JTG_10

Jeff Gould

11/3/2020

1) TSK 7.17

One dimensional points:

```
points <- c(6,12,18,24,30,42,48)
```

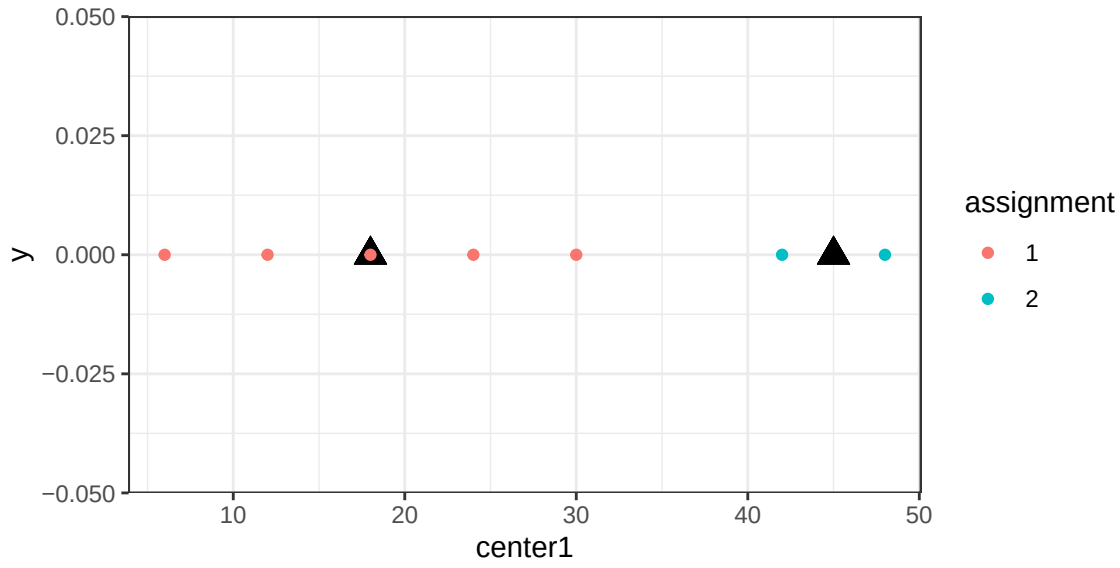
a) For each of the following sets of initial centroids, create two clusters by assigning each point to the nearest centroid, and then calculate the total squared error for each set of two clusters. Show both the clusters and the total squared error for each set of cnentroids

```
centroid1 <- c(18, 45)

results1 <- data.frame(points = points,
                       center1 = centroid1[1],
                       center2 = centroid1[2]) %>%
  mutate(distance1 = abs(points - center1),
         distance2 = abs(points - center2)) %>%
  mutate(assignment = case_when(
    distance1 <= distance2 ~ "1",
    distance2 < distance1 ~ "2"
  )) %>%
  mutate(error = case_when(
    assignment == "1" ~ distance1,
    assignment == "2" ~ distance2
  ))

TSE1 <- sum(results1$error^2)


ggplot(data = results1) +
  geom_point(aes(x = center1, y = 0), color = "black", shape = 17, size = 4) +
  geom_point(aes(x = center2, y = 0), color = "black", shape = 17, size = 4) +
  geom_point(aes(x = points, y = 0, color = assignment)) +
  theme_bw()
```




```
TSE1
```

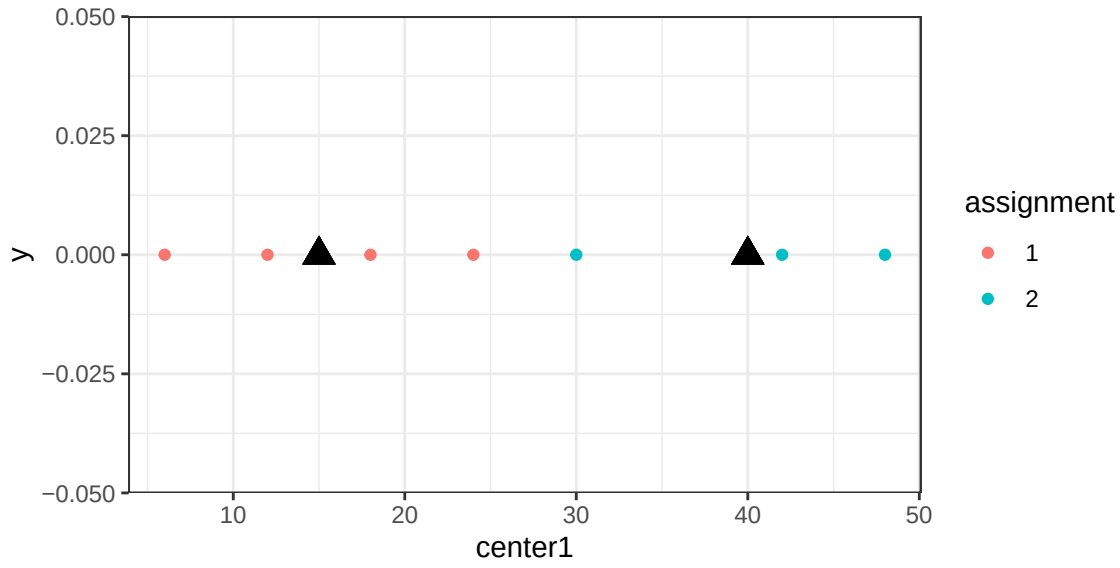
```
## [1] 378
```

```
centroid2 <- c(15, 40)

results2 <- data.frame(points = points,
                       center1 = centroid2[1],
                       center2 = centroid2[2]) %>%
  mutate(distance1 = abs(points - center1),
         distance2 = abs(points - center2)) %>%
  mutate(assignment = case_when(
    distance1 <= distance2 ~ "1",
    distance2 < distance1 ~ "2"
  )) %>%
  mutate(error = case_when(
    assignment == "1" ~ distance1,
    assignment == "2" ~ distance2
  ))

TSE2 <- sum(results2$error^2)


ggplot(data = results2) +
  geom_point(aes(x = center1, y = 0), color = "black", shape = 17, size = 4) +
  geom_point(aes(x = center2, y = 0), color = "black", shape = 17, size = 4) +
  geom_point(aes(x = points, y = 0, color = assignment)) +
  theme_bw()
```




```
TSE2
```

```
## [1] 348
```

b) Do both sets of centroids represent stable solutions?

Running the K-Means algorithm using each centroid as a starting point, we find that they arrive at the same ending centroids and have the same assignments. In fact, we actually find that the center is the same as centroid 2, $(15, 40)$. This also means that the initial assignments from centroid 2 above are the same, while 1 point that is initially assigned to centroid 1 gets moved to centroid 2 during the algorithm.

```
kmeans1 <- kmeans(points, centers = centroid1)
kmeans2 <- kmeans(points, centers = centroid2)

kmeans1$centers == kmeans2$centers
```

```
##   [,1]
## 1 TRUE
## 2 TRUE
```

```
kmeans2$cluster == kmeans1$cluster
```

```
## [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

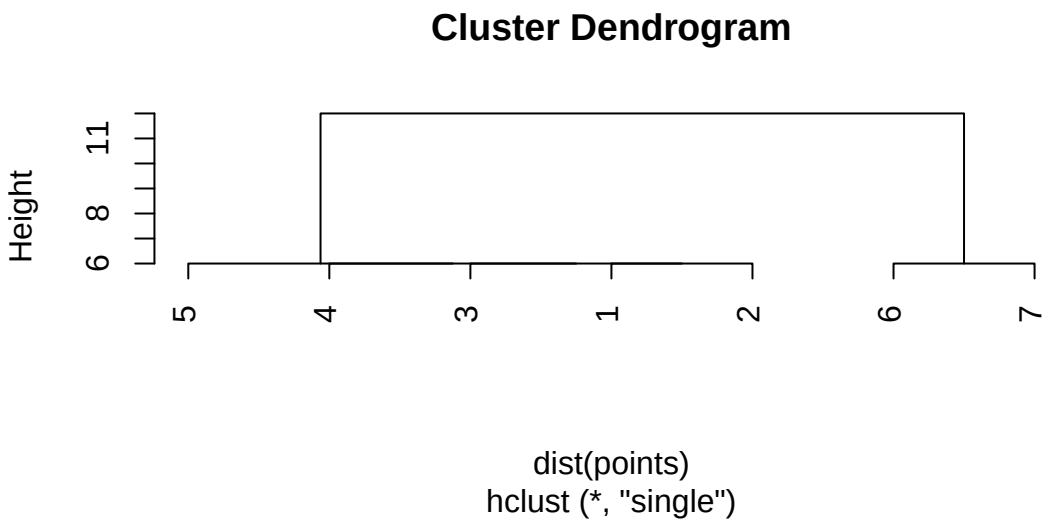
```
kmeans1$centers
```

```
##   [,1]
## 1   15
## 2   40
```

c)

The clusters produced using single-link are $(5, 4, 3, 1, 2)$ and $(6, 7)$

```
sing_link <- hclust(d = dist(points), method = "single")
plot(sing_link)
```



d) Which technique, K-mean or single link, seems to produce the "most natural" clustering in the situation?

Single-link appears to produce a more natural clustering. The driving reason for this stems from the point located at 30. This point is 6 units away from point 24, which is in cluster 1 for both K-Means and single link, and 12 units away from point 42, which is in cluster 2. In order to minimize the TSE in K-means, this point is grouped in the second cluster. But it's closest neighbor is in cluster 1, and so it gets grouped there in single link. And thus single link looks like it creates a "more natural" clustering

e) What definition(s) of clustering does this natural clustering correspond to? (Well-separated, center-based, contiguous, or density)

The "natural clustering" corresponds to contiguous clustering, as each point in a group is closer to another point in that group than it is to a point in another group. Ie, each point's nearest neighbor is in the same cluster.

f) What well-known characteristic of the K-means algorithm explains the previous behavior?

K-means clustering struggles with data that is not well-separated or non-globular, which this data is not. Because K-means is focused on ninimizing the Total Squared Error, it doesn't care about necessarily keeping closer points grouped together, which we see when the point at 30 is pushed to the second cluster.

2

```
meats <- foreign::read.arff("Meats.arff")

set.seed(123)
meatsKMeans <- kmeans(x = meats[,2:6], centers = 2)

meatsKMeans$centers
```

```
##    calories   protein       fat    calcium      iron
## 1 0.7629631 0.6315791 0.6988303 0.01043611 0.3575757
## 2 0.2681481 0.6315789 0.1432748 0.15623083 0.3343434
```

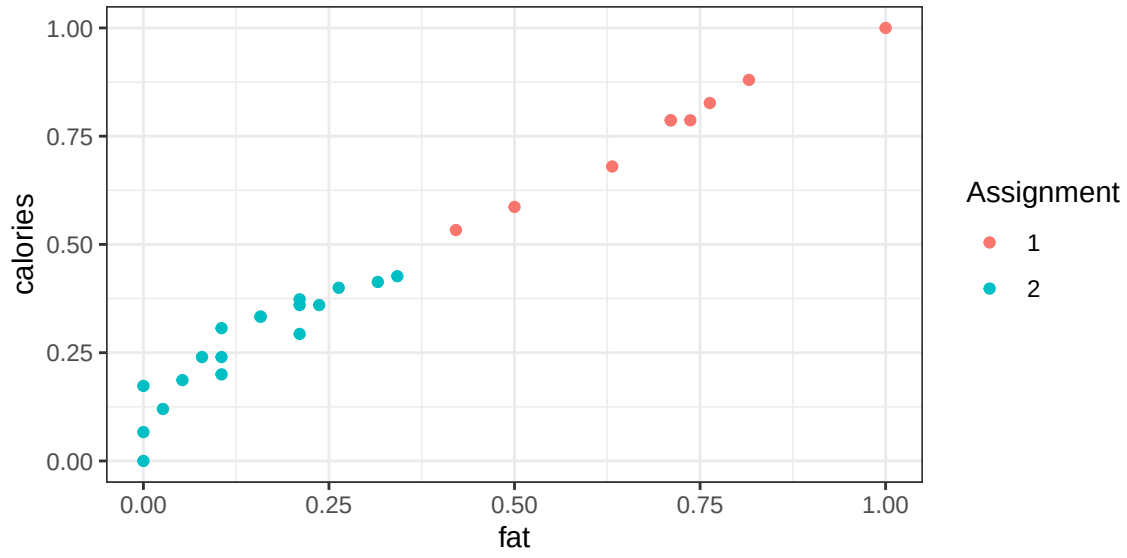
We see that the differentiating variables are fat, calories (which is a dependent variable of fat), and calcium. Then centers for protein and iron are very close together and don't appear to offer any value for dividing the data.

Since $kCalories = 4 * protein + 4 * carbs + 9 * fat$, if fat is a distinguishing variable then it easily follows that calories would appear to be a distinguishing variable as well.

First let's plot calories vs fat to confirm our intution and see the split. Indeed, there is a strong relationship between calories and fat and we see a clean split in the clusters

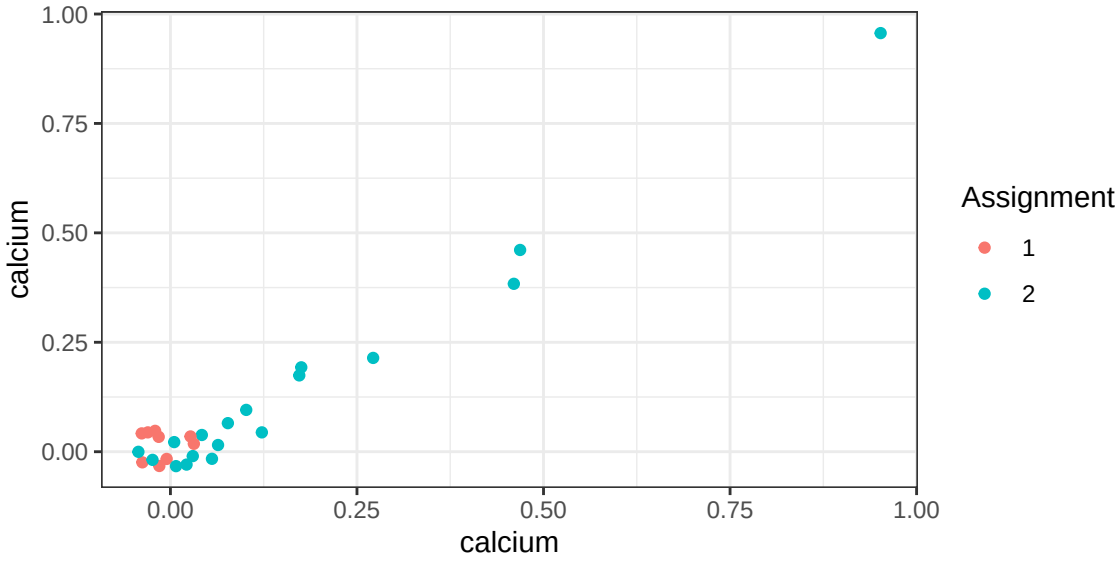
```
meats$Assignment = as.factor(meatsKMeans$cluster)

ggplot(data = meats, aes(x = fat, y = calories)) +
  geom_point(aes(color = Assignment)) +
  theme_bw()
```



Next we look at calcium. We see that most of the meats have very low calcium content. In fact, if we simply plotted the data without any manipulation, then the points in cluster 2 would appear to overlap those in cluster 1, and we'd only see cluster 2. So we add a little noise with `geom_jitter` to try separating the data a little bit to better visualize. We see that a lot of the points are all in a small cluster, without a clean split in the data. But there are some outlier points in cluster 2 that are driving the value of that centroid higher.

```
ggplot(data = meats, aes(x = calcium, y = calcium)) +
  geom_jitter(aes(color = Assignment), height = 0.05, width = 0.05) +
  theme_bw()
```



Now that we've determined fat and calcium are the two driving variables in separating the data, lets look at them plotted against each other. We see a clear break for fat, as expected. Meanwhile, there really isn't much to gain from calcium, with no clear split in the groups along the vertical axis. And so, this K-means exercise simply ended up being a long winded way to divide the meat products into groups of low-fat and high-fat meats.

```
ggplot(data = meats, aes(x = fat, y = calcium)) +
  geom_point(aes(color = Assignment)) +
  theme_bw()
```