McKibben DSC520 Ex. 10.2

Makayla McKibben

2024-08-06

DSC520 McKibben Ex. 10.2

Get All Packages and Libraries Needed

```r
#install.packages("ggplot2")
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.4.1
```

```r
#install.packages("factoextra")
library(factoextra)
```

```
## Warning: package 'factoextra' was built under R version 4.4.1

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

```r
#install.packages("tidyverse")
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.4.1

## Warning: package 'tibble' was built under R version 4.4.1

## Warning: package 'tidyr' was built under R version 4.4.1

## Warning: package 'readr' was built under R version 4.4.1

## Warning: package 'purrr' was built under R version 4.4.1

## Warning: package 'dplyr' was built under R version 4.4.1

## Warning: package 'forcats' was built under R version 4.4.1

## Warning: package 'lubridate' was built under R version 4.4.1
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.3     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```
library(gridExtra)
```

```
## Warning: package 'gridExtra' was built under R version 4.4.1

##
## Attaching package: 'gridExtra'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
#install.packages("caTools")
library(caTools)
```

```
## Warning: package 'caTools' was built under R version 4.4.1
```

```
library(class)
#install.packages("caret")
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.4.1

## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

Set Working Directory and Import Data for Binary, Trinary, and Cluster Datsets

```
# Set working directory
setwd("C:/Users/makay/OneDrive/Documents/DSC520")
clust <- as.data.frame(read.csv(file = 'clustering-data.csv',
         header = TRUE, sep =",", stringsAsFactors = FALSE))

binary <- read.csv(file = 'binary-classifier-data.csv',
```

```r
          header = TRUE, sep =",", stringsAsFactors = FALSE)

trinary <- read.csv(file = 'trinary-classifier-data.csv',
           header = TRUE, sep =",", stringsAsFactors = FALSE)
```

Check data structures

```r
head(binary, 8)
```

```
##   label        x        y
## 1     0 70.88469 83.17702
## 2     0 74.97176 87.92922
## 3     0 73.78333 92.20325
## 4     0 66.40747 81.10617
## 5     0 69.07399 84.53739
## 6     0 72.23616 86.38403
## 7     0 70.92514 89.73168
## 8     0 77.57454 98.63425
```

```r
head(trinary, 8)
```

```
##   label        x        y
## 1     0 30.08387 39.63094
## 2     0 31.27613 51.77511
## 3     0 34.12138 49.27575
## 4     0 32.58222 41.23300
## 5     0 34.65069 45.47956
## 6     0 33.80513 44.24656
## 7     0 33.63327 53.35537
## 8     0 30.32783 31.24890
```

Code from Week 9

From Week 9 Clean the Data

```r
# Remove rows missing data
binary <- binary[complete.cases(binary),]

# Create model
binary_model <- glm(label ~ x + y, data = binary, family = binomial)

# Check model
summary(binary_model)
```

```
##
## Call:
## glm(formula = label ~ x + y, family = binomial, data = binary)
```

```
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.424809   0.117224   3.624  0.00029 ***
## x           -0.002571   0.001823  -1.411  0.15836
## y           -0.007956   0.001869  -4.257 2.07e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2075.8  on 1497  degrees of freedom
## Residual deviance: 2052.1  on 1495  degrees of freedom
## AIC: 2058.1
##
## Number of Fisher Scoring iterations: 4
```

```
# Predict results using model
pred_binary <- predict.glm(binary_model)

# Bind results to single dataframe
binary_acc <- cbind(binary$label, pred_binary)
colnames(binary_acc) <- c("Data", "Model")
binary_acc <- data.frame(binary_acc)

# Check results
head(binary_acc)
```

```
##   Data      Model
## 1    0 -0.4191462
## 2    0 -0.4674600
## 3    0 -0.4984068
## 4    0 -0.3911610
## 5    0 -0.4253135
## 6    0 -0.4481342
```

```
# Make necessary transformations
results_model_bin <- ifelse(binary_acc$Model >= 0.5, "Positive", "Negative")
results_data_bin <- ifelse(binary_acc$Data == 1, "Positive", "Negative")
results_bin <- cbind(results_data_bin, results_model_bin)
colnames(results_bin) <- c("Data", "Model")
results_bin <- data.frame(results_bin)

# Find percent accuracy
num_correct_binary <- length(which(results_bin$Data == results_bin$Model))
percent_correct_binary <- (num_correct_binary/length(results_bin$Data))*100
percent_correct_binary
```

```
## [1] 51.2016
```

Week 9 Regression Model Accuracy is 51.2%

Week 10

Check Data Structures

```r
head(binary, 8)
```

```
##   label        x        y
## 1     0 70.88469 83.17702
## 2     0 74.97176 87.92922
## 3     0 73.78333 92.20325
## 4     0 66.40747 81.10617
## 5     0 69.07399 84.53739
## 6     0 72.23616 86.38403
## 7     0 70.92514 89.73168
## 8     0 77.57454 98.63425
```

```r
head(trinary, 8)
```

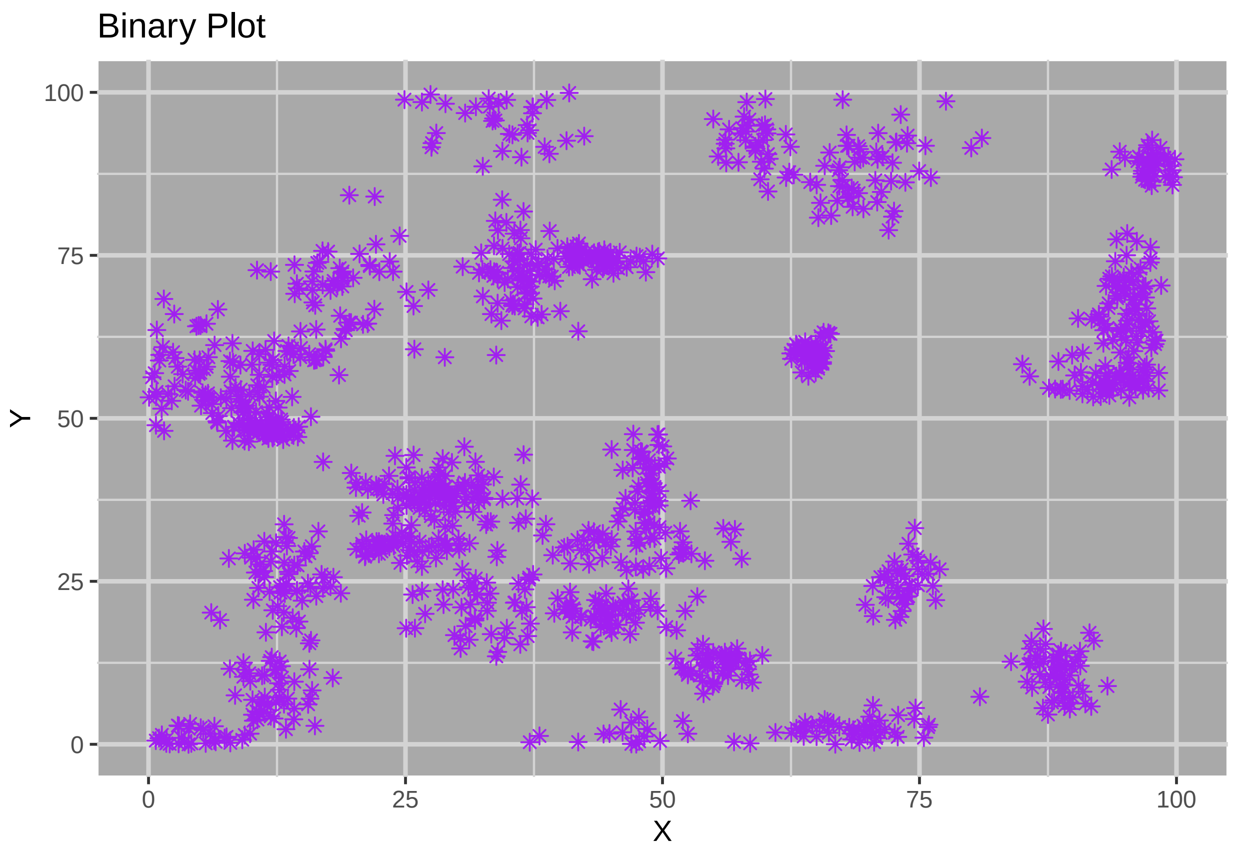
```
##   label        x        y
## 1     0 30.08387 39.63094
## 2     0 31.27613 51.77511
## 3     0 34.12138 49.27575
## 4     0 32.58222 41.23300
## 5     0 34.65069 45.47956
## 6     0 33.80513 44.24656
## 7     0 33.63327 53.35537
## 8     0 30.32783 31.24890
```

```r
head(clust, 8)
```

```
##     x   y
## 1  46 236
## 2  69 236
## 3 144 236
## 4 171 236
## 5 194 236
## 6 195 236
## 7 221 236
## 8 244 236
```

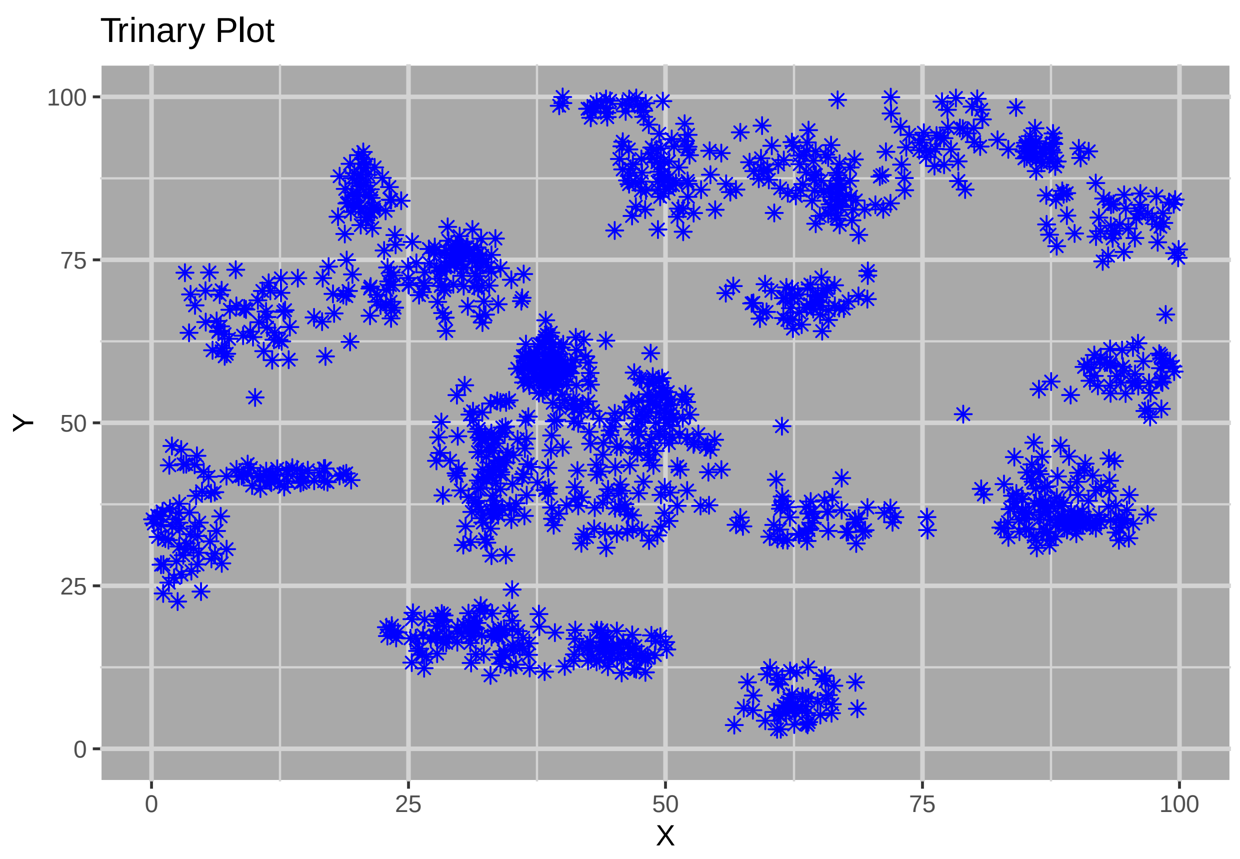
```r
# Base Plots
bin_plot <-ggplot(binary, aes(x, y))
bin_plot + geom_point(color = "purple", shape = 8, size = 1.8) +
  theme(panel.grid = element_line(color = "lightgrey", linewidth = 0.8,
        linetype=1),   panel.background = element_rect(color = "white",
        fill = "darkgrey")) + labs(title = "Binary Plot", x ="X", y = "Y") +
        xlim(0, 100) + ylim(0, 100)
```

```
## Warning: Removed 81 rows containing missing values or values outside the scale range
## (‘geom_point()‘).
```



```
trin_plot <-ggplot(trinary, aes(x, y))
trin_plot + geom_point(color = "blue", shape = 8, size = 1.8) +
theme(panel.grid = element_line(color = "lightgrey", linewidth = 0.8, linetype=1),
        panel.background = element_rect(color = "white", fill = "darkgrey")) +
  labs(title = "Trinary Plot", x ="X",
       y = "Y") + xlim(0, 100) + ylim(0, 100)
```

```
## Warning: Removed 79 rows containing missing values or values outside the scale range
## (‘geom_point()‘).
```



### I do not believe a linear fit works for either of these models. It may be slightly better for the trinary plot but not an ideal model by a long shot.

KNN Binary Models and Plots

```r
# Create objects to submit to the knn function
bin_train_index <- createDataPartition(binary$label,
                                       times = 1, p = 0.8, list = FALSE)
bin_train <- binary[bin_train_index, ]
bin_test <- binary[-bin_train_index, ]
bin_label <- binary$label[1:length(bin_train_index)]

# Transform to dataframes
bin_train <- as.data.frame(bin_train)
bin_test <- as.data.frame(bin_test)

# Scale data length
bin_train_scale <- scale(bin_train[1:length(bin_train_index), , ])
bin_test_scale <- scale(bin_test[1:length(bin_train_index), , ])

# Create list with all k values
k <- list(3, 5, 10, 15, 20, 25)

# Make binary knn models
bin_pred_3 <- knn.cv(train = bin_train_scale, cl = bin_label, k = k[1])
```

```r
bin_pred_5 <- knn.cv(train = bin_train_scale, cl = bin_label, k = k[2])
bin_pred_10 <- knn.cv(train = bin_train_scale, cl = bin_label, k = k[3])
bin_pred_15 <- knn.cv(train = bin_train_scale, cl = bin_label, k = k[4])
bin_pred_20 <- knn.cv(train = bin_train_scale, cl = bin_label, k = k[5])
bin_pred_25 <- knn.cv(train = bin_train_scale, cl = bin_label, k = k[6])

# Create Tables of Binary KNN Data
bm3 <- table(bin_label, bin_pred_3)
bm5 <- table(bin_label, bin_pred_5)
bm10 <- table(bin_label, bin_pred_10)
bm15 <- table(bin_label, bin_pred_15)
bm20 <- table(bin_label, bin_pred_20)
bm25 <- table(bin_label, bin_pred_25)

# Find Accuracy of Binary KNN Models
acc3b <- sum(diag(bm3))/length(bin_label)
acc5b <- sum(diag(bm5))/length(bin_label)
acc10b <- sum(diag(bm10))/length(bin_label)
acc15b <- sum(diag(bm15))/length(bin_label)
acc20b <- sum(diag(bm20))/length(bin_label)
acc25b <- sum(diag(bm25))/length(bin_label)

# Print Accuracies for the Kmeans Clusters
sprintf("Accuracy for k = 3: %.2f%%", acc3b*100)
```

```
## [1] "Accuracy for k = 3: 95.66%"
```

```r
sprintf("Accuracy for k = 5: %.2f%%", acc5b*100)
```

```
## [1] "Accuracy for k = 5: 95.33%"
```

```r
sprintf("Accuracy for k = 10: %.2f%%", acc10b*100)
```

```
## [1] "Accuracy for k = 10: 95.08%"
```

```r
sprintf("Accuracy for k = 15: %.2f%%", acc15b*100)
```

```
## [1] "Accuracy for k = 15: 95.83%"
```

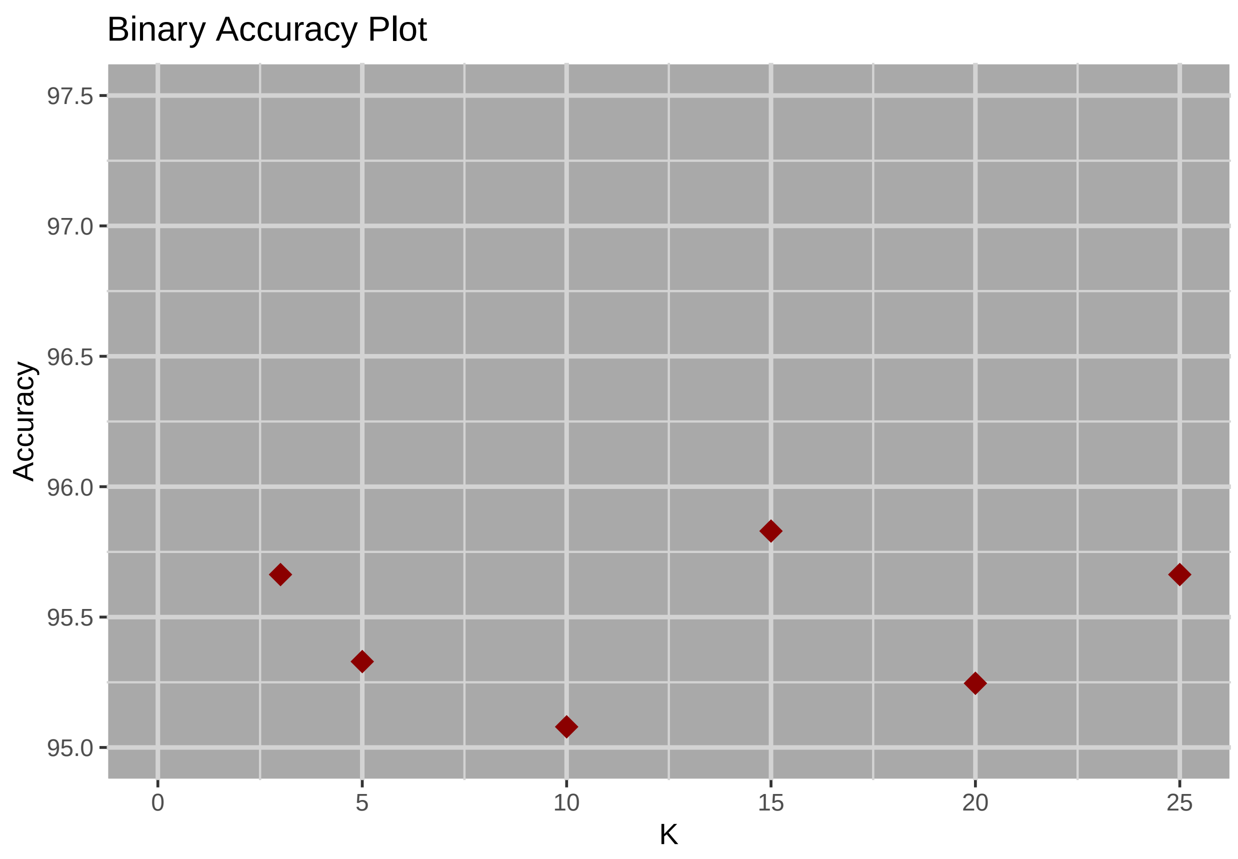
```r
sprintf("Accuracy for k = 20: %.2f%%", acc20b*100)
```

```
## [1] "Accuracy for k = 20: 95.25%"
```

```r
sprintf("Accuracy for k = 25: %.2f%%", acc25b*100)
```

```
## [1] "Accuracy for k = 25: 95.66%"
```

```r
# Make List of Accuracy Values
accb <- list(acc3b*100, acc5b*100, acc10b*100,
             acc15b*100, acc20b*100, acc25b*100)

# Accuracy Plot of Binary Dataset
accb <- as.numeric(accb)
k <- as.numeric(k)
bin_acc <- cbind(k[1:6], accb[1:6])
bin_acc <- as.data.frame(bin_acc)
bin_acc_plot <- ggplot(bin_acc, aes(x = k, y = accb,
                                    group = 1))
bin_acc_plot + geom_point(color = "darkred",
             shape = 18, size = 3.8) + theme(panel.grid =
             element_line(color = "lightgrey", linewidth=0.8, linetype=1),
             panel.background = element_rect(color = "white", fill = "darkgrey")) +
             labs(title = "Binary Accuracy Plot", x ="K", y = "Accuracy") +
             xlim(0, 25) + ylim(95, 97.5)
```



The model from last weeks regression was 51.2% compared to mid to upper 90s for the small values of k for kmeans clusters of both the binary dataset plotted above and the trinary dataset plotted in the next section. Kmeans grouping can find the optimal number of chunks/groups/areas to break thse datasets into while the regular scatter plot regression from last week tries to come up with a single fit that will work with all of the chunks/groups/areas/data points simultaneously.

KNN Trinary Models and Plots

```
# Check Data Structure
head(trinary, 8)
```

```
##   label        x        y
## 1     0 30.08387 39.63094
## 2     0 31.27613 51.77511
## 3     0 34.12138 49.27575
## 4     0 32.58222 41.23300
## 5     0 34.65069 45.47956
## 6     0 33.80513 44.24656
## 7     0 33.63327 53.35537
## 8     0 30.32783 31.24890
```

```
# Create Objects for the KNN Model of the Trinary Dataset
trin_train_index <- createDataPartition(trinary$label, times = 1,
                    p = 0.8, list = FALSE)
trin_train <- trinary[trin_train_index, ]
trin_test <- trinary[-trin_train_index, ]
trin_label <- trinary$label[1:length(trin_train_index)]

# Transform to Dataframe
trin_train <- as.data.frame(trin_train)
trin_test <- as.data.frame(trin_test)

# Scale Values
trin_train_scale <- scale(trin_train[1:1255, , ])
trin_test_scale <- scale(trin_test[1:1255, , ])

# Re-establish the List of K Values for KNN
k <- list(3, 5, 10, 15, 20, 25)

# Make Trinary KNN Models
trin_pred_3 <- knn.cv(train = trin_train_scale,
                      cl = trin_label, k = k[1])
trin_pred_5 <- knn.cv(train = trin_train_scale,
                      cl = trin_label, k = k[2])
trin_pred_10 <- knn.cv(train = trin_train_scale,
                       cl = trin_label, k = k[3])
trin_pred_15 <- knn.cv(train = trin_train_scale,
                       cl = trin_label, k = k[4])
trin_pred_20 <- knn.cv(train = trin_train_scale,
                       cl = trin_label, k = k[5])
trin_pred_25 <- knn.cv(train = trin_train_scale,
```

```r
                        cl = trin_label, k = k[6])

# Make Trinary Tables
cm3 <- table(trin_label, trin_pred_3)
cm5 <- table(trin_label, trin_pred_5)
cm10 <- table(trin_label, trin_pred_10)
cm15 <- table(trin_label, trin_pred_15)
cm20 <- table(trin_label, trin_pred_20)
cm25 <- table(trin_label, trin_pred_25)

# Find Accuracy of Trinary KNN Models
acc3 <- sum(diag(cm3))/length(trin_label)
acc5 <- sum(diag(cm5))/length(trin_label)
acc10 <- sum(diag(cm10))/length(trin_label)
acc15 <- sum(diag(cm15))/length(trin_label)
acc20 <- sum(diag(cm20))/length(trin_label)
acc25 <- sum(diag(cm25))/length(trin_label)

# Print Accuracy of Trinary KNN Models
sprintf("Accuracy for k = 3: %.2f%%", acc3*100)
```

```
## [1] "Accuracy for k = 3: 97.13%"
```

```r
sprintf("Accuracy for k = 5: %.2f%%", acc5*100)
```

```
## [1] "Accuracy for k = 5: 97.77%"
```

```r
sprintf("Accuracy for k = 10: %.2f%%", acc10*100)
```

```
## [1] "Accuracy for k = 10: 97.77%"
```

```r
sprintf("Accuracy for k = 15: %.2f%%", acc15*100)
```

```
## [1] "Accuracy for k = 15: 97.85%"
```

```r
sprintf("Accuracy for k = 20: %.2f%%", acc20*100)
```

```
## [1] "Accuracy for k = 20: 97.77%"
```

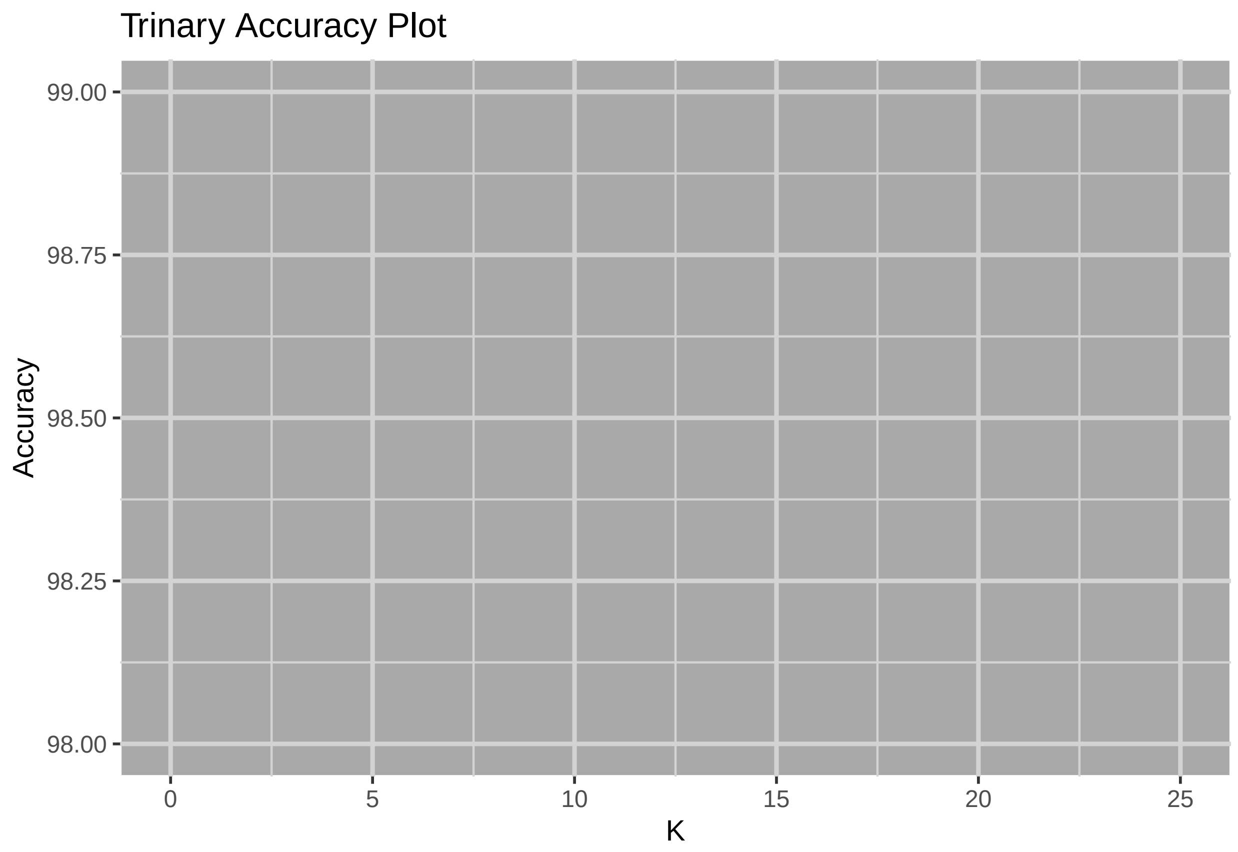
```r
sprintf("Accuracy for k = 25: %.2f%%", acc25*100)
```

```
## [1] "Accuracy for k = 25: 97.77%"
```

```r
# Accuracy Trinary List
acct <- list(acc3*100, acc5*100, acc10*100,
             acc15*100, acc20*100, acc25*100)

# Accuracy Plot Trinary
k <- as.numeric(k)
```

```
acct <- as.numeric(unlist(acct))
trin_acc <- cbind(k[1:6], acct)
trin_acc <- as.data.frame(trin_acc)
trin_acc_plot <- ggplot(trin_acc, aes(x = k, y = acct, group = 1))
trin_acc_plot + geom_point(color = "darkred", shape = 18,
                          size = 3.8) + theme(panel.grid = element_line
                         (color = "lightgrey", linewidth = 0.8, linetype=1),
panel.background = element_rect(color = "white", fill = "darkgrey")) +
  labs(title = "Trinary Accuracy Plot", x ="K", y = "Accuracy") +
  xlim(0, 25) + ylim(98, 99)
```

```
## Warning: Removed 6 rows containing missing values or values outside the scale range
## (‘geom_point()‘).
```

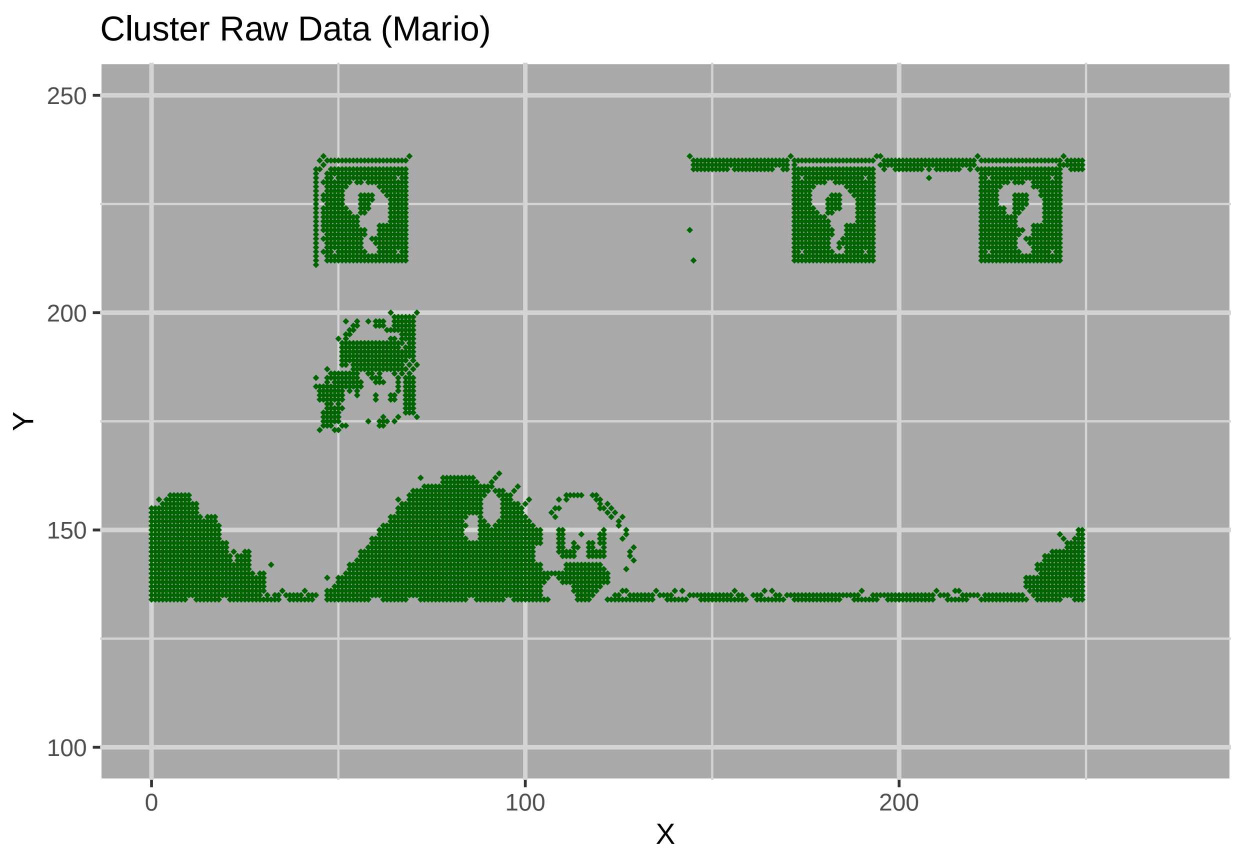


K-means clustering

Look at clustering data

```
# Set working directory
setwd("C:/Users/makay/OneDrive/Documents/DSC520")
# Import Cluster Data
clust <- as.data.frame(read.csv(file = 'clustering-data.csv', header = TRUE,
```

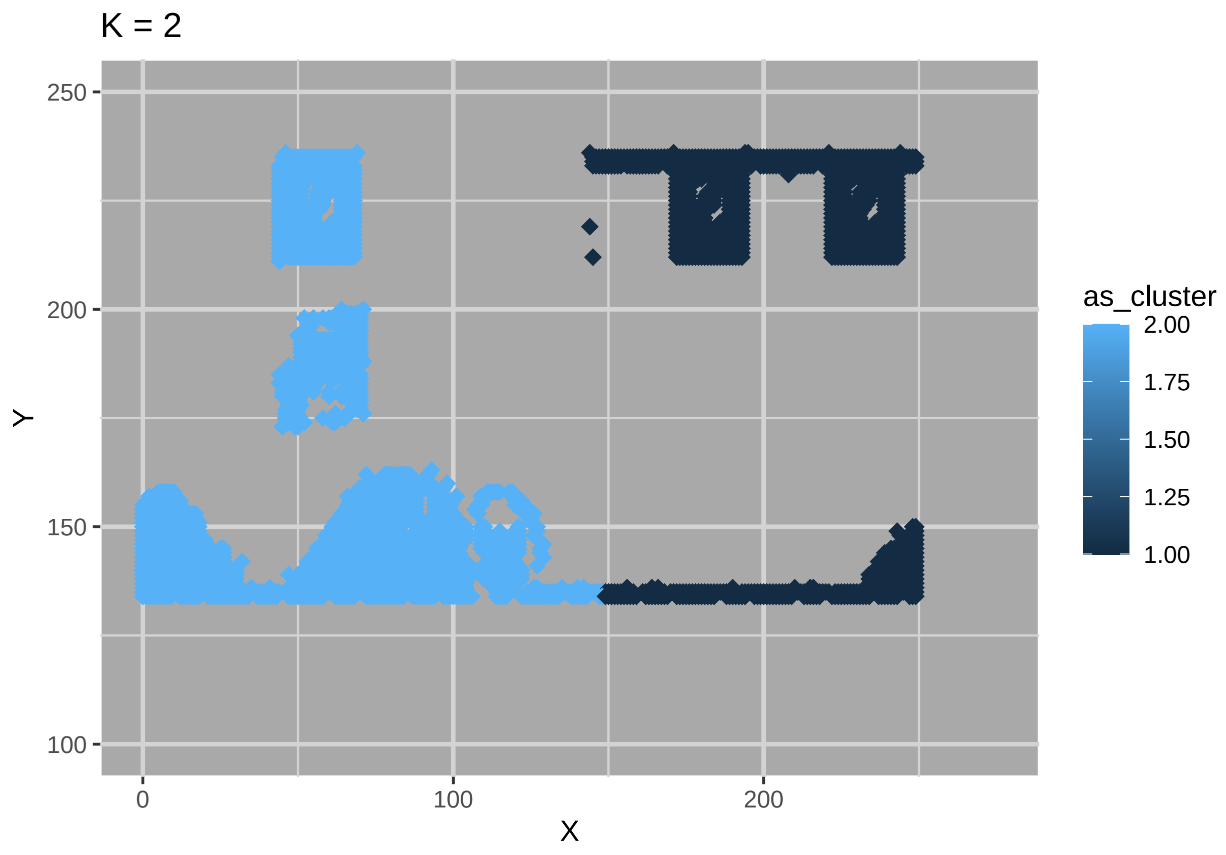
```r
                    sep =",", stringsAsFactors = FALSE))
# Check Data Structure
clust <- data.frame(clust)

# Plot Cluster Data
clust_plot <- ggplot(clust, aes(x = x, y = y, group = 1))
clust_plot + geom_point(color = "darkgreen", shape = 18, size = 0.8) +
  theme(panel.grid = element_line(color = "lightgrey",
  linewidth = 0.8, linetype = 1), panel.background = element_rect(color =
  "white", fill = "darkgrey")) + labs(title = "Cluster Raw Data (Mario)",
  x ="X", y = "Y") + xlim(0, 275) + ylim(100, 250)
```



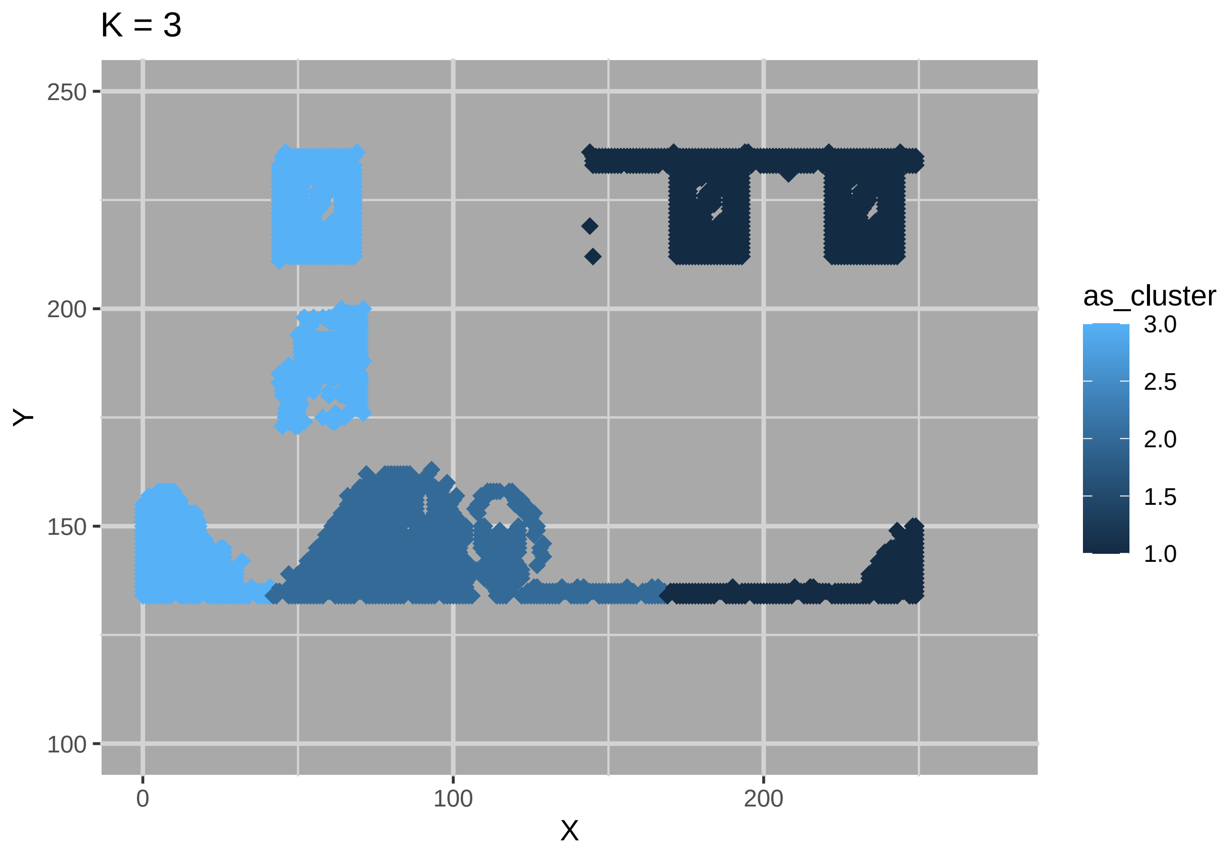
Make Kmeans Models and Create Their Plot Objects

The following two sections of code would be better in a loop, creating all the variables and plots but I'm not super comfortable with loops in R yet. My computer is also auto setting my directory to the wrong thing so I need to sort that out, for now I'm brute forcing it to work by manually setting the directory myself as the clust.

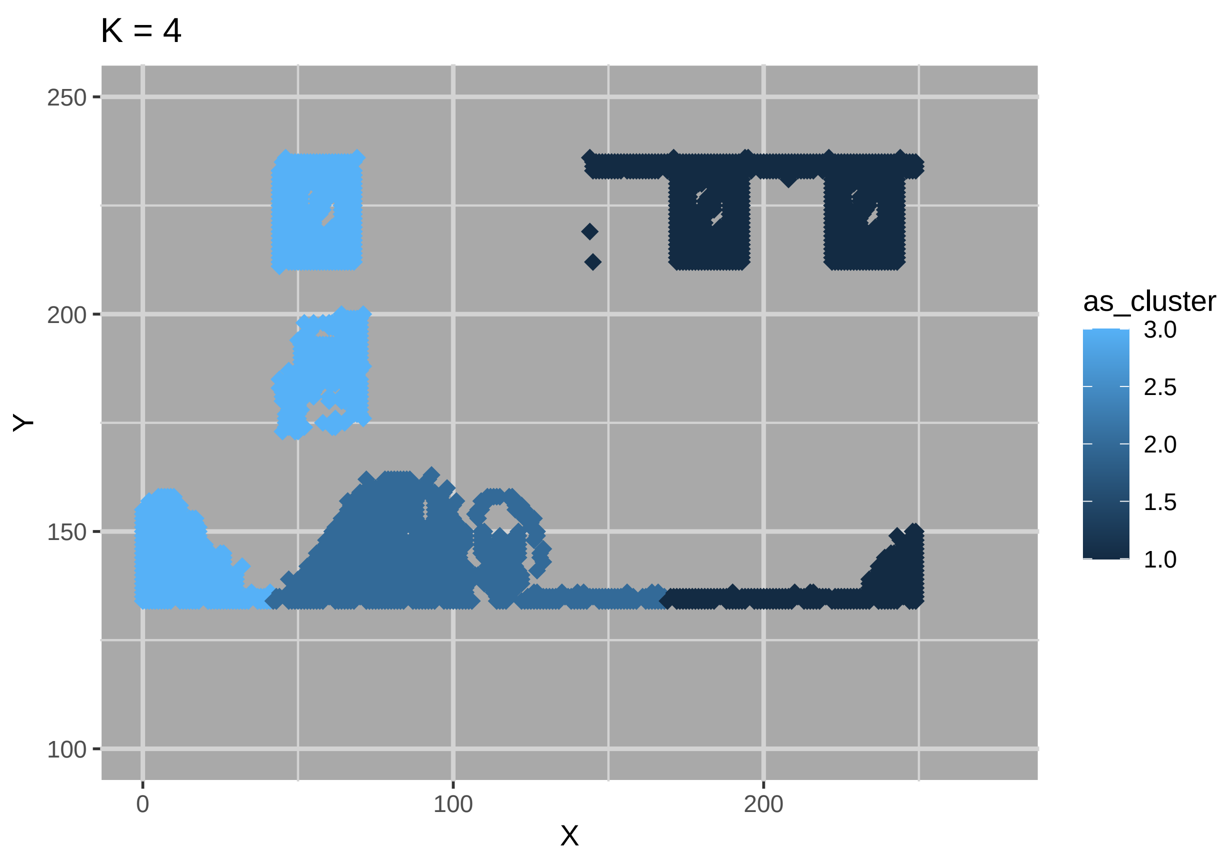
```r
# Set wd
setwd("C:/Users/makay/OneDrive/Documents/DSC520")
# Import Cluster Data
clust <- as.data.frame(read.csv(file = 'clustering-data.csv',
                                 header = TRUE, sep =",",
                                 stringsAsFactors = FALSE))
clust <- data.frame(clust)
km2 <- kmeans(clust, centers = 2, nstart = 20)
km3 <- kmeans(clust, centers = 3, nstart = 20)
km4 <- kmeans(clust, centers = 4, nstart = 20)
km5 <- kmeans(clust, centers = 5, nstart = 20)
km6 <- kmeans(clust, centers = 6, nstart = 20)
km7 <- kmeans(clust, centers = 7, nstart = 20)
km8 <- kmeans(clust, centers = 8, nstart = 20)
km9 <- kmeans(clust, centers = 9, nstart = 20)
km10 <- kmeans(clust, centers = 10, nstart = 20)
km11 <- kmeans(clust, centers = 11, nstart = 20)
km12 <- kmeans(clust, centers = 12, nstart = 20)

# Kmeans Plots
# K=2
df2 <- cbind(km2$cluster, clust$x, clust$y)
colnames(df2)<- c("as_cluster", "x", "y")
p2 <- ggplot(df2, aes(x, y, color = as_cluster))
p2 + geom_point(shape = 18, size = 2.8) + theme(panel.grid =
   element_line(color = "lightgrey", linewidth = 0.8, linetype = 1),
   panel.background = element_rect(color = "white", fill = "darkgrey")) +
   labs(title = "K = 2", x ="X", y = "Y") + xlim(0, 275) + ylim(100, 250)
```



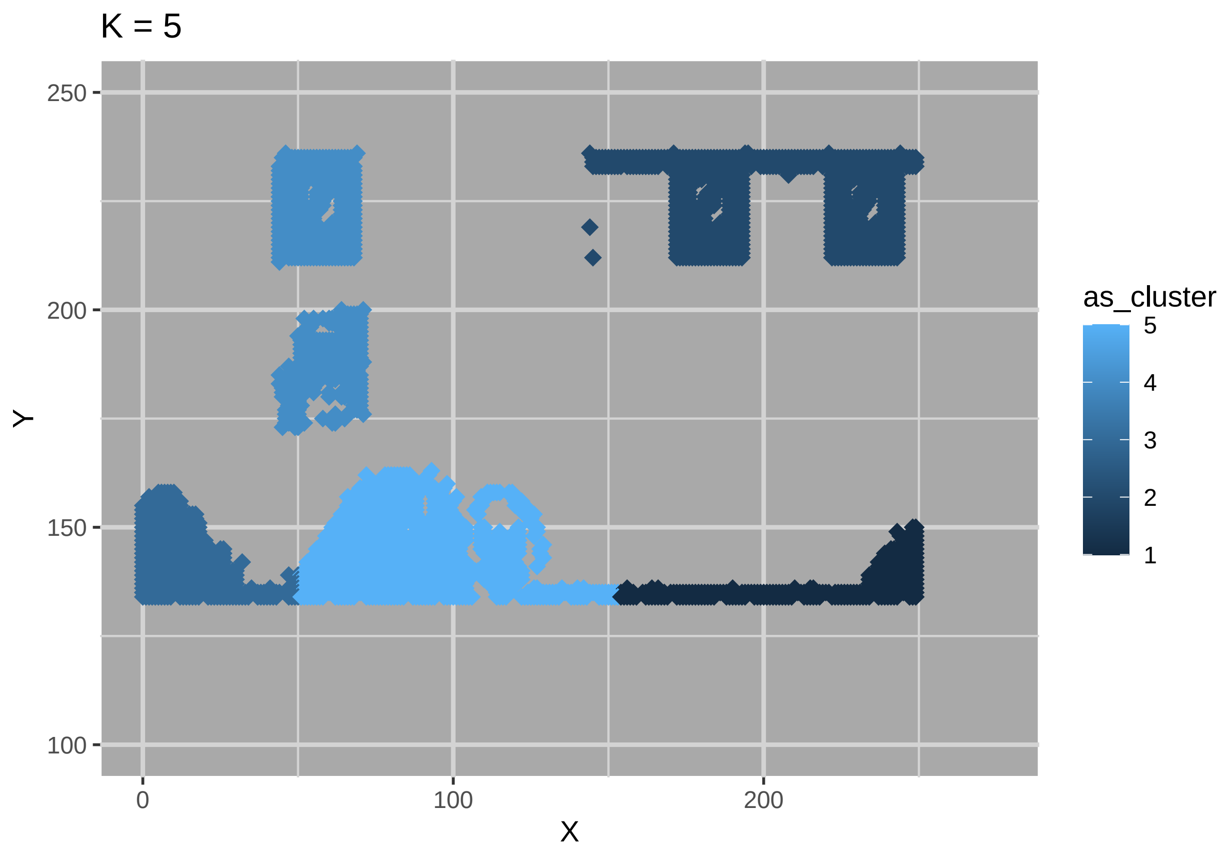
```r
# K=3
df3 <- cbind(km3$cluster, clust$x, clust$y)
colnames(df3)<- c("as_cluster", "x", "y")
p3 <- ggplot(df3, aes(x, y, color = as_cluster))
p3 + geom_point(shape = 18, size = 2.8) + theme(panel.grid =
   element_line(color = "lightgrey", linewidth = 0.8, linetype = 1),
   panel.background = element_rect(color = "white", fill = "darkgrey")) +
   labs(title = "K = 3", x ="X", y = "Y") + xlim(0, 275) + ylim(100, 250)
```



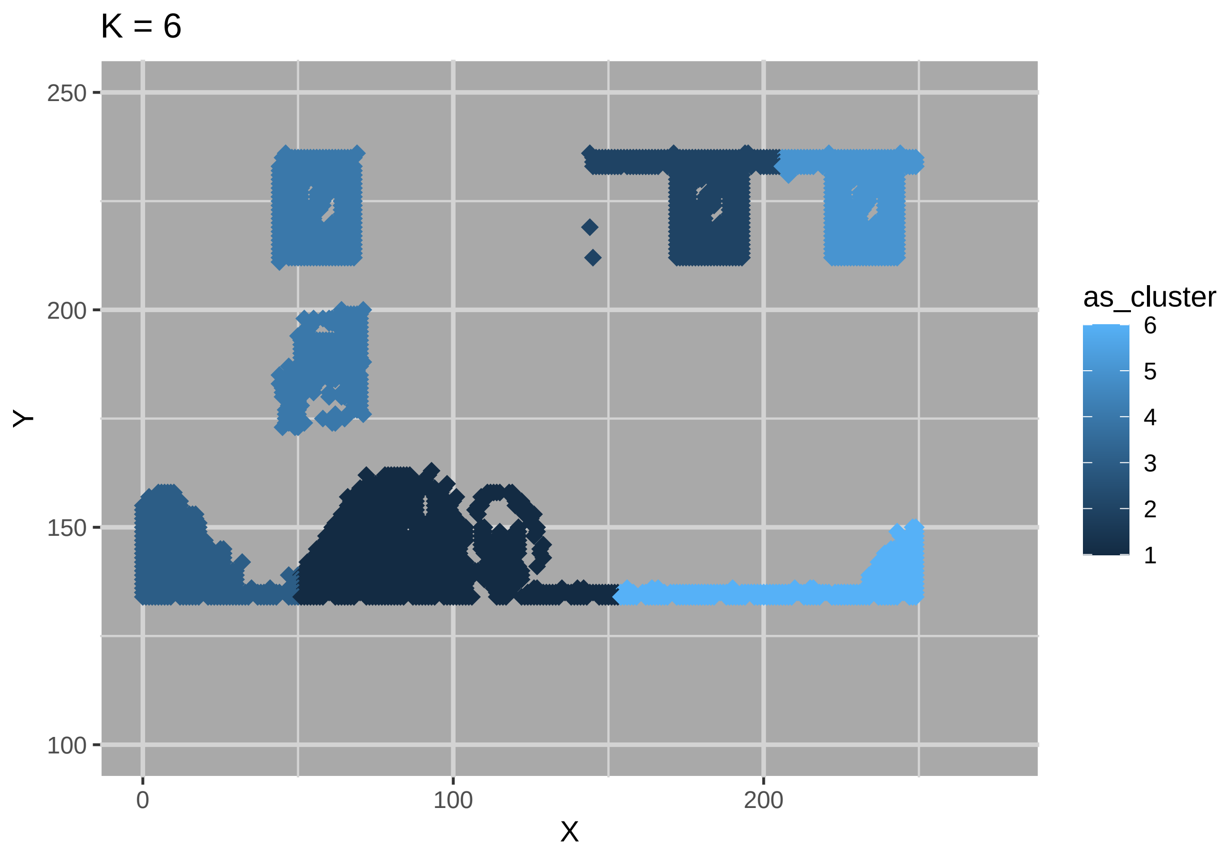
```r
# K=4
df4 <- cbind(km4$cluster, clust$x, clust$y)
colnames(df4)<- c("as_cluster", "x", "y")
p4 <- ggplot(df3, aes(x, y, color = as_cluster))
p4 + geom_point(shape = 18, size = 2.8) + theme(panel.grid =
  element_line(color = "lightgrey", linewidth = 0.8, linetype = 1),
  panel.background = element_rect(color = "white", fill = "darkgrey")) +
  labs(title = "K = 4", x ="X", y = "Y") + xlim(0, 275) + ylim(100, 250)
```



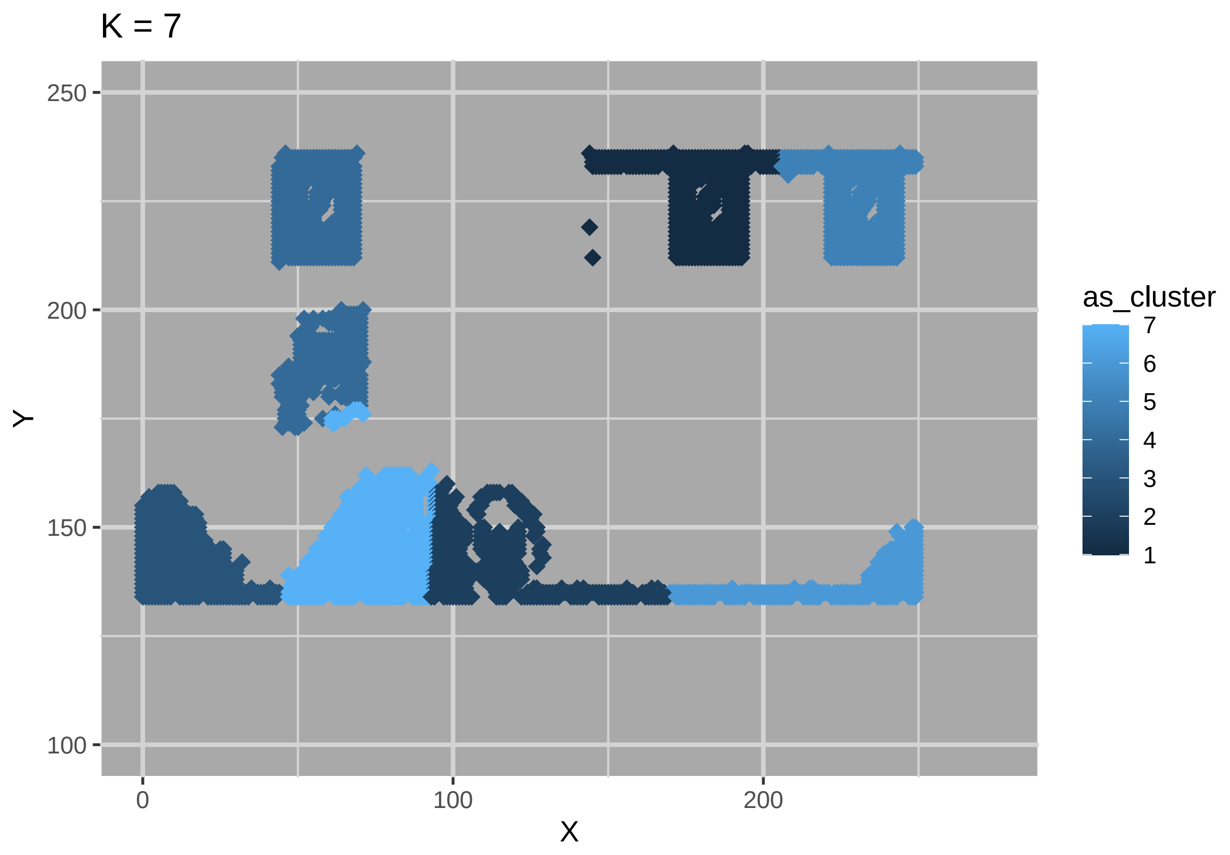
```r
# K=5
df5 <- cbind(km5$cluster, clust$x, clust$y)
colnames(df5)<- c("as_cluster", "x", "y")
p5 <- ggplot(df5, aes(x, y, color = as_cluster))
p5 + geom_point(shape = 18, size = 2.8) + theme(panel.grid =
  element_line(color = "lightgrey", linewidth = 0.8, linetype = 1),
  panel.background = element_rect(color = "white", fill = "darkgrey")) +
  labs(title = "K = 5", x ="X", y = "Y") + xlim(0, 275) + ylim(100, 250)
```



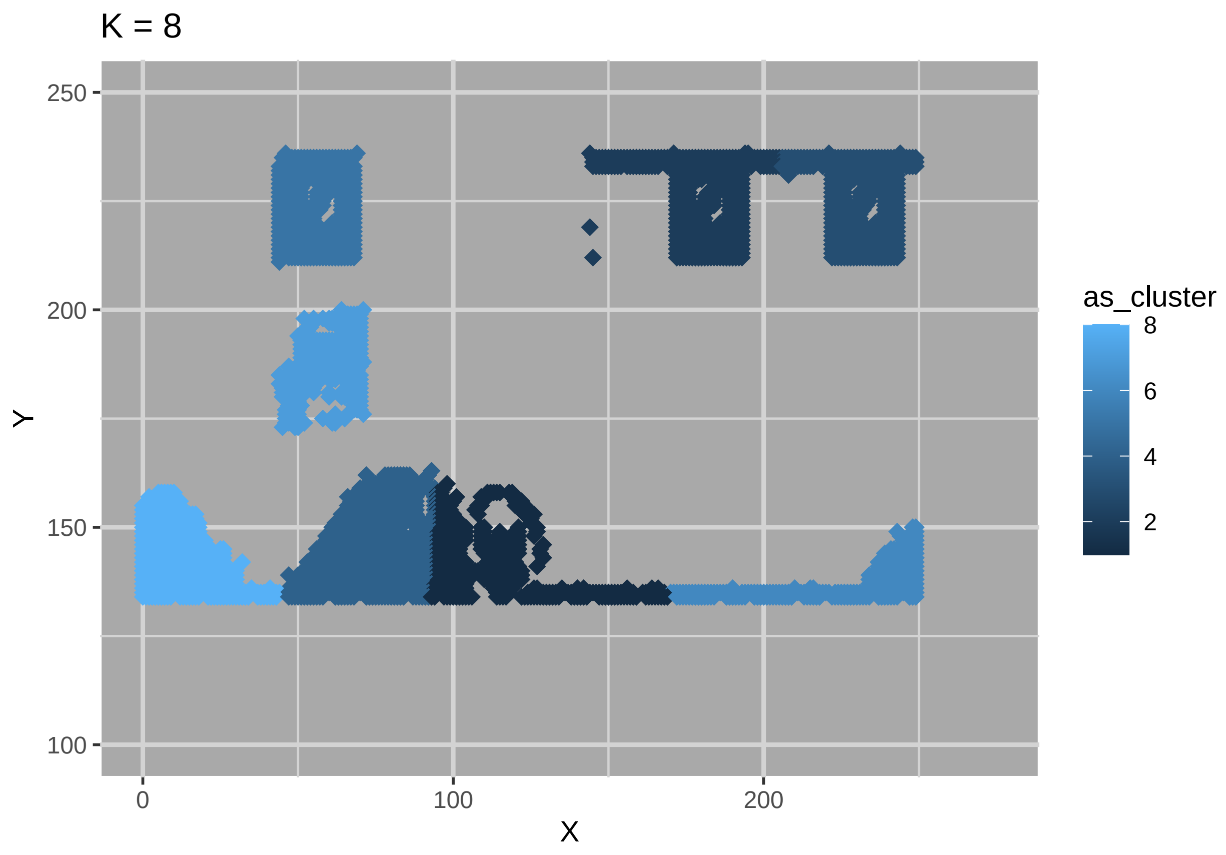
```
# K=6
df6 <- cbind(km6$cluster, clust$x, clust$y)
colnames(df6)<- c("as_cluster", "x", "y")
p6 <- ggplot(df6, aes(x, y, color = as_cluster))
p6 + geom_point(shape = 18, size = 2.8) + theme(panel.grid =
  element_line(color = "lightgrey", linewidth = 0.8, linetype = 1),
  panel.background = element_rect(color = "white", fill = "darkgrey")) +
  labs(title = "K = 6", x ="X", y = "Y") + xlim(0, 275) + ylim(100, 250)
```



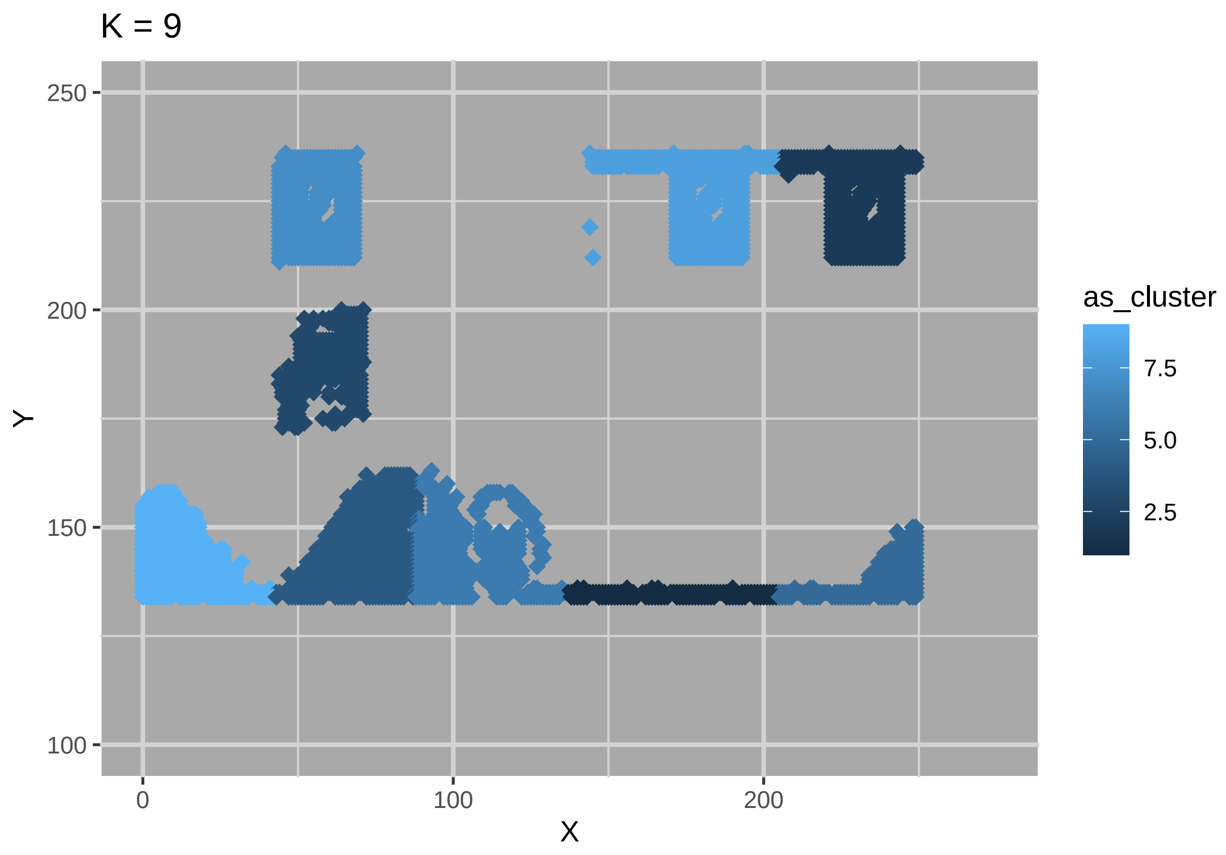
```r
# K=7
df7 <- cbind(km7$cluster, clust$x, clust$y)
colnames(df7)<- c("as_cluster", "x", "y")
p7 <- ggplot(df7, aes(x, y, color = as_cluster))
p7 + geom_point(shape = 18, size = 2.8) + theme(panel.grid =
  element_line(color = "lightgrey", linewidth = 0.8, linetype = 1),
  panel.background = element_rect(color = "white", fill = "darkgrey")) +
  labs(title = "K = 7", x ="X", y = "Y") + xlim(0, 275) + ylim(100, 250)
```



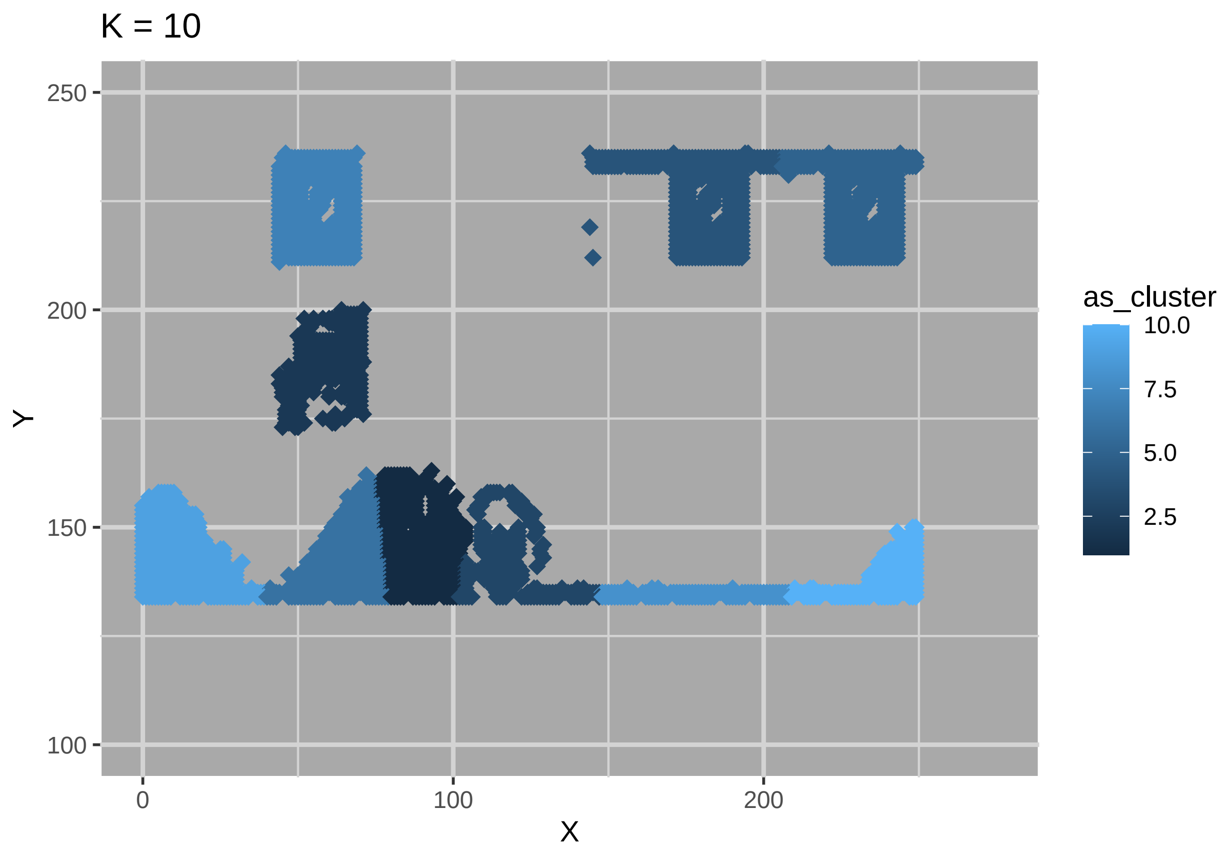
```
# K=8
df8 <- cbind(km8$cluster, clust$x, clust$y)
colnames(df8)<- c("as_cluster", "x", "y")
p8 <- ggplot(df8, aes(x, y, color = as_cluster))
p8 + geom_point(shape = 18, size = 2.8) + theme(panel.grid =
  element_line(color = "lightgrey", linewidth = 0.8, linetype = 1),
  panel.background = element_rect(color = "white", fill = "darkgrey")) +
  labs(title = "K = 8", x ="X", y = "Y") + xlim(0, 275) + ylim(100, 250)
```



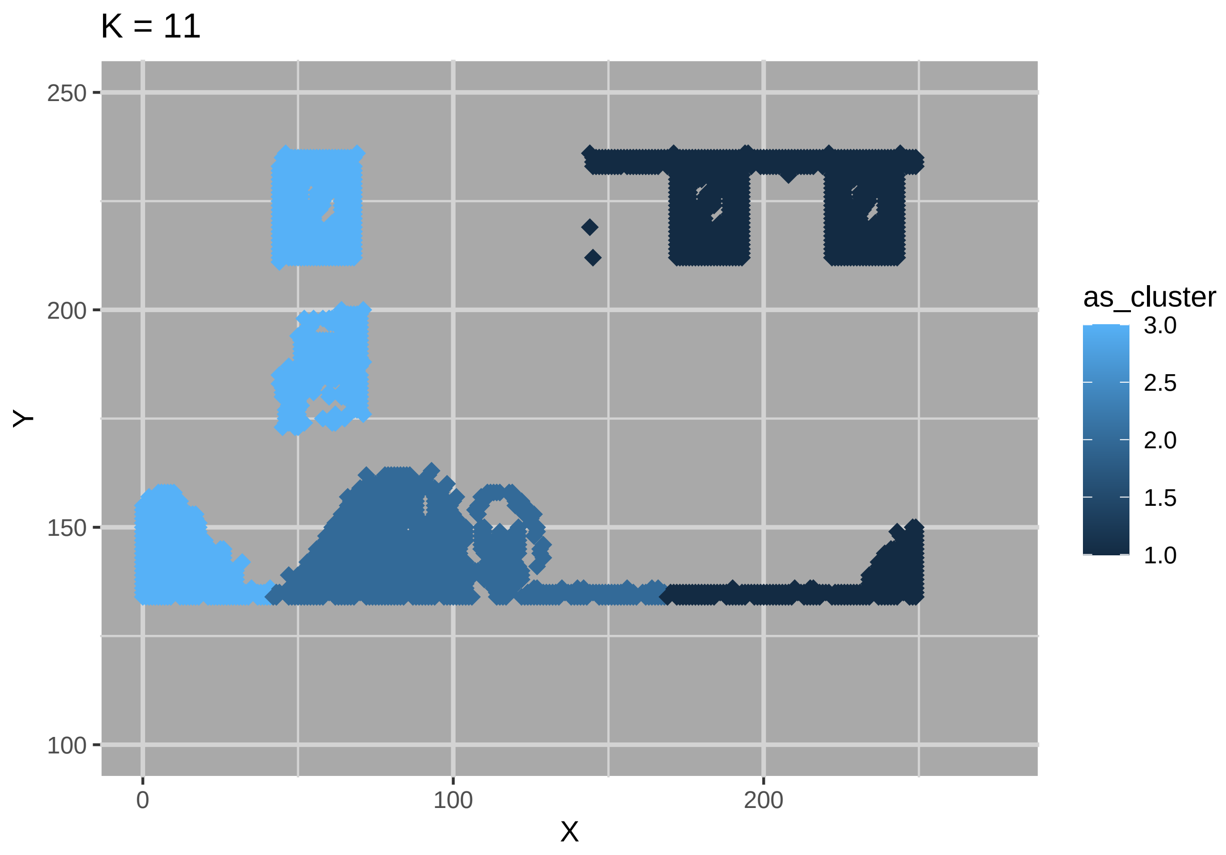
```
# K=9
df9 <- cbind(km9$cluster, clust$x, clust$y)
colnames(df9)<- c("as_cluster", "x", "y")
p9 <- ggplot(df9, aes(x, y, color = as_cluster))
p9 + geom_point(shape = 18, size = 2.8) + theme(panel.grid =
  element_line(color = "lightgrey", linewidth = 0.8, linetype = 1),
  panel.background = element_rect(color = "white", fill = "darkgrey")) +
  labs(title = "K = 9", x ="X", y = "Y") + xlim(0, 275) + ylim(100, 250)
```



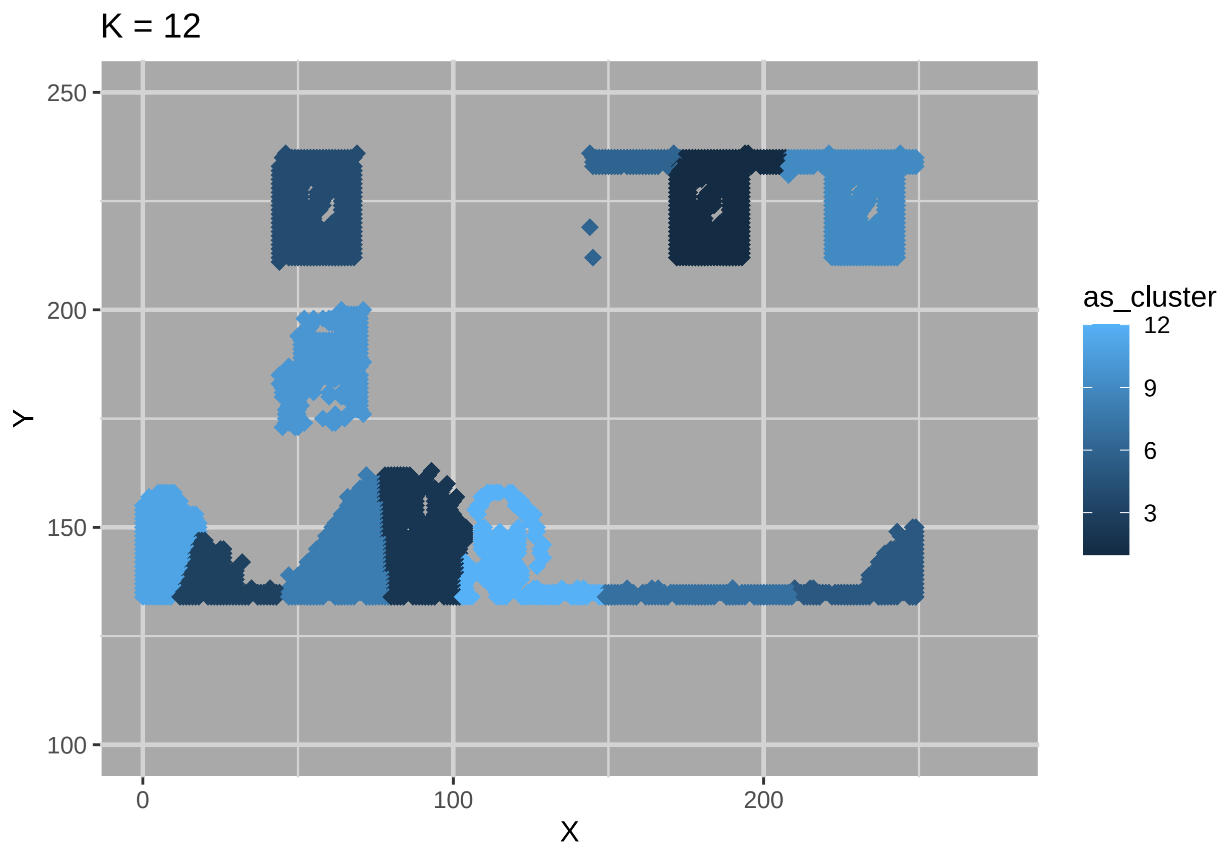
```r
# K=10
df10 <- cbind(km10$cluster, clust$x, clust$y)
colnames(df10)<- c("as_cluster", "x", "y")
p10 <- ggplot(df10, aes(x, y, color = as_cluster))
p10 + geom_point(shape = 18, size = 2.8) + theme(panel.grid =
  element_line(color = "lightgrey", linewidth = 0.8, linetype = 1),
  panel.background = element_rect(color = "white", fill = "darkgrey")) +
  labs(title = "K = 10", x ="X", y = "Y") + xlim(0, 275) + ylim(100, 250)
```



```
# K=11
df11 <- cbind(km11$cluster, clust$x, clust$y)
colnames(df11)<- c("as_cluster", "x", "y")
p11 <- ggplot(df3, aes(x, y, color = as_cluster))
p11 + geom_point(shape = 18, size = 2.8) + theme(panel.grid =
  element_line(color = "lightgrey", linewidth = 0.8, linetype = 1),
  panel.background = element_rect(color = "white", fill = "darkgrey")) +
  labs(title = "K = 11", x ="X", y = "Y") + xlim(0, 275) + ylim(100, 250)
```



```
# K=12
df12 <- cbind(km12$cluster, clust$x, clust$y)
colnames(df12)<- c("as_cluster", "x", "y")
p12 <- ggplot(df12, aes(x, y, color = as_cluster))
p12 + geom_point(shape = 18, size = 2.8) + theme(panel.grid =
  element_line(color = "lightgrey", linewidth = 0.8, linetype = 1),
  panel.background = element_rect(color = "white", fill = "darkgrey")) +
  labs(title = "K = 12", x ="X", y = "Y") + xlim(0, 275) + ylim(100, 250)
```

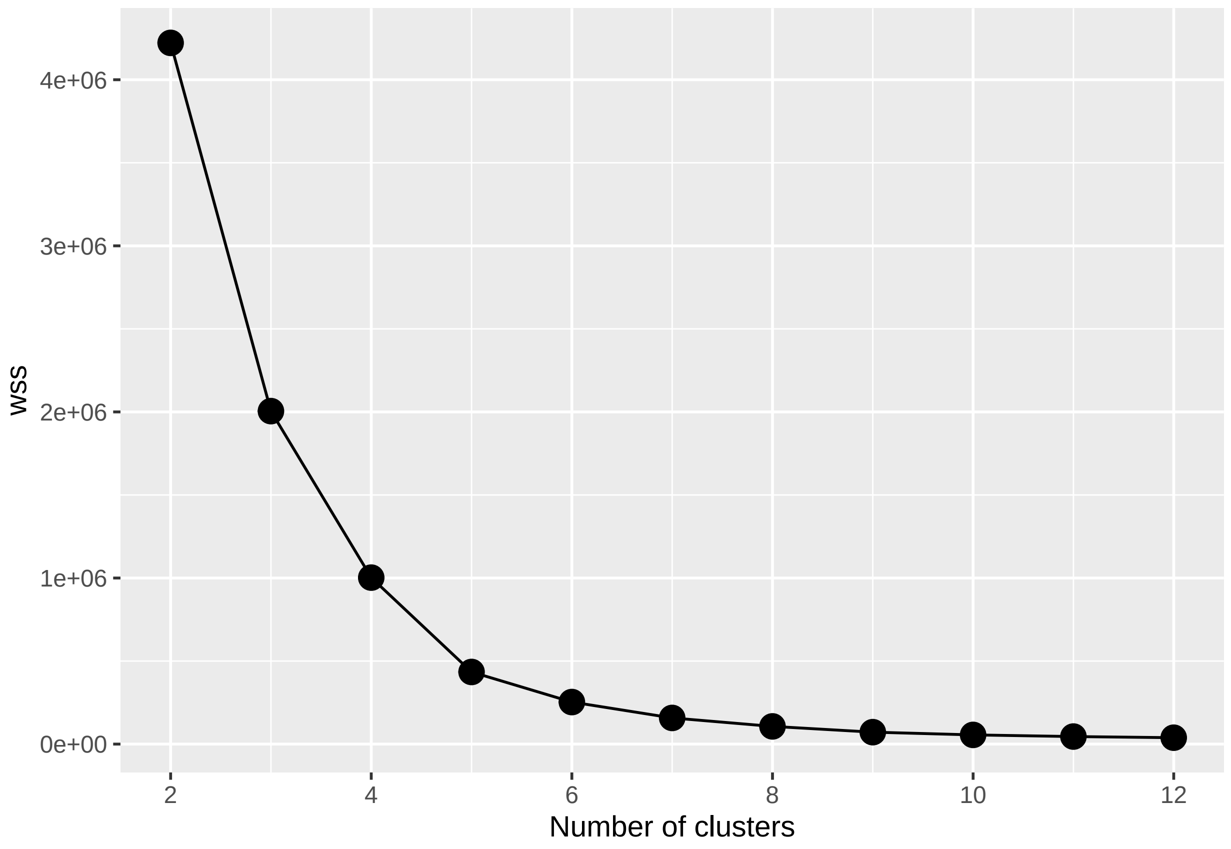


## Calculate Kmeans Averages and Plot Them

```
# This could again be made cleaner with a for loop.
# Calculating the Averages
wss2 <- km2$tot.withinss/2
wss3 <- km3$tot.withinss/3
wss4 <- km4$tot.withinss/4
wss5 <- km5$tot.withinss/5
wss6 <- km6$tot.withinss/6
wss7 <- km7$tot.withinss/7
wss8 <- km8$tot.withinss/8
wss9 <- km9$tot.withinss/9
wss10 <- km10$tot.withinss/10
wss11 <- km11$tot.withinss/11
wss12 <- km12$tot.withinss/12

# Make Kmeans Table and Plot
km <- list(2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12)
km <- as.numeric(km)
wss <- list(wss2, wss3, wss4, wss5, wss6, wss7,
            wss8, wss9, wss10, wss11, wss12)
wss <- as.numeric(wss)
wss_df <- tibble(km, wss)
wss_df
```

```
## # A tibble: 11 x 2
##       km     wss
##     <dbl>     <dbl>
##  1      2 4221841.
##  2      3 2004793.
##  3      4 1002420.
##  4      5  434323.
##  5      6  253174.
##  6      7  157553.
##  7      8  106645.
##  8      9   71926.
##  9     10   55472.
## 10     11   45458.
## 11     12   38550.
```

```
dist_plot <- ggplot(wss_df, aes(x = km, 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')
dist_plot
```



I'd say the "right" number of clusters would be five based on the elbow of the plot.