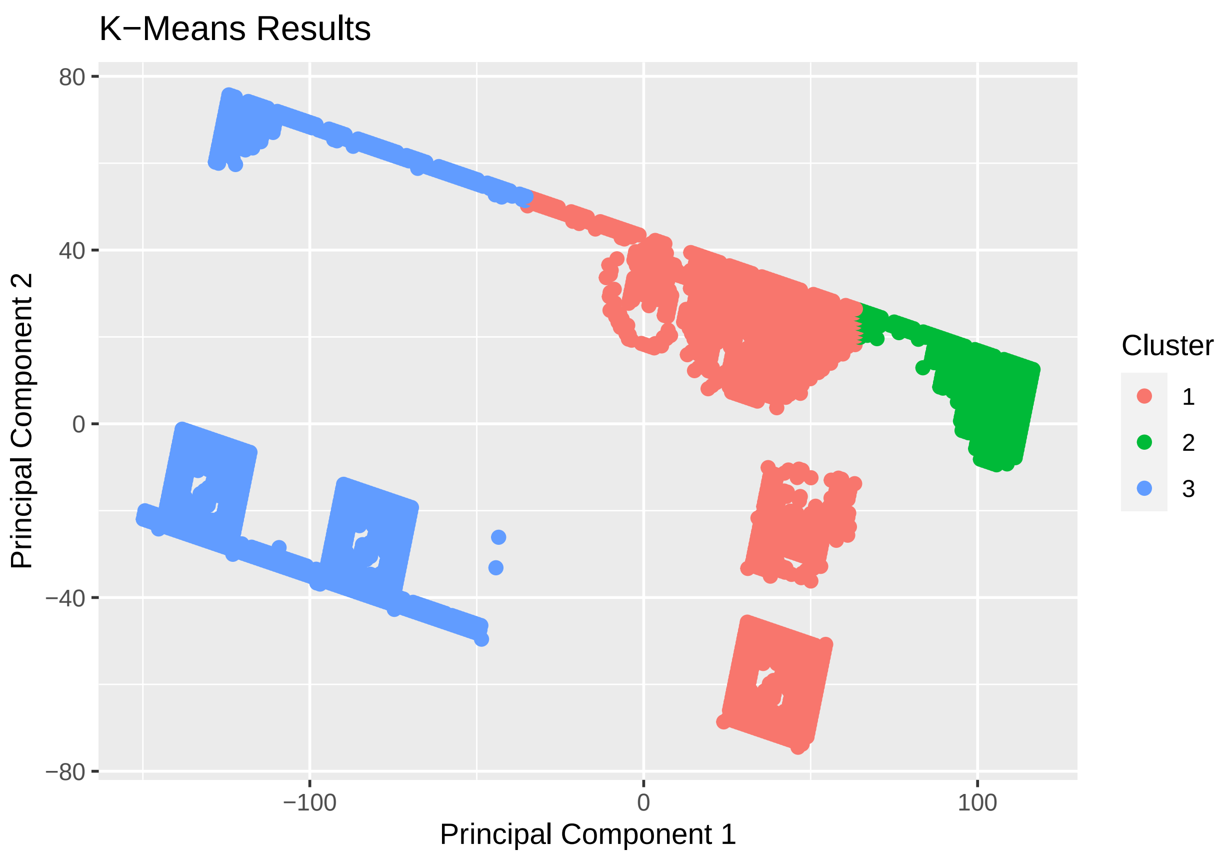
2dii

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.

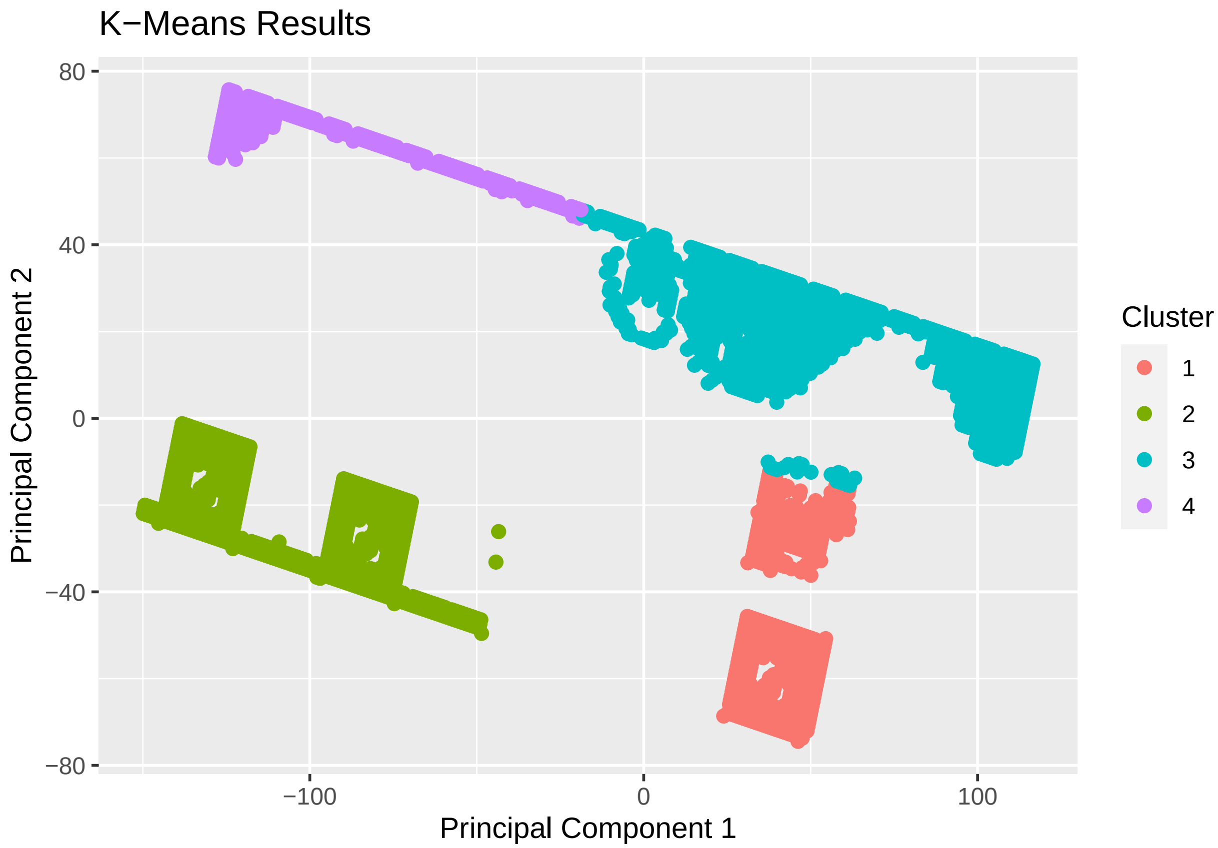
```
library(useful)
set.seed(278613)
clusteringK2 <- kmeans(x=clustering_df, centers=2)
plot(clusteringK2, data=clustering_df)
```



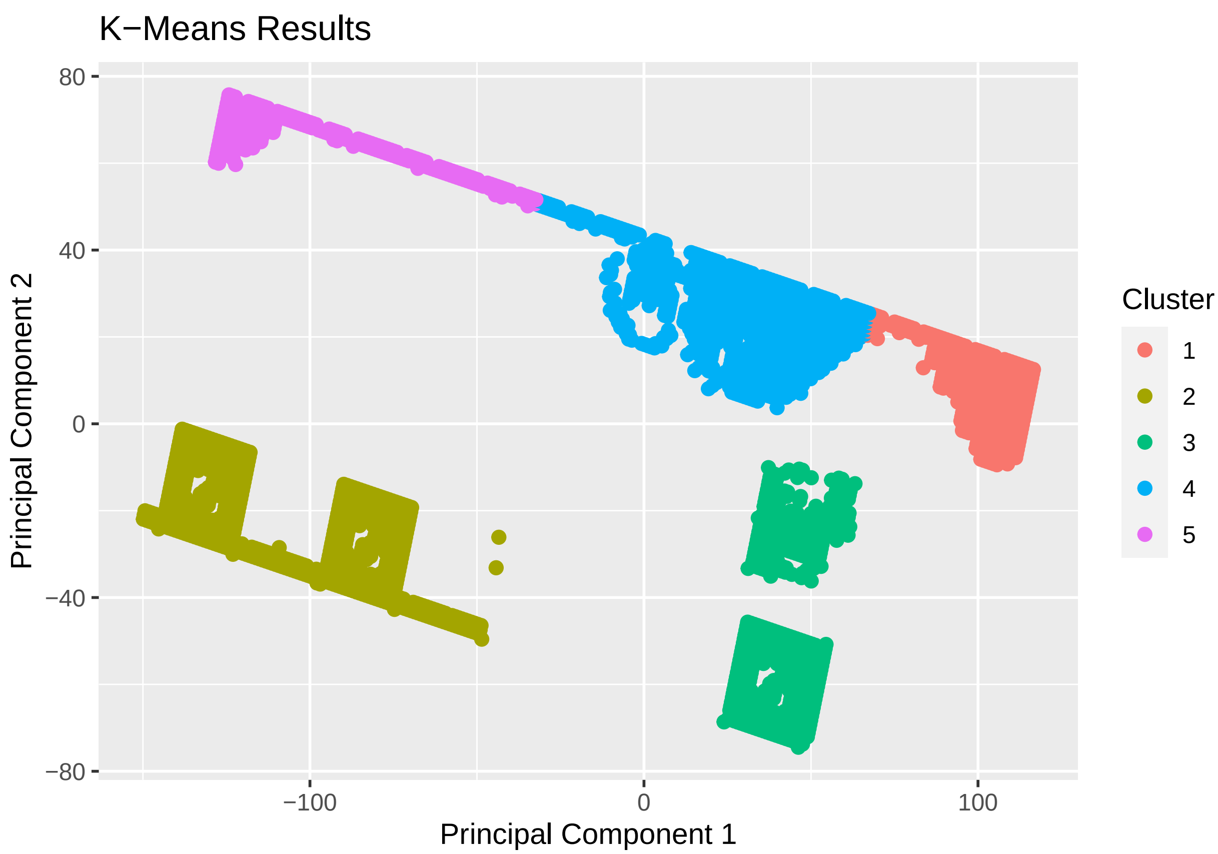

```
clusteringK3 <- kmeans(x=clustering_df, centers=3)
plot(clusteringK3, data=clustering_df)
```



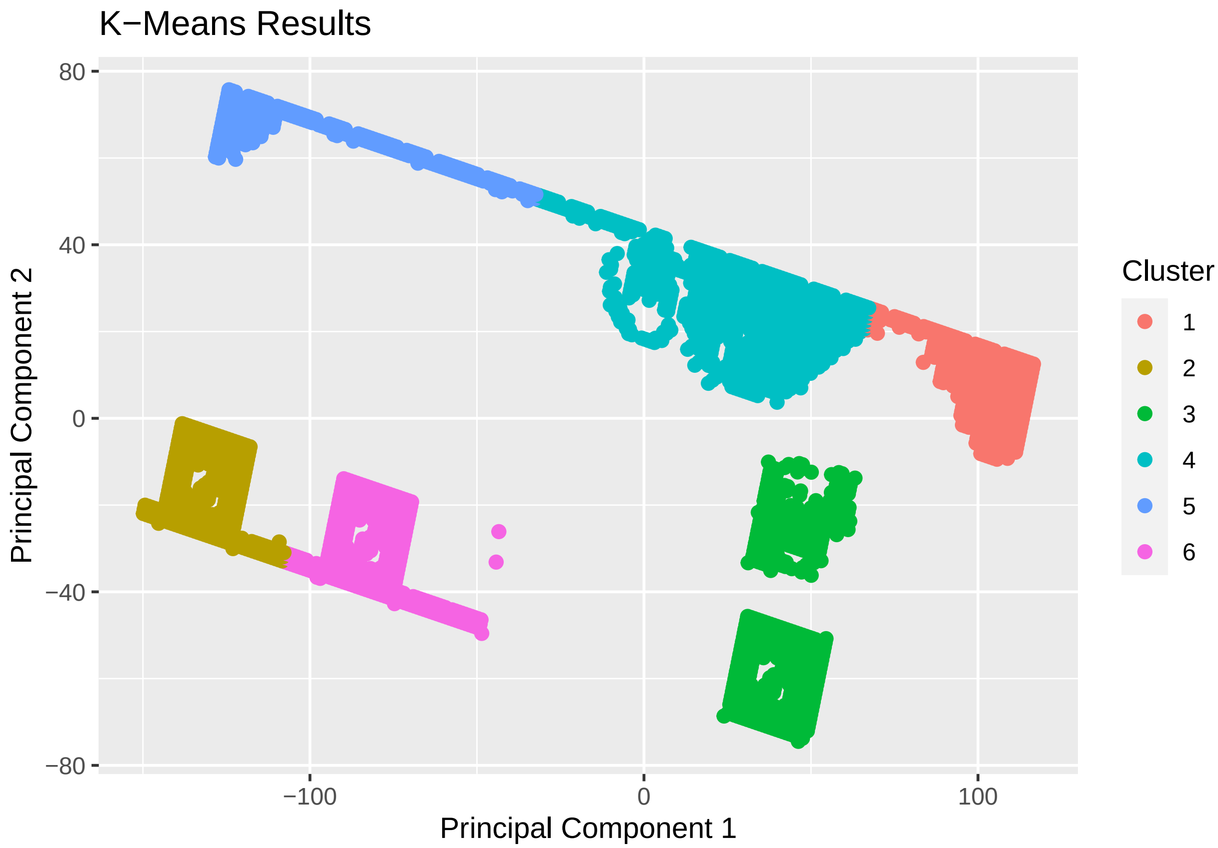

```
clusteringK4 <- kmeans(x=clustering_df, centers=4)
plot(clusteringK4, data=clustering_df)
```



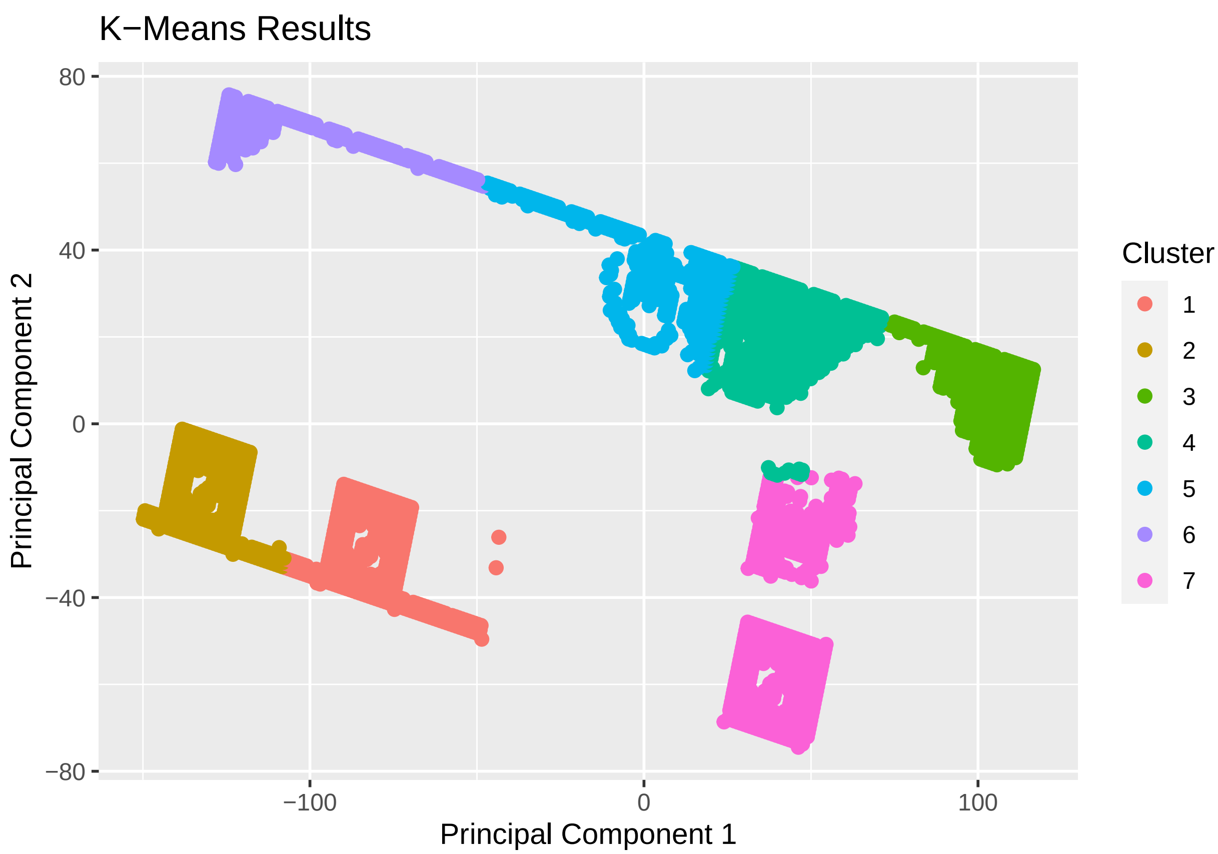

```
clusteringK5 <- kmeans(x=clustering_df, centers=5)
plot(clusteringK5, data=clustering_df)
```



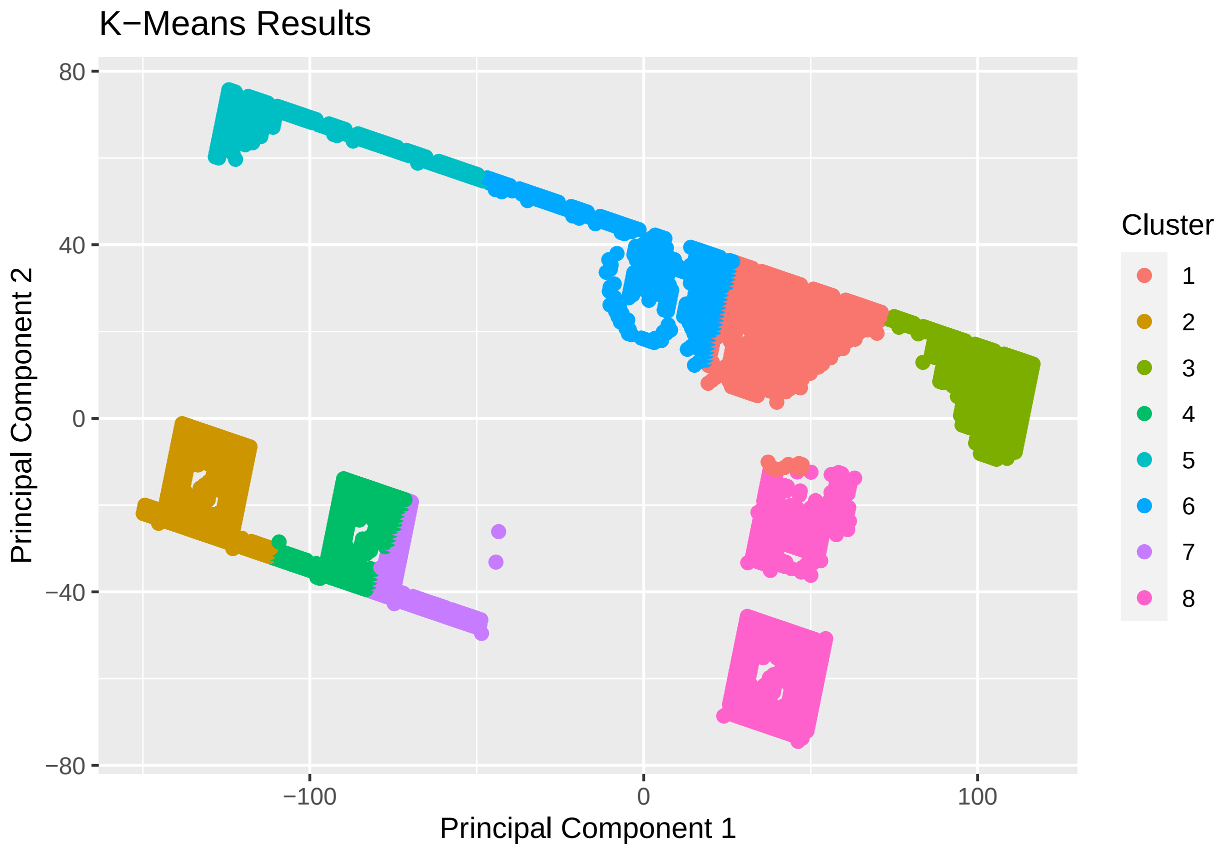

```r
clusteringK6 <- kmeans(x=clustering_df, centers=6)
plot(clusteringK6, data=clustering_df)
```



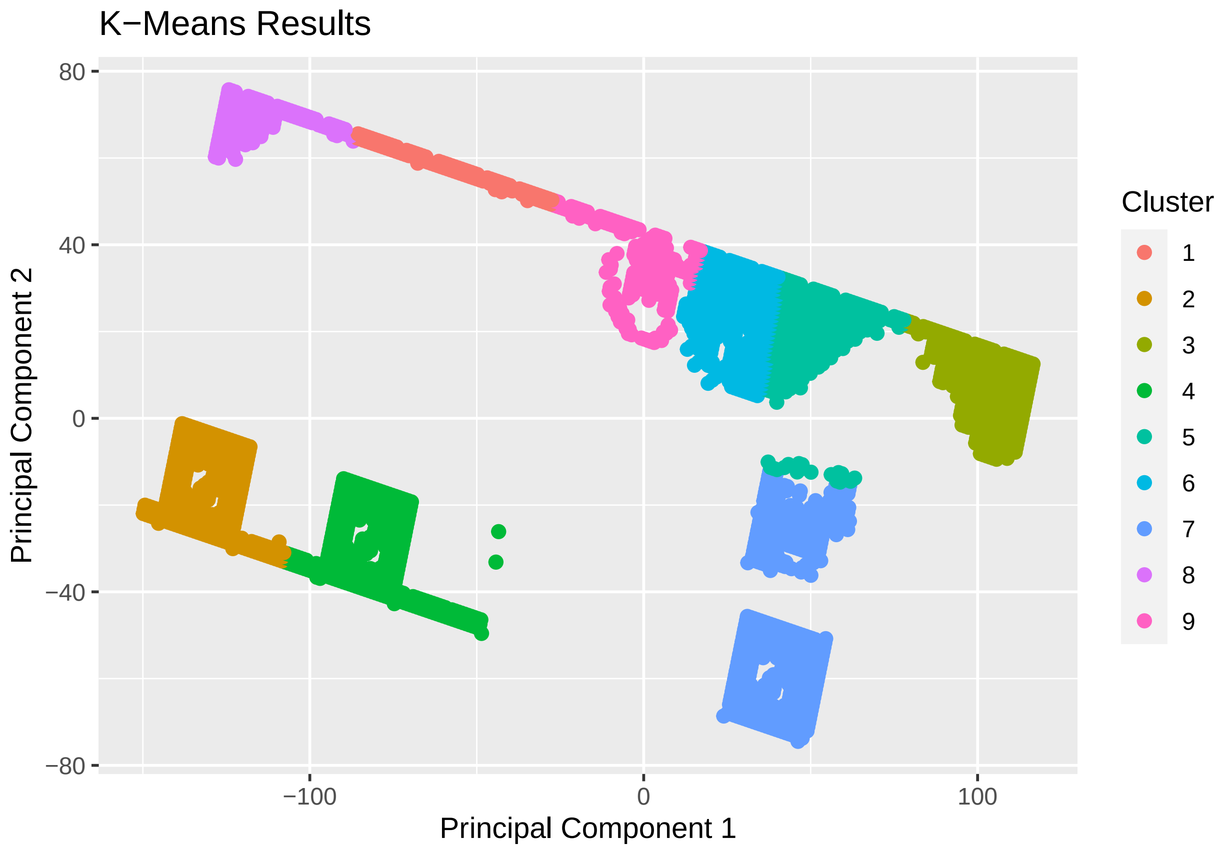

```r
clusteringK7 <- kmeans(x=clustering_df, centers=7)
plot(clusteringK7, data=clustering_df)
```



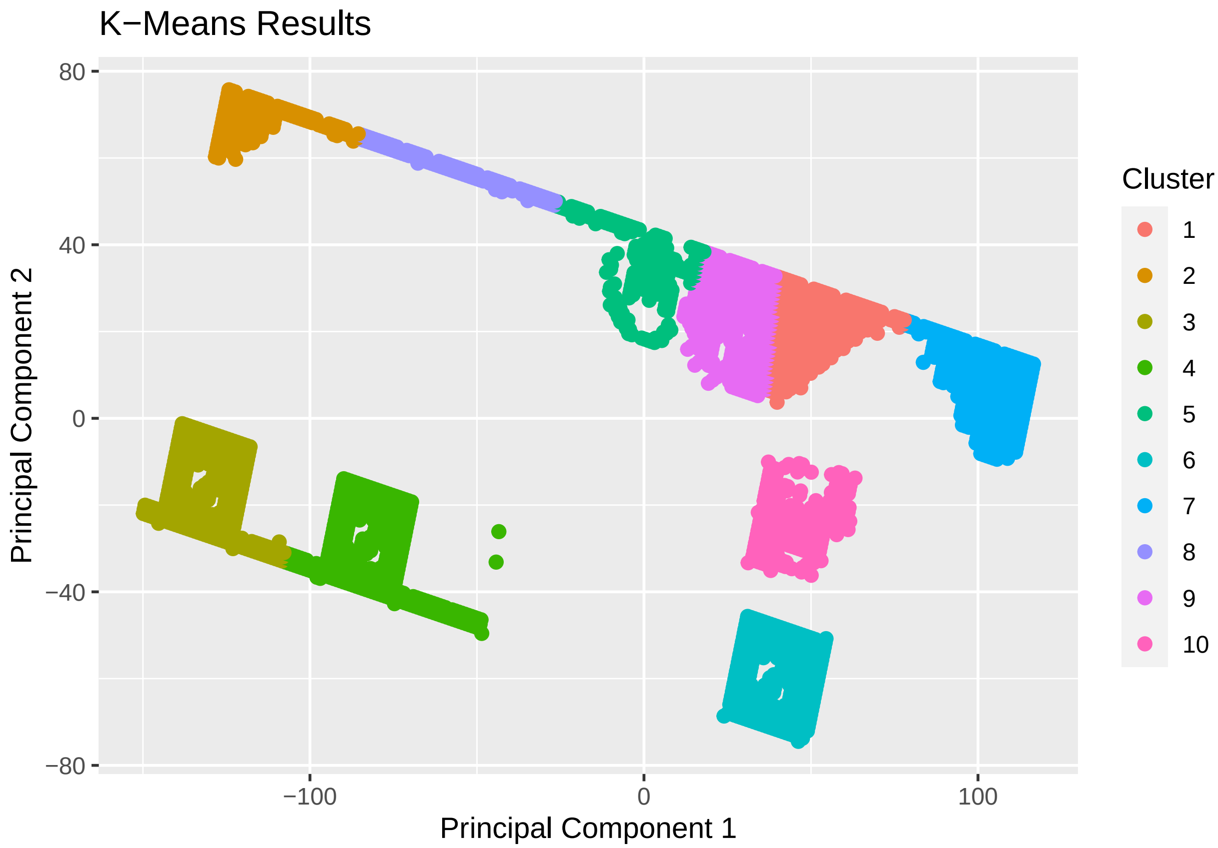

```
clusteringK8 <- kmeans(x=clustering_df, centers=8)
plot(clusteringK8, data=clustering_df)
```



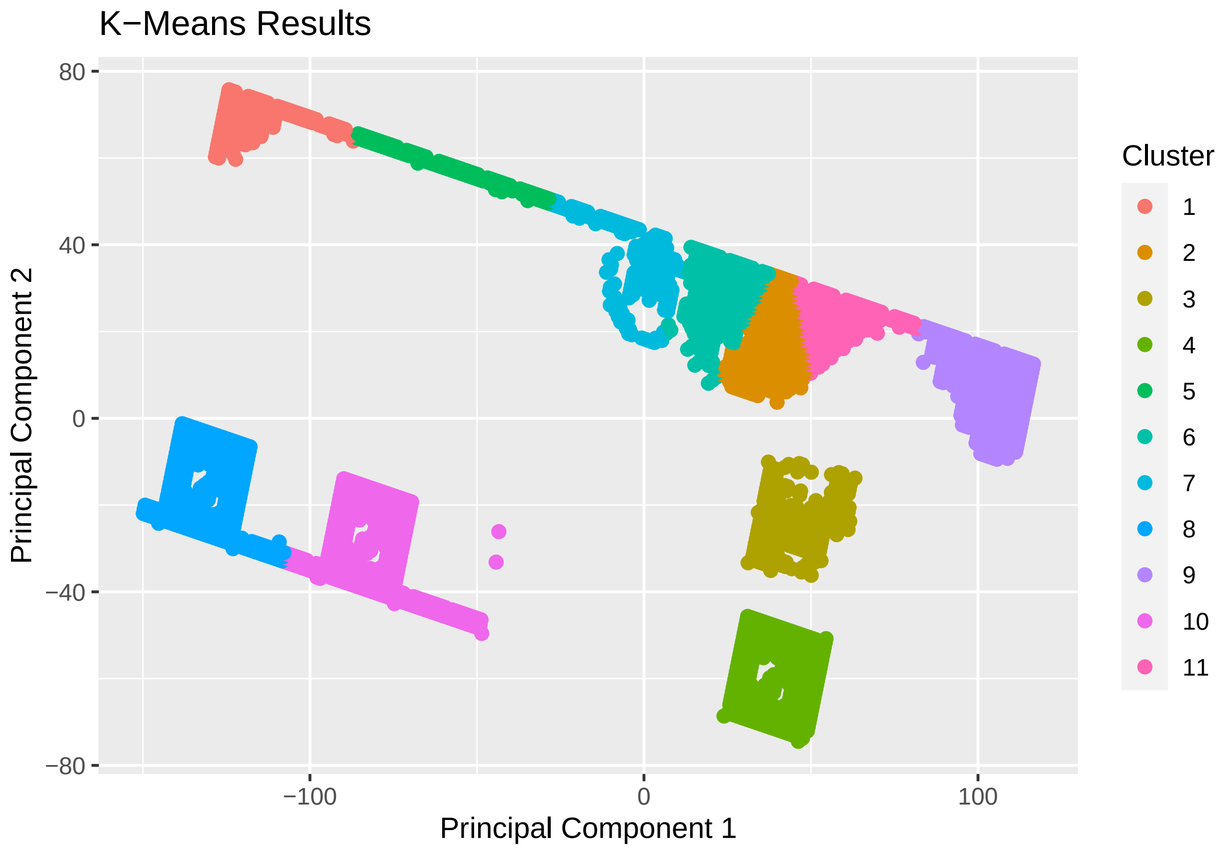

```
clusteringK9 <- kmeans(x=clustering_df, centers=9)
plot(clusteringK9, data=clustering_df)
```



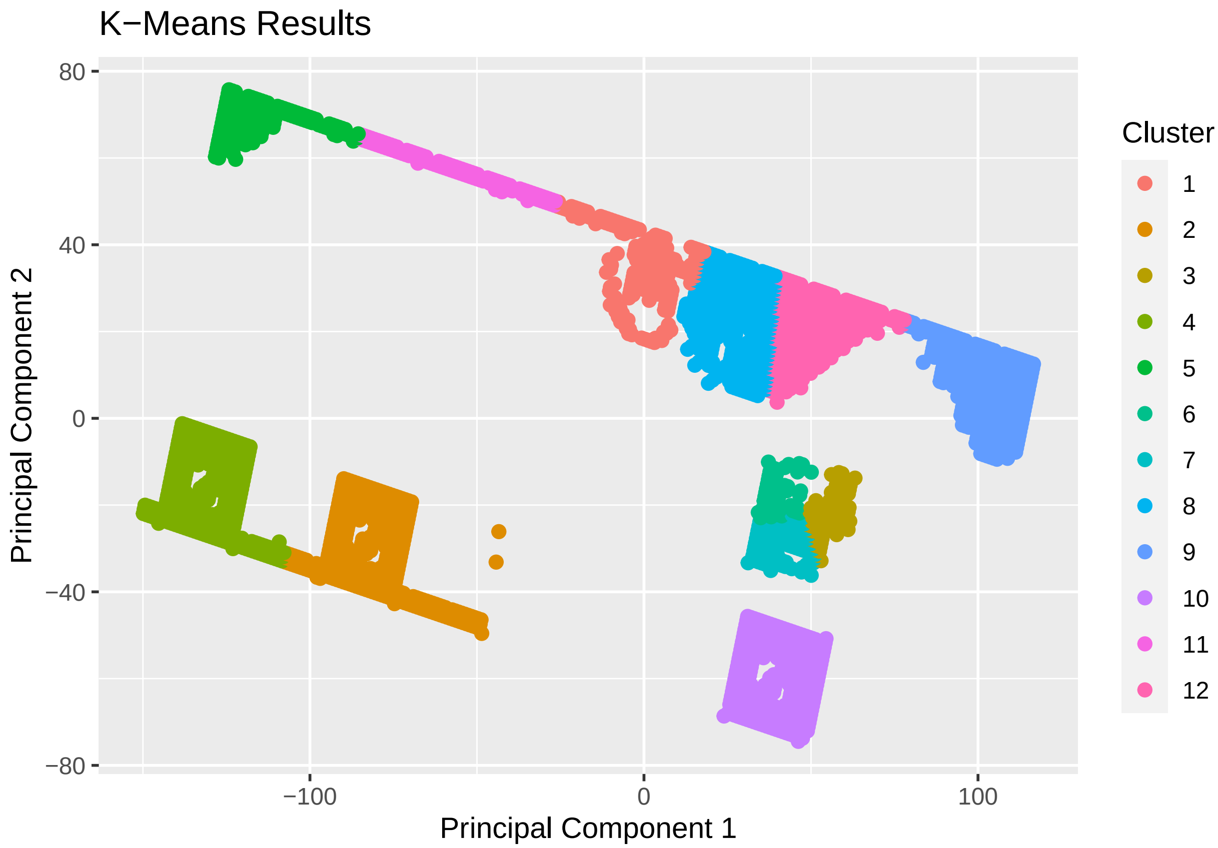

```r
clusteringK10 <- kmeans(x=clustering_df, centers=10)
plot(clusteringK10, data=clustering_df)
```




```
clusteringK11 <- kmeans(x=clustering_df, centers=11)
plot(clusteringK11, data=clustering_df)
```




```
clusteringK12 <- kmeans(x=clustering_df, centers=12)
plot(clusteringK12, data=clustering_df)
```


K−Means Results
Principal Component 2
80
40
0
−40
−80
−100
0
100
Principal Component 1
Cluster
1
2
3
4
5
6
7
8
9
10
11
12