Assignment 9: K-Means

DA 410

Marjorie Blanco

1. Which part does really catch your attention?

Two aspect that caught my attention are model assumptions and choosing K.

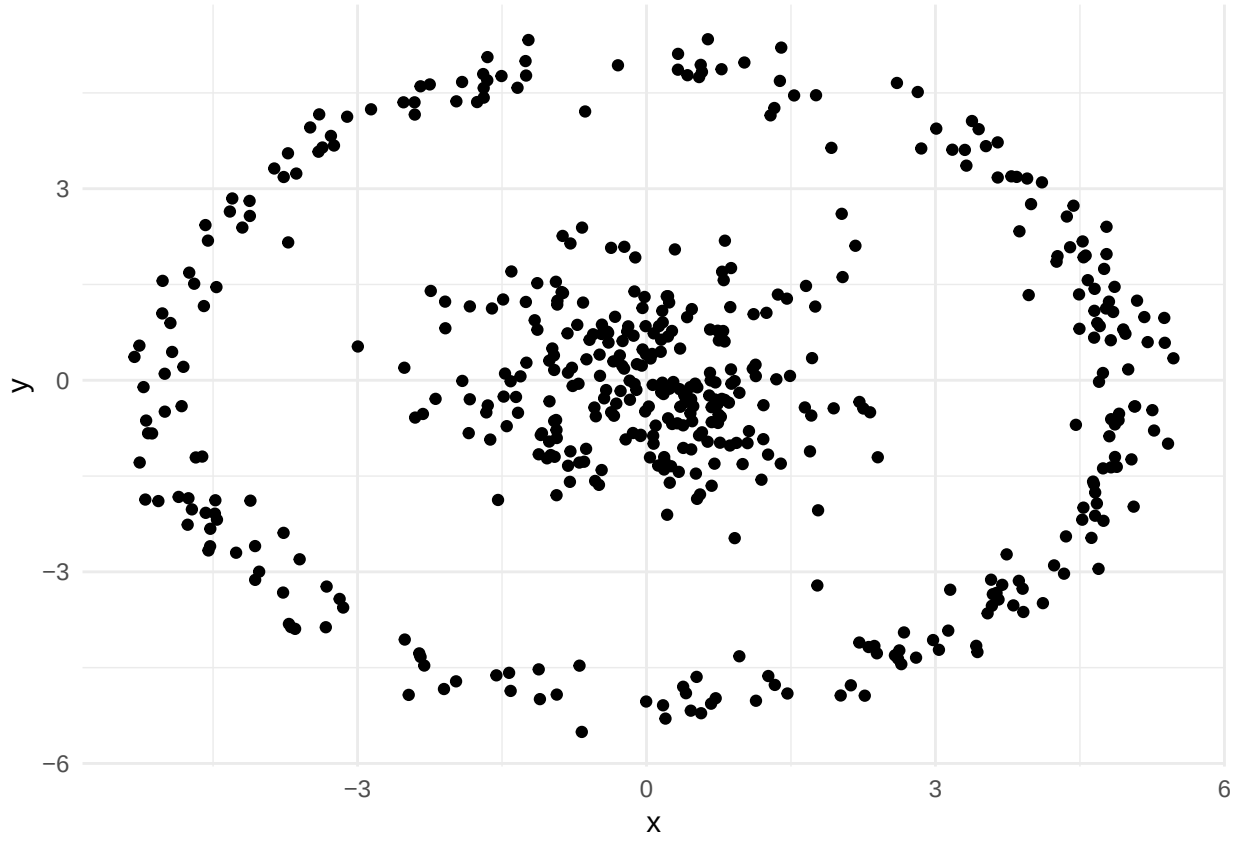
2. When you try to use it, what kind of concerns you may have?

While K means is a simple algorithm it does come with assumptions/limitations that if not meet can make K means impractical.

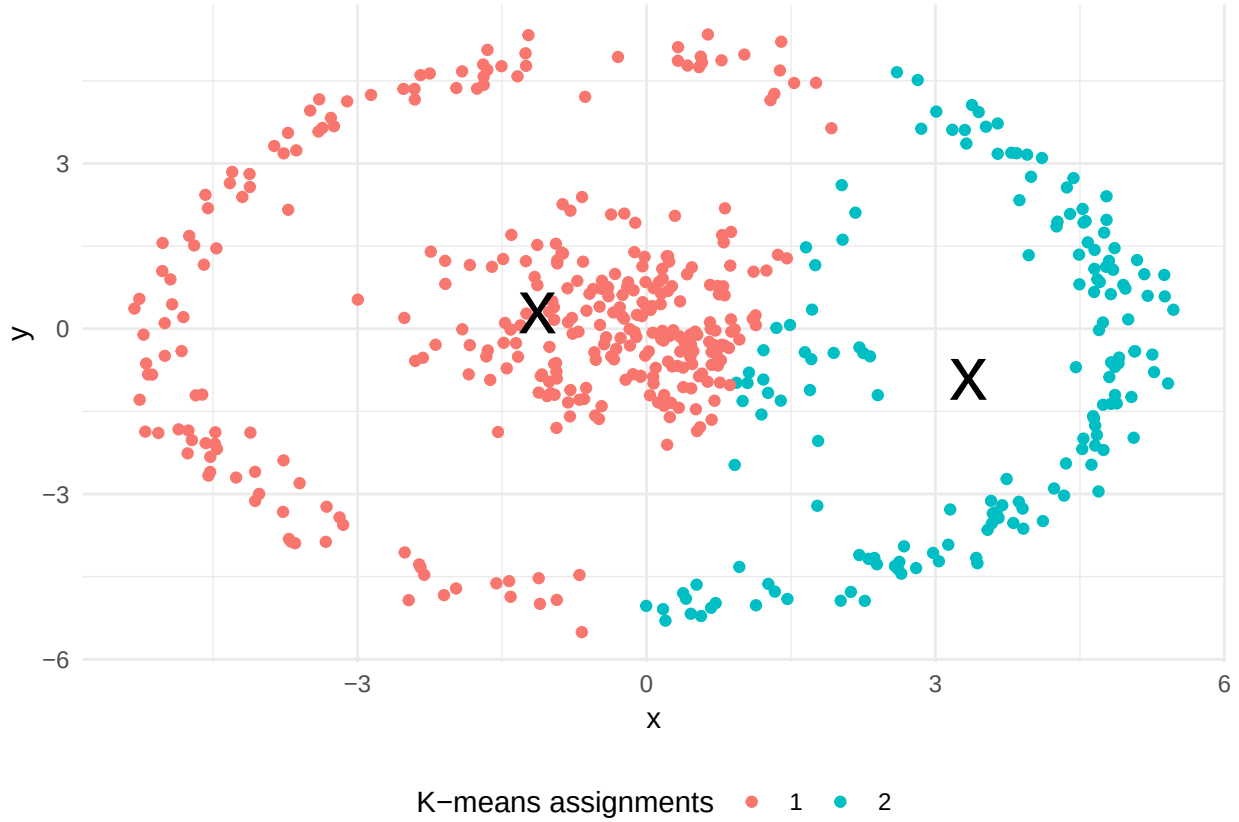
- Assumes the variance of the distribution of each attribute (variable) is spherical
- Assumes all variables have the same variance
- Assumes the prior probability for all k clusters is the same (each cluster has roughly equal number of observations)

Broken Assumption: Non-Spherical Data

```
set.seed(1)
n <- 250
c1 <- tibble(x = rnorm(n), y = rnorm(n))
c2 <- tibble(r = rnorm(n, 5, .25), theta = runif(n, 0, 2 * pi),
             x = r * cos(theta), y = r * sin(theta)) %>%
  dplyr::select(x, y)

c1 <- tibble(x = rnorm(n), y = rnorm(n), cluster = 1)
c2 <- tibble(r = rnorm(n, 5, .25), theta = runif(n, 0, 2 * pi),
                 x = r * cos(theta), y = r * sin(theta), cluster = 2) %>%
    dplyr::select(x, y, cluster)
points1 <- rbind(c1, c2) %>% mutate(cluster = factor(cluster))

points1 <- rbind(c1, c2)
ggplot(points1, aes(x, y)) +
  geom_point() +
  theme_minimal()
```

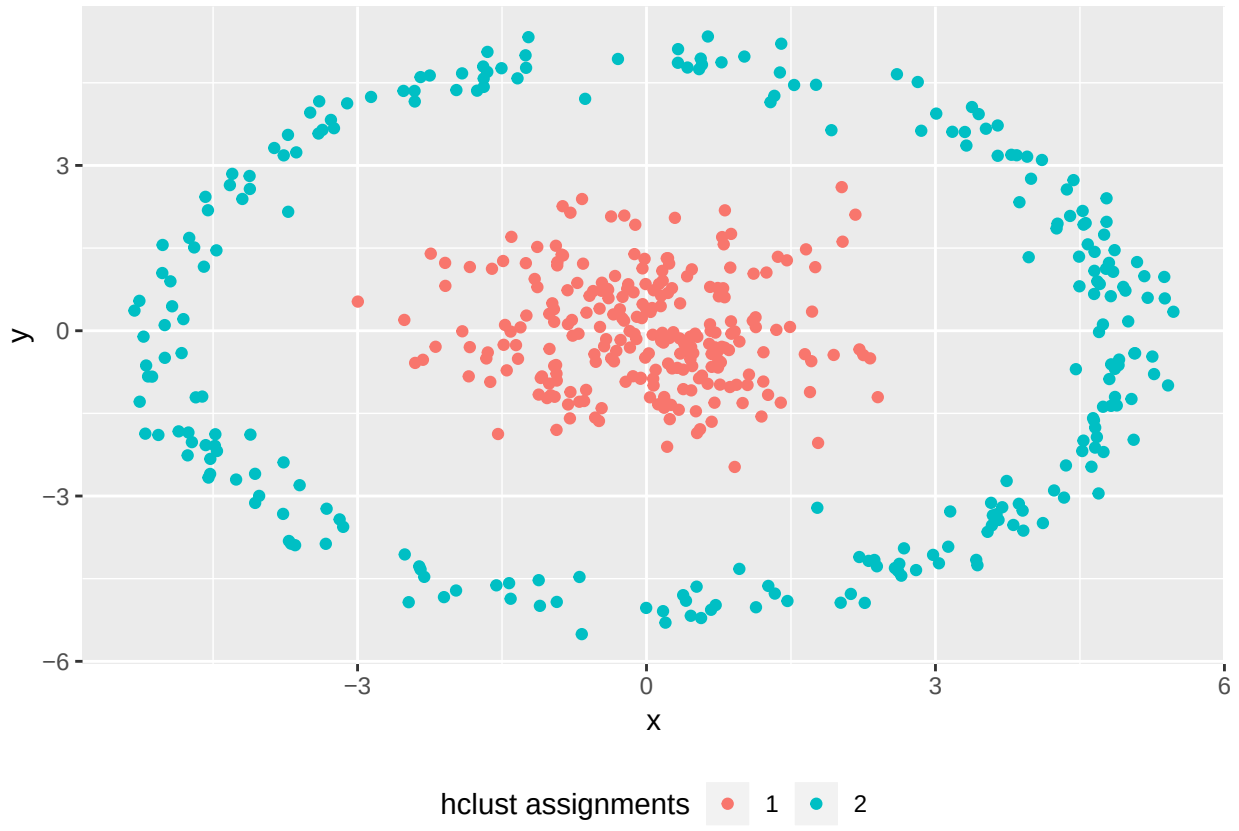



```
clust <- kmeans(points1 %>% select(x,y), 2)
points1$cluster <- clust$cluster

ggplot(points1, aes(x, y)) +
  geom_point(aes(color = factor(cluster)))  +
  geom_point(aes(x1, x2), data = tidy(clust), size = 10, shape = "x") +
  labs(color = "K-means assignments") +
  theme_minimal() +
  theme(legend.position="bottom")
```



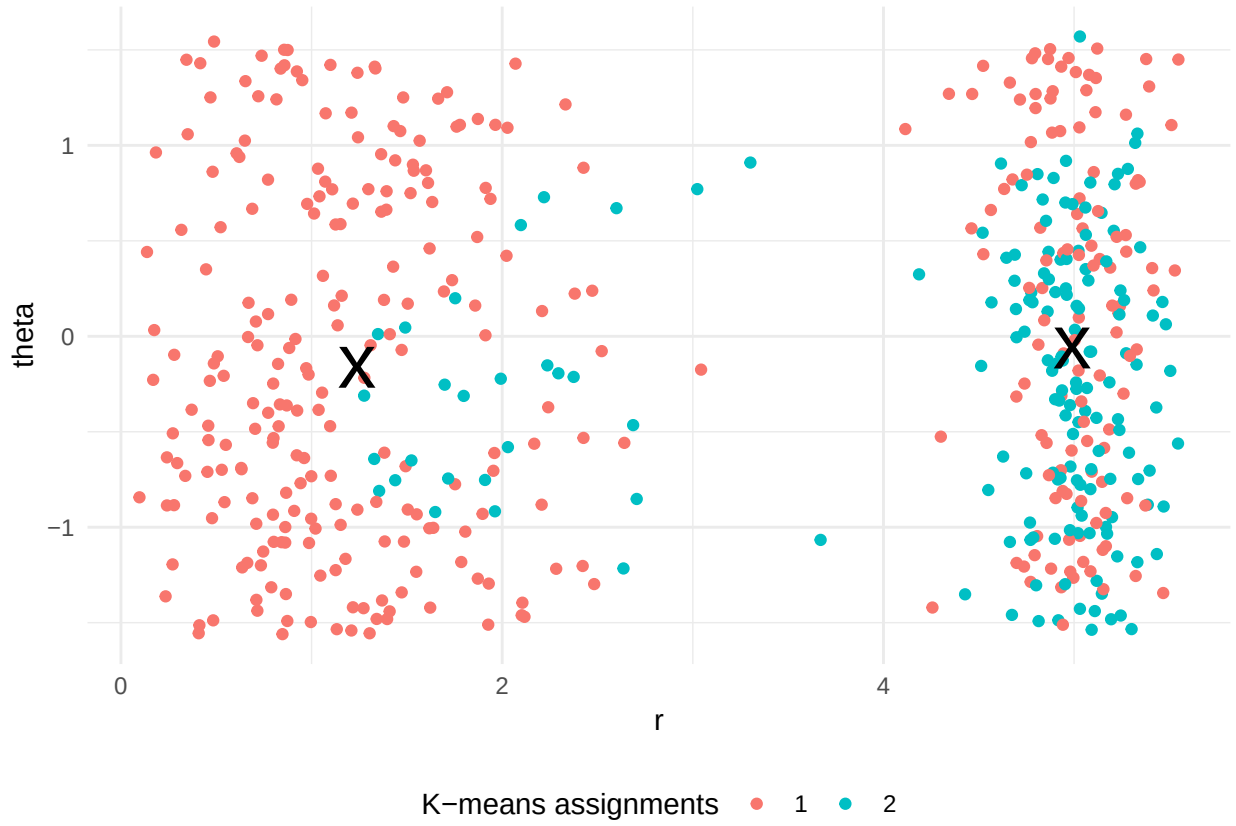
This issue can be solved by using a single linkage hierachical clustering instead.

```
points1$hclust_assignments <- points1 %>% dplyr::select(x, y) %>%
  dist() %>% hclust(method = "single") %>%
  cutree(2) %>% factor()

ggplot(points1, aes(x, y, color = hclust_assignments)) +
  geom_point() +
  labs(color = "hclust assignments") +
  theme(legend.position="bottom")
```

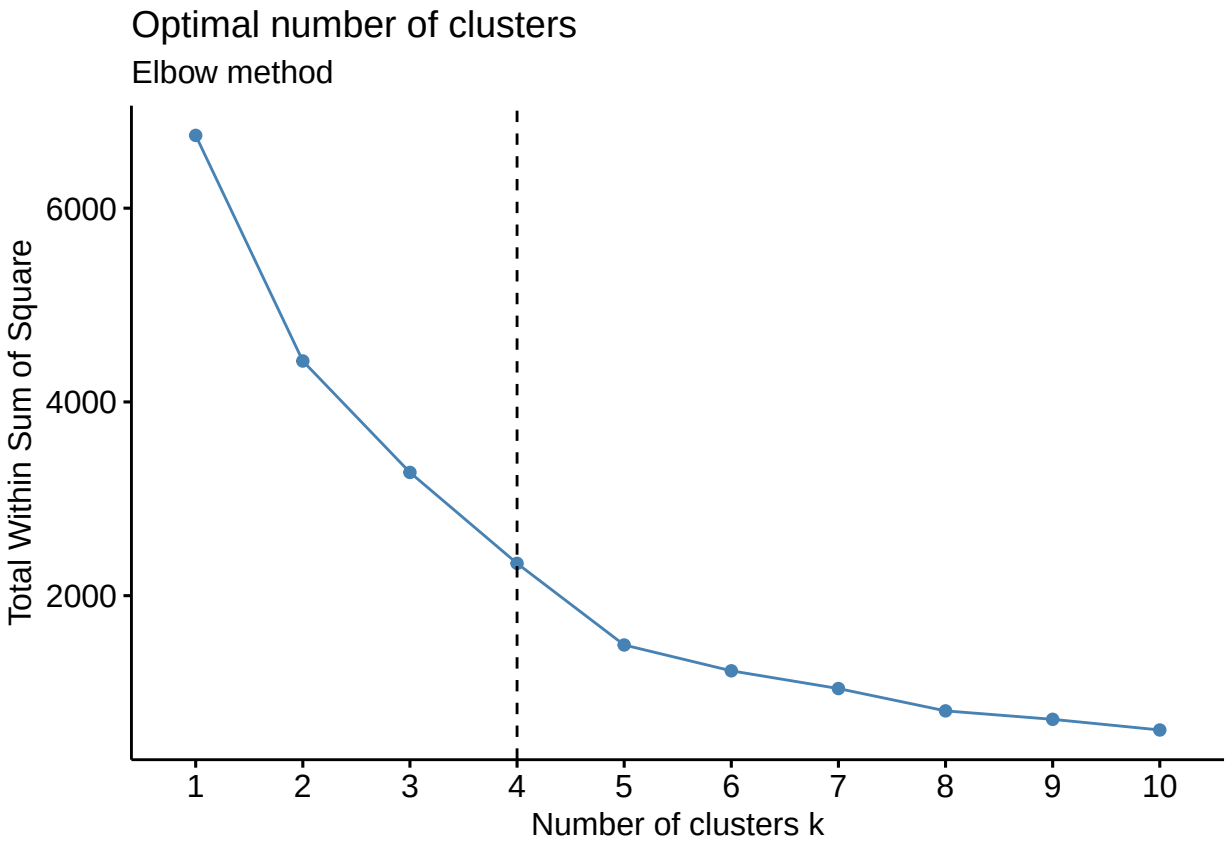


Another way to solve this problem is to transformu the data into polar coordinates.

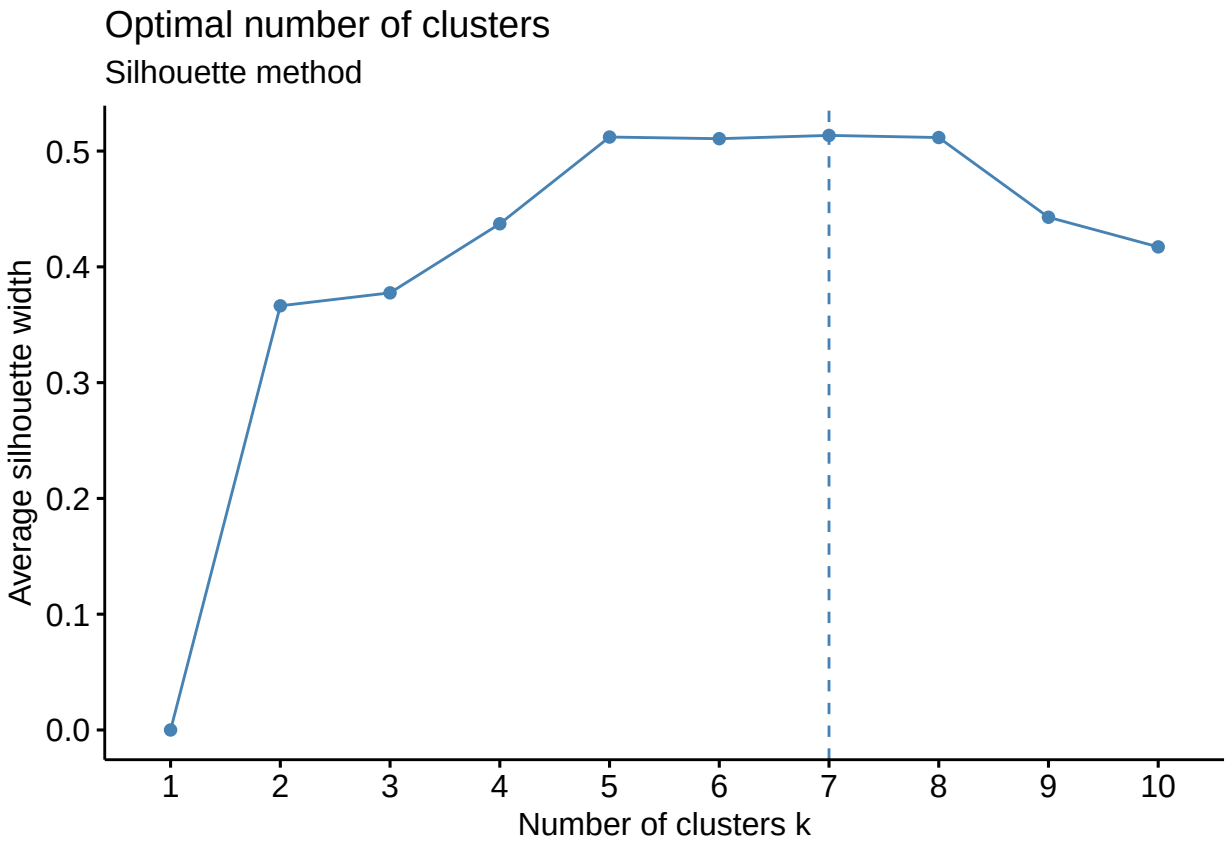
```
points1_polar <- points1 %>% transform(r = sqrt(x^2 + y^2), theta = atan(y / x))

points1_polar$cluster <- clust$cluster

clust <- points1_polar %>% ungroup %>% dplyr::select(r, theta) %>% kmeans(2)
ggplot(points1_polar, aes(r, theta)) +
  geom_point(aes(color = factor(cluster))) +
  geom_point(aes(x1, x2), data = tidy(clust), size = 10, shape = "x") +
  labs(color = "K-means assignments") +
  theme_minimal() +
  theme(legend.position="bottom")
```



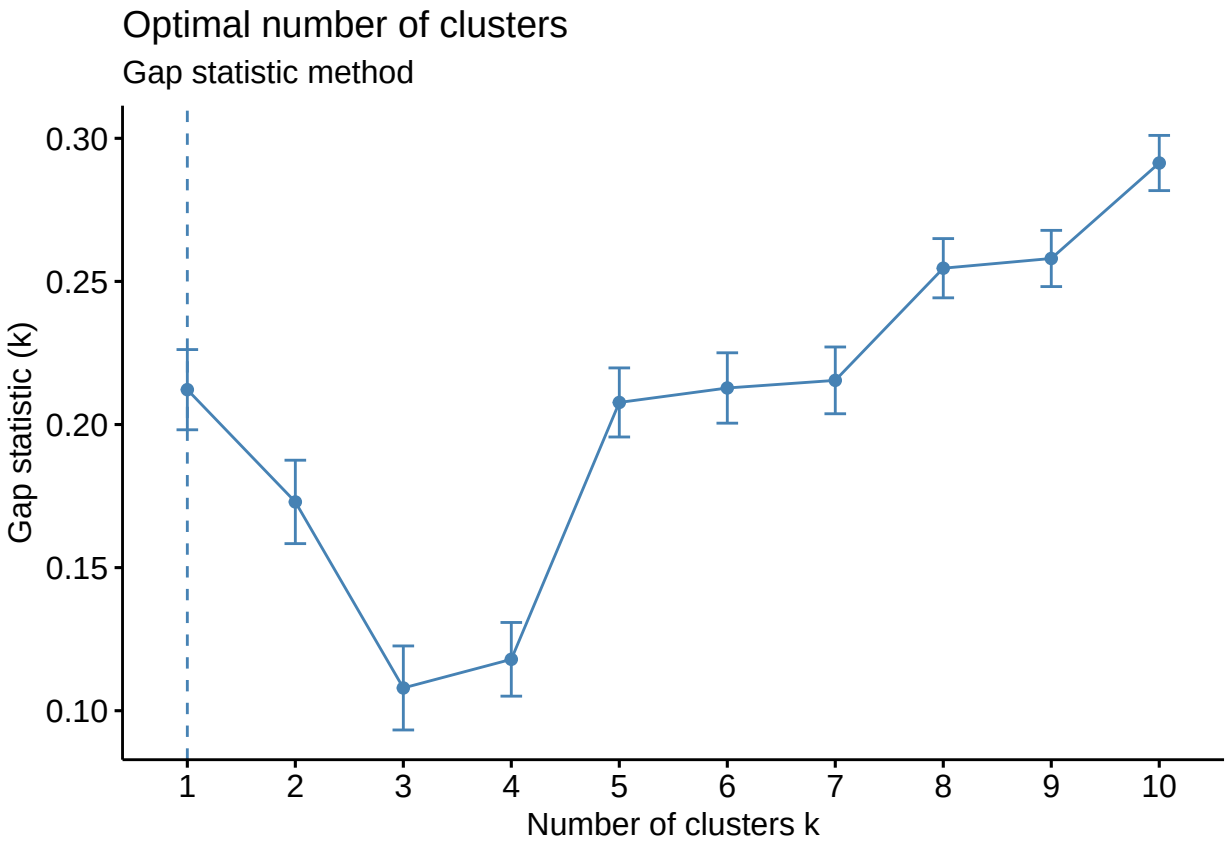

```
points1 <- points1 %>% select(x,y)

# Elbow method
fviz_nbclust(points1, kmeans, method = "wss") +
    geom_vline(xintercept = 4, linetype = 2)+
  labs(subtitle = "Elbow method")
```




```
# Silhouette method
fviz_nbclust(points1, kmeans, method = "silhouette")+
  labs(subtitle = "Silhouette method")
```




```
# Gap statistic
fviz_nbclust(points1, kmeans, nstart = 25,  method = "gap_stat", nboot = 50) +
  labs(subtitle = "Gap statistic method")
```



Elbow method: 4 clusters solution suggested

Silhouette method: 7 clusters solution suggested

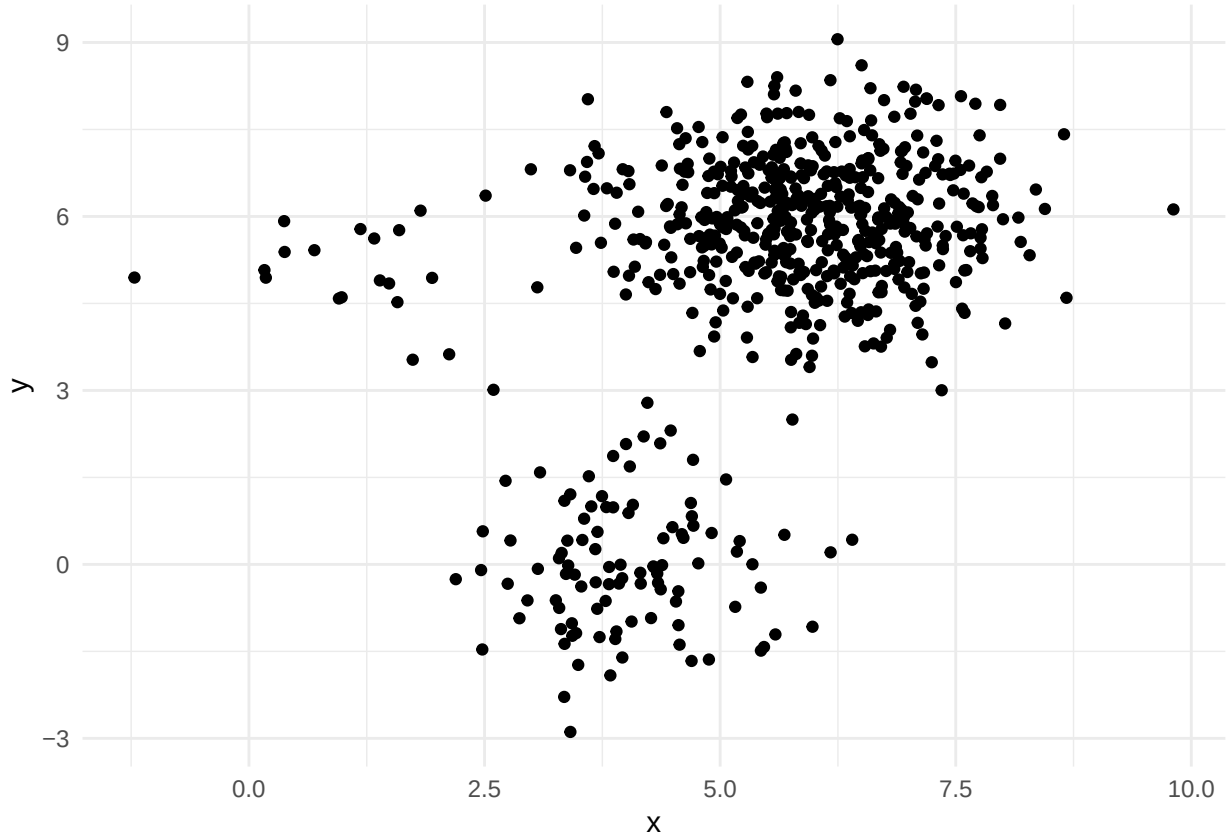
Gap statistic method: 1 clusters solution suggested

Unevenly sized clusters

In this case 3 cluster of sizes 20, 100, 500 was generated from a multivariate Gaussian distribution.

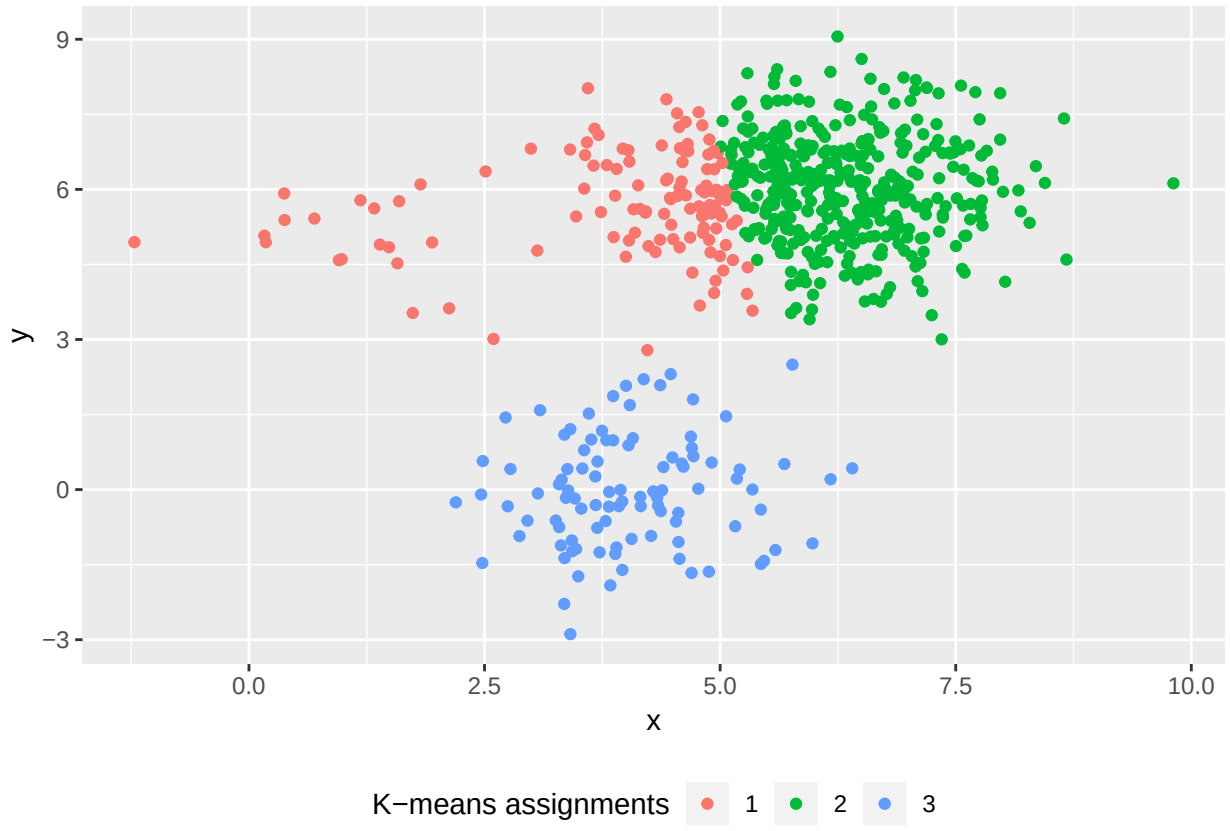
```
sizes <- c(20, 100, 500)
```

```
set.seed(1)

centers <- data.frame(x = c(1, 4, 6), y = c(5, 0, 6), n = sizes,
                      cluster = factor(1:3))
points <- centers %>% group_by(cluster) %>%
  do(tibble(x = rnorm(.$n, .$x), y = rnorm(.$n, .$y)))

ggplot(points, aes(x, y)) +
  geom_point() +
  theme_minimal()
```

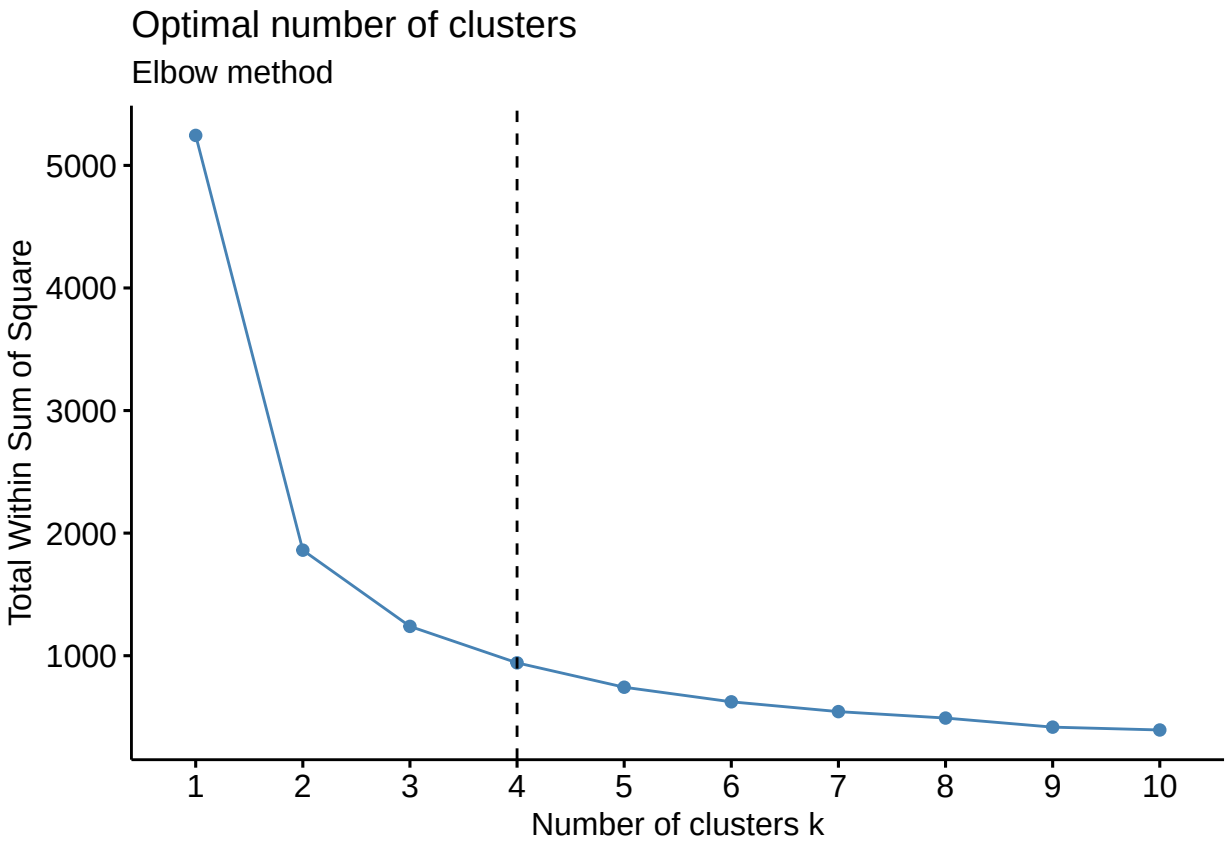


The k-means algorithm gives more "weight" to larger clusters as part of goal to minimize the within-cluster sum of squares.

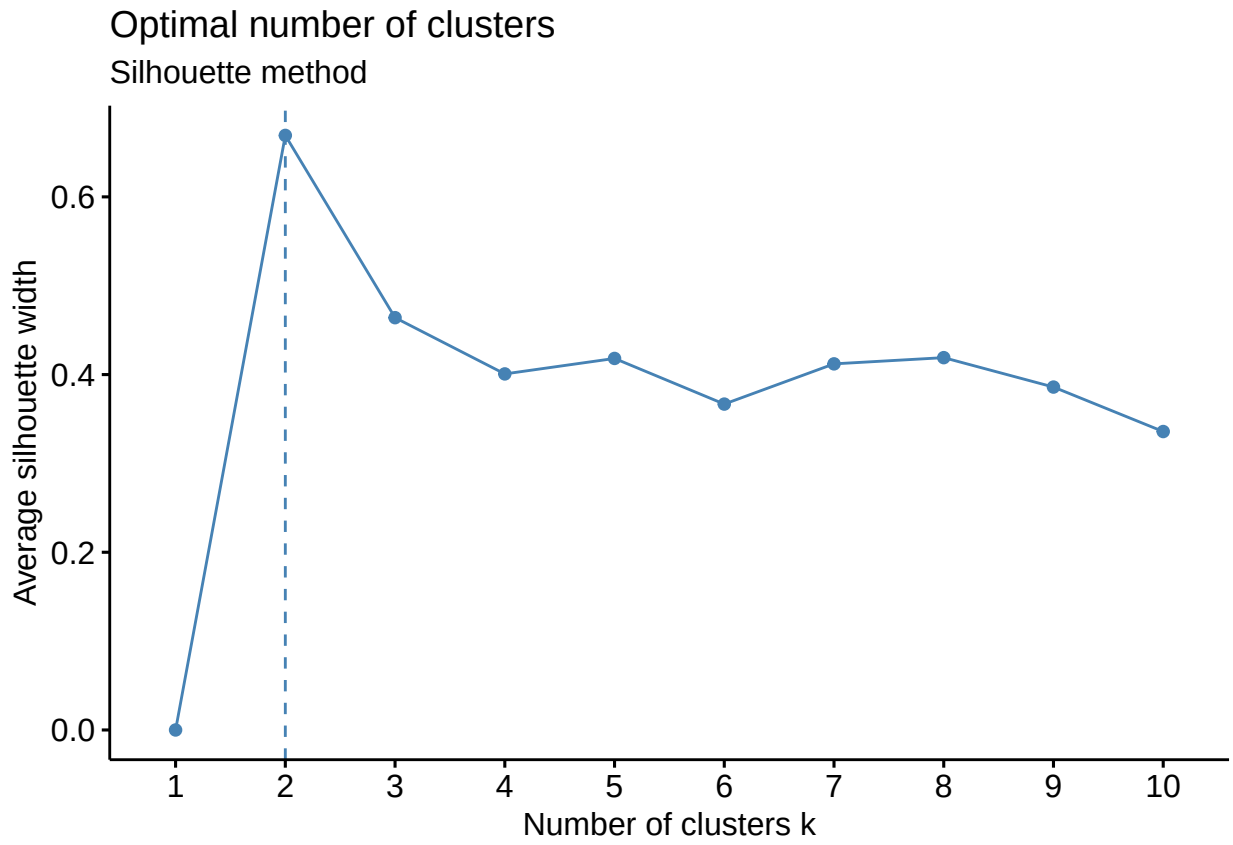
```
clust <- kmeans(points %>% select(x,y), 3)
points$cluster <- clust$cluster

ggplot(points, aes(x, y)) +
  geom_point(aes(color = factor(cluster)))  +
  labs(color = "K-means assignments")  +
  theme(legend.position="bottom")
```



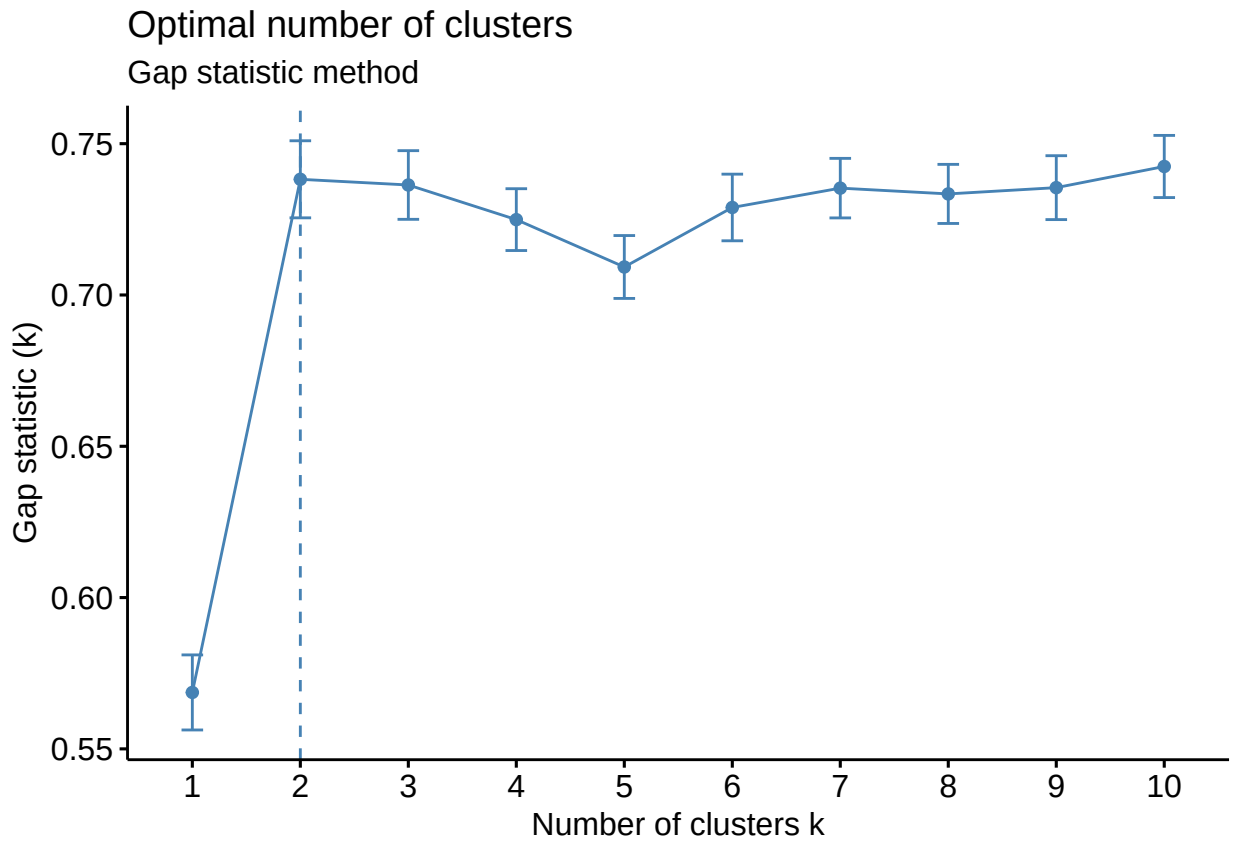

```
points <- points %>% select(x,y)

# Elbow method
fviz_nbclust(points, kmeans, method = "wss") +
    geom_vline(xintercept = 4, linetype = 2)+
  labs(subtitle = "Elbow method")
```




```
# Silhouette method
fviz_nbclust(points, kmeans, method = "silhouette")+
  labs(subtitle = "Silhouette method")
```




```
# Gap statistic
fviz_nbclust(points, kmeans, nstart = 25,  method = "gap_stat", nboot = 50) +
  labs(subtitle = "Gap statistic method")
```



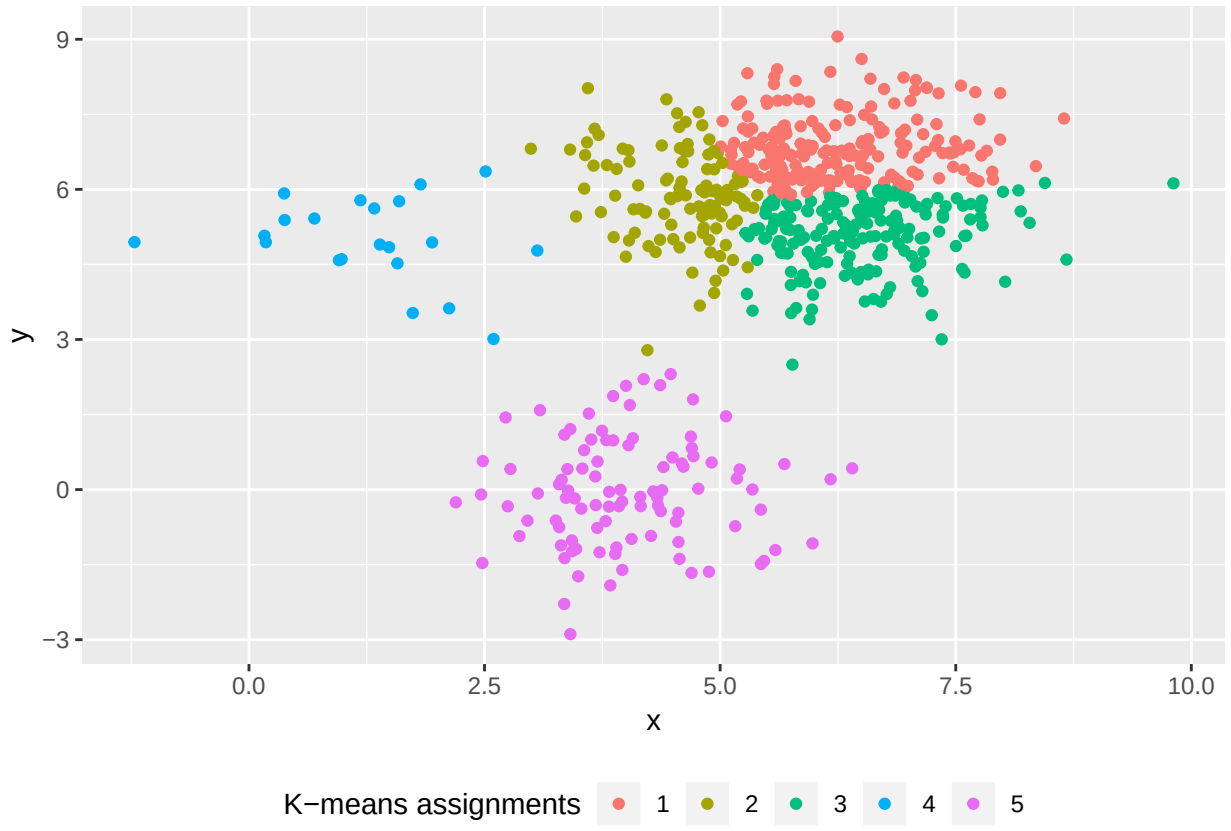
Elbow method: 4 clusters solution suggested

Silhouette method: 2 clusters solution suggested

Gap statistic method: 2 clusters solution suggested

A way to solve this problem is to increase K.

```
clust <- kmeans(points %>% select(x,y), 5)
points$cluster <- clust$cluster

ggplot(points, aes(x, y)) +
  geom_point(aes(color = factor(cluster)))  +
  labs(color = "K-means assignments")  +
  theme(legend.position="bottom")
```



3. Do you like it? Why or why not?

Like many of the modeling methods we learned in class K-means have advantages, disadvantage and limitations. K-means does have different methods (elbow, the silhouette and the gap statistic methods) for choosing the optimal number of clusters in a data set. One thing to keep in mind is assumptions is where the power comes from. Just accepting the drawbacks is not enough. It is necessary to know them to make an informed choices. Understand the limitations so that the algorithm can be tweaked. Often one simple solution is to transform the data to solve some of the drawbacks. In order for the model be right it must first be wrong.