Week11-12_assignment_RiazAhmed_TamimAnsari.R

Riaz

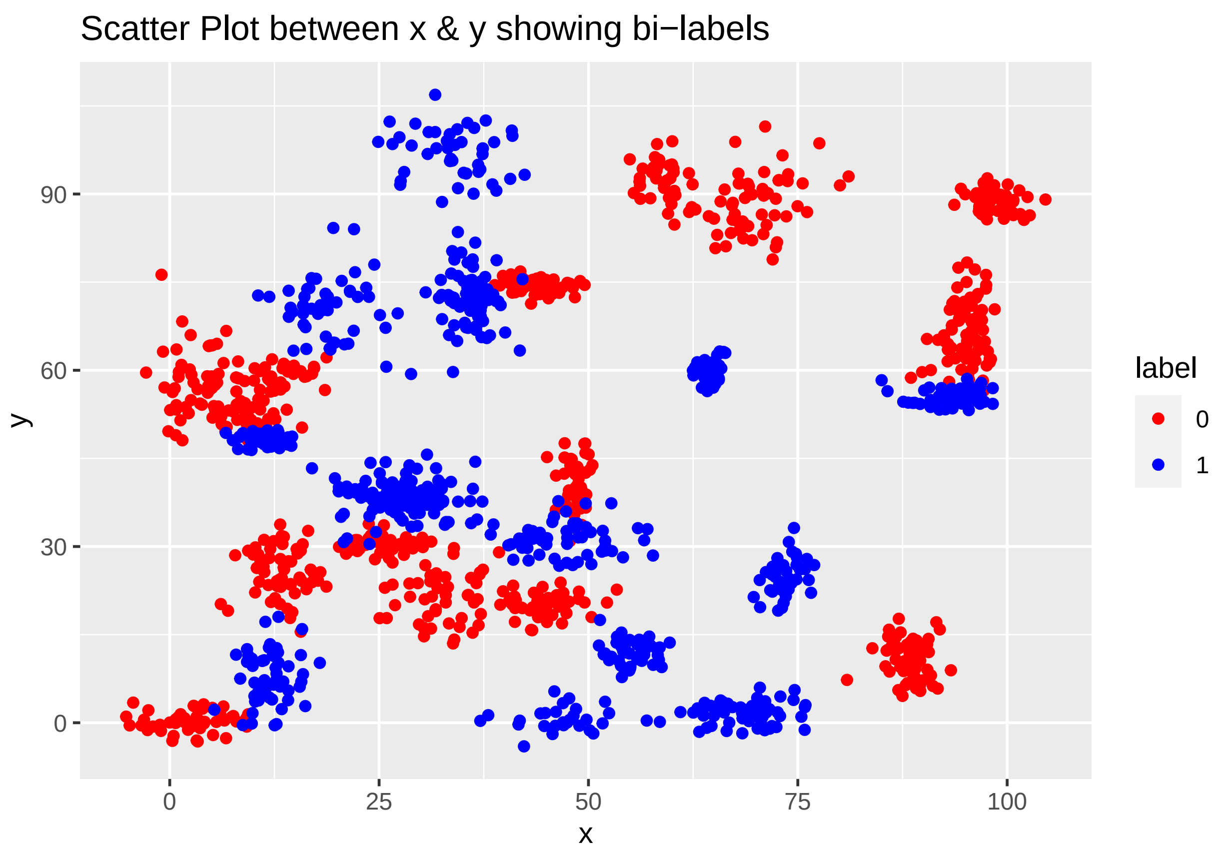
2023-11-18

```
# title: "Week_11&12_Excercise_Riaz"
# author: "Riaz Ahmed Tamim Ansari"
# date: "2023-11-18"

#KNN:
#===

#Read the input data,

binclass <- read.delim("C:\\Users\\Riaz\\Desktop\\MSDS\\Introduction to Statistics\\Week11&12\\binary-cl
triclass <- read.delim("C:\\Users\\Riaz\\Desktop\\MSDS\\Introduction to Statistics\\Week11&12\\trinary-c

#Loading the required libraries

library(ggplot2)
library(class)
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.3.2
```

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

```
#Plot the data from each dataset using a scatter plot,

custom_colors <- c("0" = "red", "1" = "blue")

binclass_plot <- ggplot(binclass, aes(x = x, y = y, color = as.factor(label)))+
         geom_point() + scale_color_manual(values=custom_colors)+
         labs(title = "Scatter Plot between x & y showing bi-labels", color = "label")
binclass_plot
```




```
triclass_plot <- ggplot(triclass, aes(x = x, y = y, colour = as.factor(label)))+
  geom_point() +scale_color_manual(values=c("0" = "red","1" = "blue","2" = "green"))+
  labs(title = "Scatter Plot between x & y showing tri-labels", color = "label")
triclass_plot
```


```
#Split the given binclass and triclass into Test and Train by using base R package.

set.seed(2)
sample_binclass <- sample(c(TRUE,FALSE),nrow(binclass),replace=TRUE,prob=c(0.7,0.3))
sample_triclass <- sample(c(TRUE,FALSE),nrow(triclass),replace=TRUE,prob=c(0.7,0.3))

train_binclass <- binclass[sample_binclass,]
test_binclass <- binclass[!sample_binclass,]
train_triclass <- triclass[sample_triclass,]
test_triclass <- triclass[!sample_triclass,]

#Scaling the values,

train_binclass_scale <- scale(train_binclass[-1])
test_binclass_scale <- scale(test_binclass[-1])
train_triclass_scale <- scale(train_triclass[-1])
test_triclass_scale <- scale(test_triclass[-1])

#Fitting the model,
#Write a function to get the models  for different k values,

model_binclass <- function(kvalue)
{
  knn_model_binclass <- knn(train = train_binclass_scale[, c("x", "y")],
                            test = test_binclass_scale[, c("x", "y")],
                            cl = as.factor(train_binclass$label), k = kvalue)
```

```
  test_binclass$predicted <- as.factor(knn_model_binclass)
  confusionmatrix_binclass <- confusionMatrix(data = factor(test_binclass$predicted, levels = c('0', '1
                                              reference = factor(test_binclass$label, levels = c('0', '
  print(paste("Accuracy for kvalue",kvalue,"is",
              round(confusionmatrix_binclass$overall["Accuracy"],digits=2)))
}

for (i in c(3,5,10,15,20,25))
{
  myvector_binclass <- model_binclass(i)
}
```

```
## [1] "Accuracy for kvalue 3 is 0.95"
## [1] "Accuracy for kvalue 5 is 0.95"
## [1] "Accuracy for kvalue 10 is 0.95"
## [1] "Accuracy for kvalue 15 is 0.95"
## [1] "Accuracy for kvalue 20 is 0.94"
## [1] "Accuracy for kvalue 25 is 0.94"
```

```
model_triclass <- function(kvalue)
{
  knn_model_triclass <- knn(train = train_triclass_scale[, c("x", "y")],
                            test = test_triclass_scale[, c("x", "y")],
                            cl = as.factor(train_triclass$label), k = kvalue)
  test_triclass$predicted <- as.factor(knn_model_triclass)
  confusionmatrix_triclass <- confusionMatrix(data = factor(test_triclass$predicted, levels = c('0', '1
                                              reference = factor(test_triclass$label, levels = c('0', '
  print(paste("Accuracy for kvalue",kvalue,"is",
              round(confusionmatrix_triclass$overall["Accuracy"],digits=2)))
}

for (i in c(3,5,10,15,20,25))
{
  myvector_triclass <- model_triclass(i)
}
```

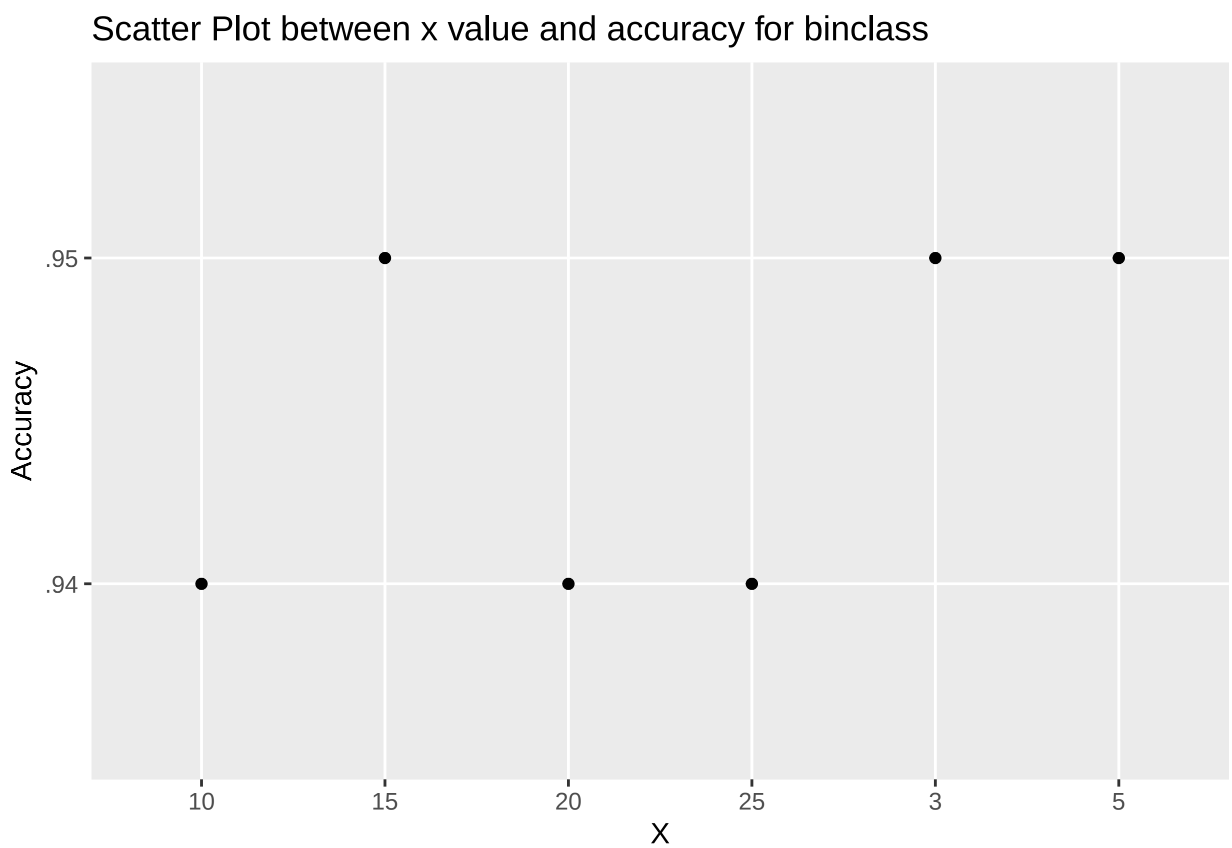
```
## [1] "Accuracy for kvalue 3 is 0.89"
## [1] "Accuracy for kvalue 5 is 0.89"
## [1] "Accuracy for kvalue 10 is 0.89"
## [1] "Accuracy for kvalue 15 is 0.91"
## [1] "Accuracy for kvalue 20 is 0.91"
## [1] "Accuracy for kvalue 25 is 0.9"
```

```
#Graph between k value and accuracy for binary class.

binclass_plot <- data.frame(X=c('3','5','10','15','20','25'),Accuracy=c('.95','.95','.94','.95','.94','
binclass_plot
```

```
##    X Accuracy
## 1  3      .95
## 2  5      .95
## 3 10      .94
```

```
## 4 15      .95
## 5 20      .94
## 6 25      .94
```

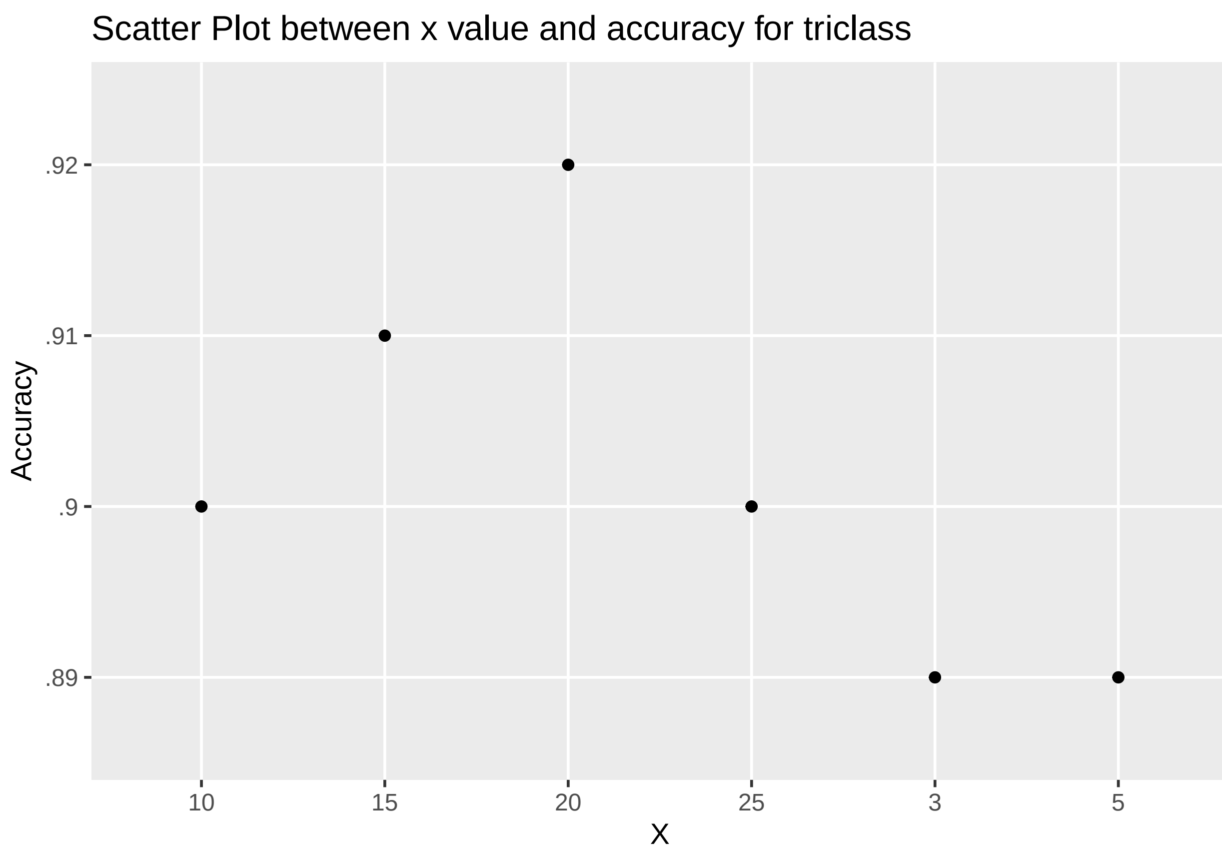
```r
ggplot(binclass_plot, aes(X,Accuracy)) + geom_point() +
  labs(title = "Scatter Plot between x value and accuracy for binclass")
```




```r
#From the above graph, it is evident that for binaryclass model the accuracy is highest for K = 3,5,15

#Graph between k value and accuracy for trinary class.

triclass_plot <- data.frame(X=c('3','5','10','15','20','25'),Accuracy=c('.89','.89','.9','.91','.92','.
triclass_plot
```

```
##    X Accuracy
## 1  3      .89
## 2  5      .89
## 3 10       .9
## 4 15      .91
## 5 20      .92
## 6 25       .9
```

```r
ggplot(triclass_plot, aes(x=X,y=Accuracy)) + geom_point() +
  labs(title = "Scatter Plot between x value and accuracy for triclass")
```

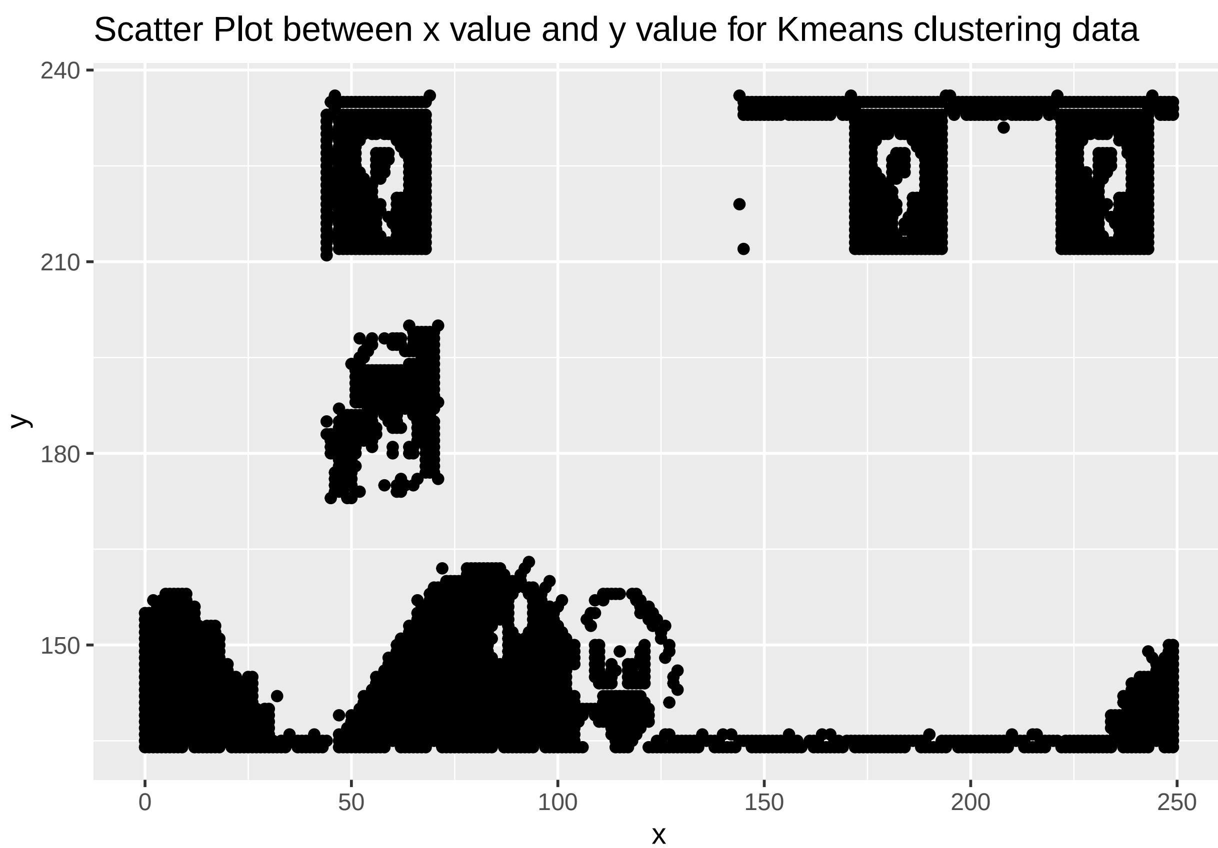



```
#From the above graph, it is evident that for triclass model the accuracy is highest for K = 20

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

# No, linear classifier would not work well, if we are looking at the initial plots.  Because to apply
#linear classifier the predictor and predicted variables should be linear.  But looking at the plots
#for x and y there is no linearity and the observations are randomly distributed.  Moreover, linear
#regression is used to predict continuous variable responses, whereas this is a categorical prediction.

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

#Last week's, logistic regression classifier gave very poor accuracy of 54%.  This is because logistic
#regression assumes a linear decision boundary, whereas KNN does not assume that.  Hence KNN is better
#in predicting, if there is a non linear relationship between predictor and response variables.


#KMeans Clustering:
#=================

#Read the input data,

clusterdata <- read.delim("C:\\Users\\Riaz\\Desktop\\MSDS\\Introduction to Statistics\\Week11&12\\clust
head(clusterdata)
```

```
##      x    y
```

```
## 1  46 236
## 2  69 236
## 3 144 236
## 4 171 236
## 5 194 236
## 6 195 236
```

```r
#Plot the dataset using a scatter plot.
ggplot(clusterdata, aes(x,y)) +
  geom_point() + labs(title = "Scatter Plot between x value and y value for Kmeans clustering data")
```




```r
#Scale the data
clusterdata <- scale(clusterdata)
head(clusterdata)
```

```
##               x        y
## [1,] -0.8482235 1.561107
## [2,] -0.5415045 1.561107
## [3,]  0.4586659 1.561107
## [4,]  0.8187273 1.561107
## [5,]  1.1254462 1.561107
## [6,]  1.1387818 1.561107
```

```r
# Initialize total within sum of squares error: wss
wss <- c()
wss
```

```
## NULL
```

```r
#Fit the dataset using Kmeans algorithm,

set.seed(123)
# Look from 2 to 12 possible clusters
for (i in 2:12) {
  # Fit the model: km.out
  print (i)
  km.out <- kmeans(clusterdata, centers = i, nstart = 20)
  print(km.out$tot.withinss)
  # Save the within cluster sum of squares
  wss[i] <- km.out$tot.withinss
}
```

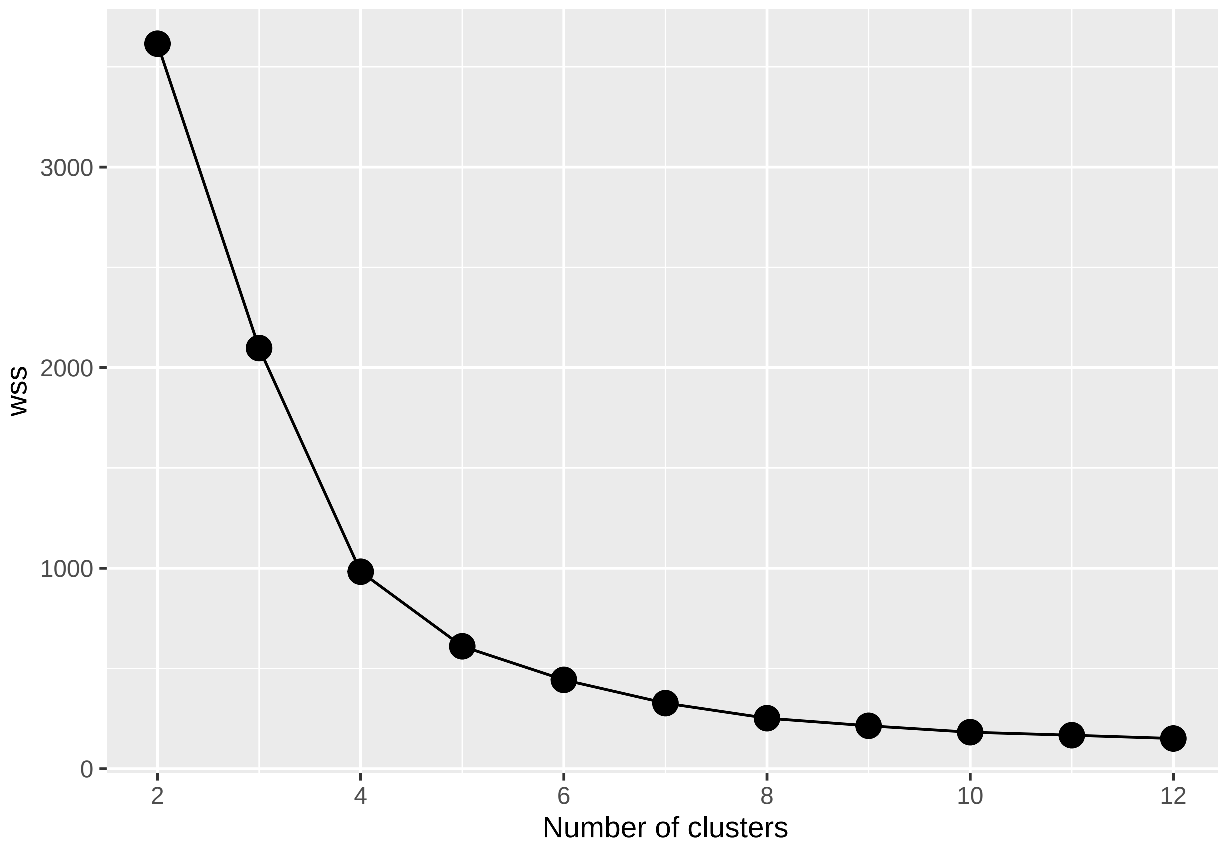
```
## [1] 2
## [1] 3615.245
## [1] 3
## [1] 2097.592
## [1] 4
## [1] 982.5633
## [1] 5
## [1] 610.6719
## [1] 6
## [1] 443.3626
## [1] 7
## [1] 327.211
## [1] 8
## [1] 252.1299
## [1] 9
## [1] 214.324
## [1] 10
## [1] 182.3197
## [1] 11
## [1] 167.2537
## [1] 12
## [1] 150.8658
```

```r
wss <- na.omit(wss)

#Create a dataframe of wss for easy understanding and plotting purpose,

wss_df = data.frame(clusters = 2:12, wss = wss)
wss_df
```

```
##    clusters       wss
## 1         2 3615.2454
## 2         3 2097.5918
```

```
## 3          4  982.5633
## 4          5  610.6719
## 5          6  443.3626
## 6          7  327.2110
## 7          8  252.1299
## 8          9  214.3240
## 9         10  182.3197
## 10        11  167.2537
## 11        12  150.8658
```

```
#Calculate this average distance from the center of each cluster for each value of k and plot it as a l
#chart where k is the x-axis and the average distance is the y-axis.
#Plotting the wss,
ggplot(wss_df, aes(x = clusters, y = wss, group = 1)) +
  geom_point(size = 4)+
  geom_line() +
  scale_x_continuous(breaks = c(2, 4, 6, 8, 10,12)) +
  xlab('Number of clusters')
```




```
#Elbow point of dataset,

#I would say the elbow point would be 7 as there is not much difference in the reduction of WSS, by inc
#the number of clusters further.

#Choosing the centers as 7 and performing the kmeans once again,
set.seed(123)
```

```r
km.final <- kmeans(clusterdata, centers = 7, nstart = 20)
km.final
```

```
## K-means clustering with 7 clusters of sizes 315, 521, 341, 441, 1349, 464, 591
##
## Cluster means:
##            x          y
## 1  1.4879630 -0.9853019
## 2  0.9347803  1.2845090
## 3 -0.6771679  0.2920086
## 4 -0.7102710  1.2249978
## 5 -0.3069600 -0.8018532
## 6  1.6222263  1.2583422
## 7 -1.2693894 -0.8474261
##
## Clustering vector:
##    [1] 4 4 2 2 2 2 6 6 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 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 2 2
##   [75] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
##  [112] 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 4 2 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 6 6 6 6 6 6 6 6 6 6 6 6 6 6
##  [186] 6 6 6 6 6 6 6 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 2 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 2 2
##  [260] 2 2 2 2 2 2 2 2 2 2 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
##  [297] 6 6 6 6 6 6 6 6 6 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 2 2 2 2 2
##  [334] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
##  [371] 6 6 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [408] 2 2 2 2 2 2 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 4 4 4 4 4 4 4 4 4 4
##  [445] 4 4 4 4 4 4 4 4 4 4 4 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 6 6 6 6 6
##  [482] 6 6 6 6 6 6 6 6 6 6 6 6 6 6 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 2 2 2 2 2 2 2 2
##  [519] 2 2 2 2 2 2 6 6 6 6 6 6 6 6 6 6 6 6 6 6 4 4 4 4 4 4 4 4 4 4 4 4 4 2 2 2 2
##  [556] 2 2 2 2 2 2 2 2 6 6 6 6 6 6 6 6 6 6 6 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 2
##  [593] 2 2 2 2 2 2 2 2 2 2 2 2 2 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 4 4 4 4 4 4 4 4 4
##  [630] 4 4 4 4 4 4 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 6 6 6 6 6 6 6 6 6 6 6 6 6 6 4 4
##  [667] 4 4 4 4 4 4 4 4 4 4 4 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 6 6 6 6 6 6 6 6 6 6 6
##  [704] 6 6 6 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 6 6 6
##  [741] 6 6 6 6 6 6 6 6 6 6 6 6 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 2 2 2 2 2 2 2 2 2
##  [778] 2 2 2 2 2 6 6 6 6 6 6 6 6 6 6 6 6 6 6 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 2 2
##  [815] 2 2 2 2 2 2 2 2 2 2 2 2 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 4 4 4 4 4 4 4 4 4 4
##  [852] 4 4 4 4 4 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 4
##  [889] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 6
##  [926] 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
##  [963] 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
## [1000] 6 6 6 6 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 2 2 2 2 2 2 2 2 2 2 2 2 2
## [1037] 2 2 2 2 2 2 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 4 4 4 4 4 4 4 4 4 4 4 4
## [1074] 4 4 4 4 4 4 4 4 4 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 6 6 6 6 6 6 6 6
## [1111] 6 6 6 6 6 6 6 6 6 6 6 6 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 2 2 2 2 2
## [1148] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 4 4 4
## [1185] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 6 6
## [1222] 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 2 2
## [1259] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 4 4 4 4 4
## [1296] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [1333] 2 2 2 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 4 4 4 4 4 4 4 4 4 4 4 4
```

```
## [1370] 4 4 4 4 4 4 4 4 4 4 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 6 6 6
## [1407] 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 4 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [1444] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [1481] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [1518] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [1555] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [1592] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [1629] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [1666] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [1703] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [1740] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 5 5 5 5 5 5 5 5 5
## [1777] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [1814] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 7 7 7 7 7 7 5 5 5 5 5 5 5 5 5
## [1851] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 7 7 7 7 7 7 7 7 5 5 5 5 5 5
## [1888] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 7 7 7 7 7 7 7 7 7 7 5 5 5 5
## [1925] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 7 7 7 7 7 7 7 7 7 7 7
## [1962] 7 7 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 7
## [1999] 7 7 7 7 7 7 7 7 7 7 7 7 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [2036] 5 5 5 5 5 5 5 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 5 5 5 5 5 5 5 5 5 5 5 5 5
## [2073] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
## [2110] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 7 7 7 7 7
## [2147] 7 7 7 7 7 7 7 7 7 7 7 7 7 7 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [2184] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 5 5 5
## [2221] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [2258] 5 5 5 5 1 1 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 5 5 5 5 5 5 5 5 5 5 5 5
## [2295] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 1 1 1
## [2332] 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [2369] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 1 1 1 1 7 7 7
## [2406] 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [2443] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 1 1 1
## [2480] 1 1 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [2517] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [2554] 5 5 1 1 1 1 1 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 5 5 5 5 5 5
## [2591] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [2628] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 1 1 1 1 1 1 1 1 1 7 7 7 7 7 7 7 7 7 7 7 7 7
## [2665] 7 7 7 7 7 7 7 7 7 7 7 7 7 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [2702] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 1 1 1 1
## [2739] 1 1 1 1 1 1 1 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 5 5 5 5
## [2776] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [2813] 5 5 5 5 5 5 5 5 1 1 1 1 1 1 1 1 1 1 1 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
## [2850] 7 7 7 7 7 7 7 7 7 7 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [2887] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 1 1
## [2924] 1 1 1 1 1 1 1 1 1 1 1 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
## [2961] 7 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [2998] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 1 1 1 1 1 1 1 1 1
## [3035] 1 1 1 1 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 5 5 5
## [3072] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [3109] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 1 1 1 1 1 1
## [3146] 1 1 1 1 1 1 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
## [3183] 7 7 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [3220] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 1 1
## [3257] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
## [3294] 7 7 7 7 7 7 7 7 7 7 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [3331] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
```

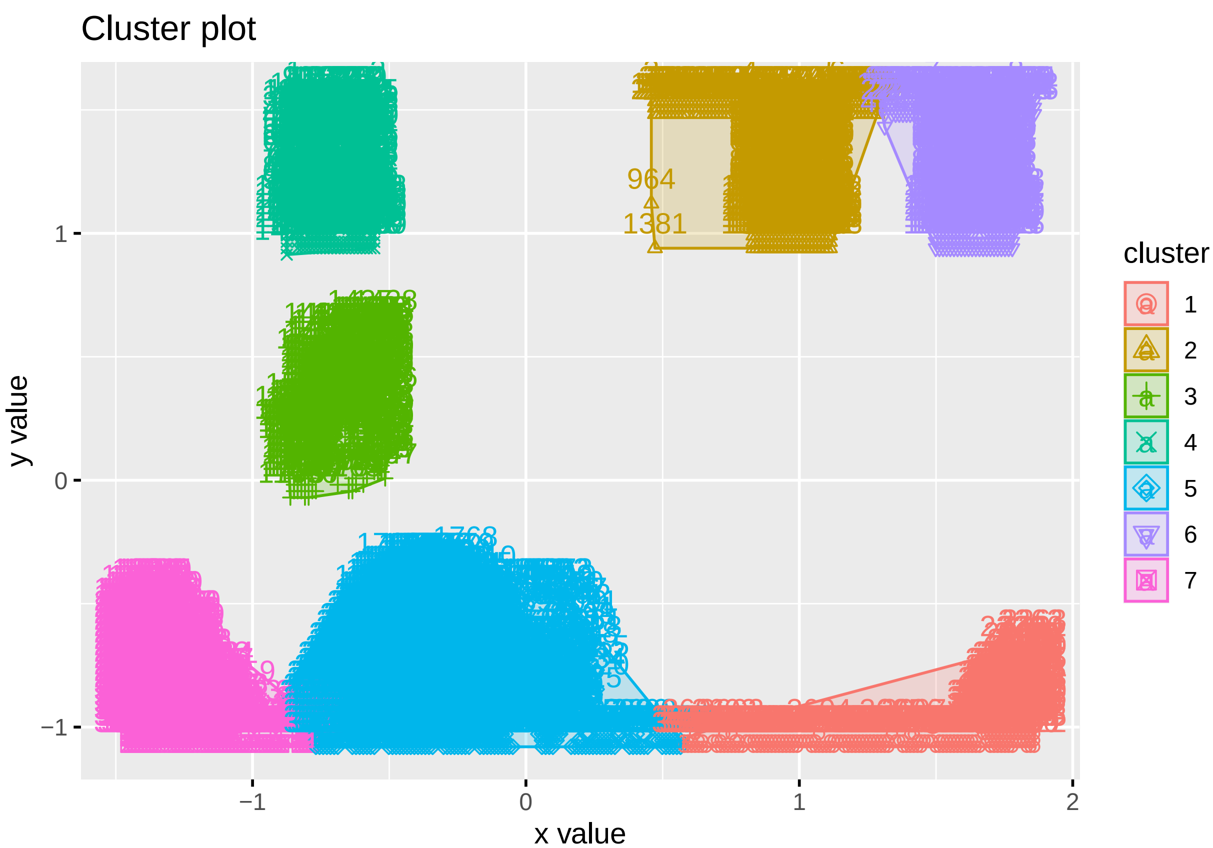
```
## [3368] 5 5 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
## [3405] 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [3442] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [3479] 5 5 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
## [3516] 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [3553] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [3590] 5 5 5 5 5 5 5 5 5 5 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 7 7 7 7
## [3627] 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
## [3664] 7 7 7 7 7 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [3701] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [3738] 5 5 5 5 5 5 5 5 5 5 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3775] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3812] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
## [3849] 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [3886] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
## [3923] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3960] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [3997] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##
## Within cluster sum of squares by cluster:
## [1]  51.85475  36.27618  13.78287  19.52538 149.57157  24.58883  31.61138
##  (between_SS / total_SS =  95.9 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
#Using the visualization function provided from the factoextra package,
library(factoextra)
```

```
## Warning: package 'factoextra' was built under R version 4.3.2
```

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

```
fviz_cluster(km.final,data = clusterdata )
```


Cluster plot
y value
x value
cluster
1
2
3
4
5
6
7
−1
0
1
2
964
1381