Clustering Lab

Charlie Curtin

2024-02-29

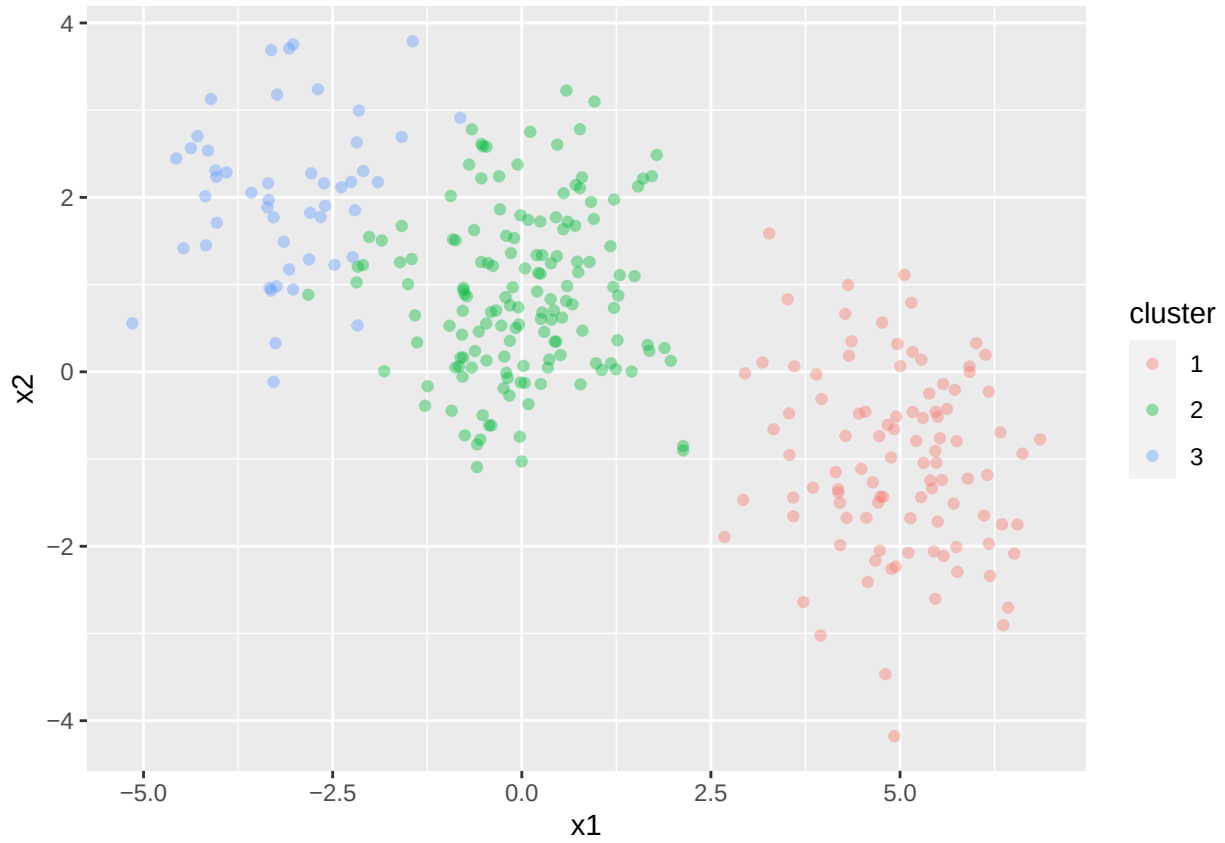
We'll start off with some simulated data that has a structure that is amenable to clustering analysis.

```
#Set the parameters of our simulated data
set.seed(101)

# create our simulated cluster structure
cents <- tibble(
  # cluster IDs
  cluster = factor(1:3),
  # 3 different cluster size
  num_points = c(100,150,50),
  # coordinates for the centers of each cluster
  x1 = c(5,0,-3),
  x2 = c(-1,1,2)
)
```

```
# create our clusters to basically validate on
# Simulate the data by passing n and mean to rnorm using map2()
labeled_pts <- cents %>%
  mutate(
    # fills in points around our cluster centers
    x1 = map2(num_points, x1, rnorm),
    x2 = map2(num_points, x2, rnorm)
  ) %>%
  select(-num_points) %>%
  unnest(cols = c(x1, x2))

# plot our labeled points
ggplot(data = labeled_pts,
       aes(x = x1, y = x2, color = cluster)) +
  geom_point(alpha = .4)
```




```
# remove clusters from the points so we can test our clustering method
points <- labeled_pts %>%
  select(-cluster)

# set up clusters
kclust <- kmeans(points, centers = 3, nstart = 25)

kclust
```

```
## K-means clustering with 3 clusters of sizes 57, 100, 143
##
## Cluster means:
##           x1         x2
## 1 -3.0016429  1.9087115
## 2  4.9628089 -1.0254665
## 3  0.1195836  0.8743487
##
## Clustering vector:
##   [1] 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
##  [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
##  [75] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 1 3 3 3
## [112] 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3
## [149] 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 1 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3
## [186] 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 1 3 3 3 3 3 3 3 3 3 3
## [223] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1
## [260] 1 1 1 1 1 3 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
```

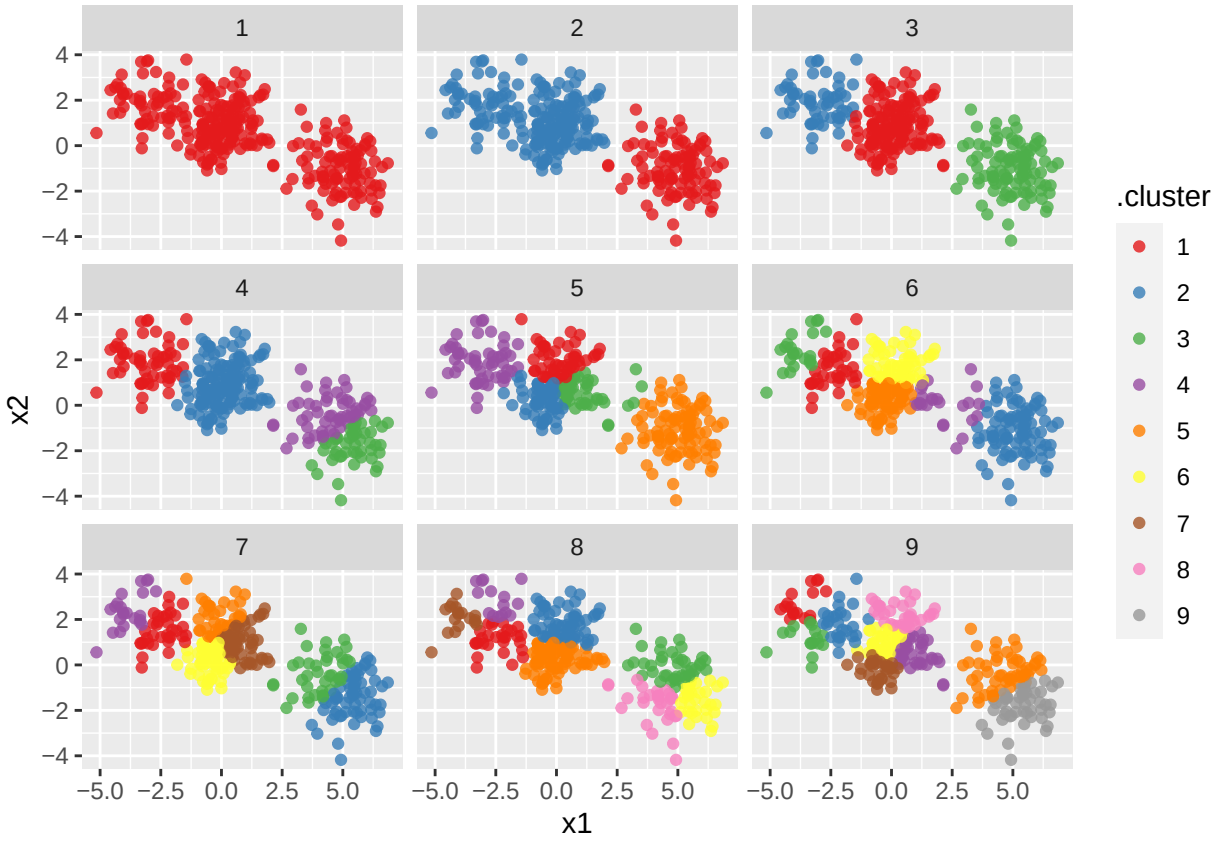
```
## [297] 1 1 1 1
##
## Within cluster sum of squares by cluster:
## [1]  86.85618 195.90406 236.30693
##  (between_SS / total_SS =  85.0 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
# now let's try a systematic method for setting k
# runs something a number of times, returns a list of dataframes
kclusts <- tibble(k = 1:9) %>%
  mutate(kclust = map(k, ~kmeans(points, .x)),
         augmented = map(kclust, augment, points))
```

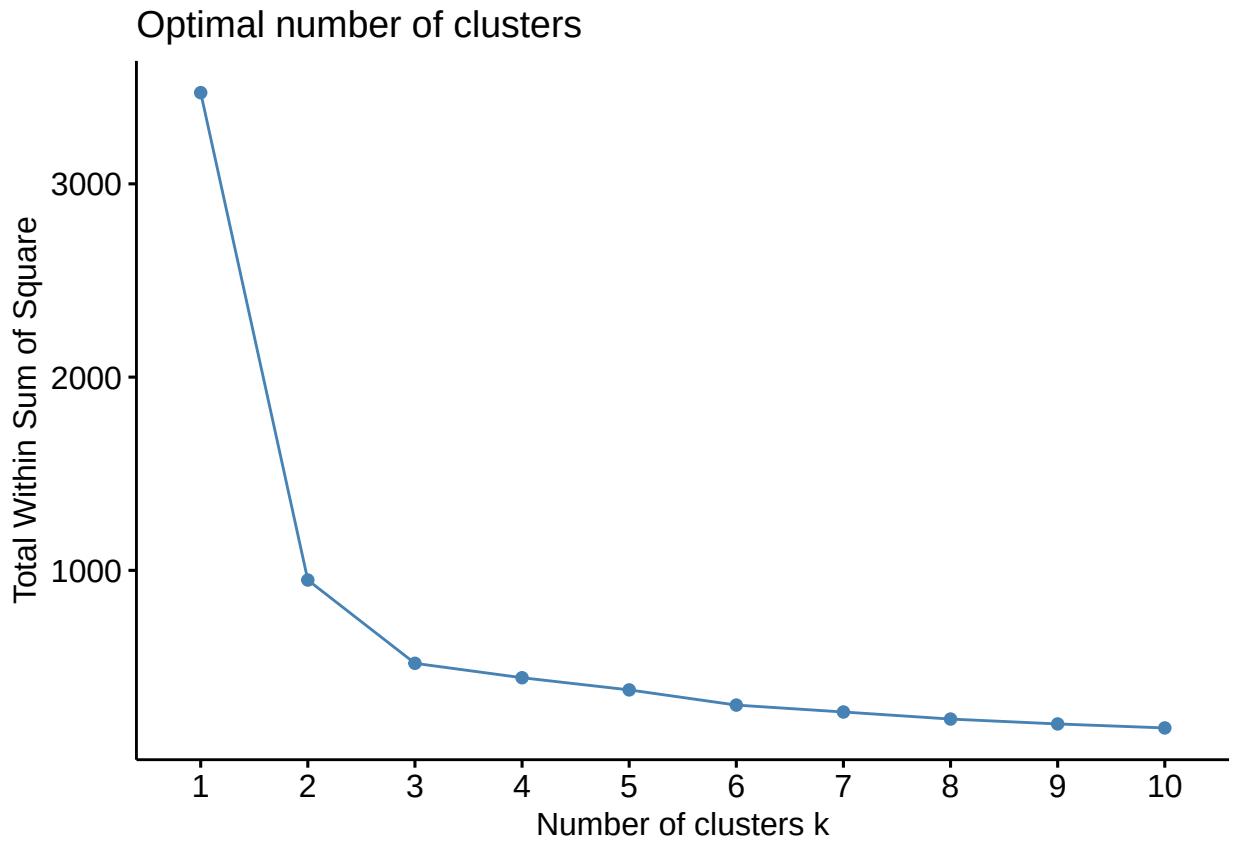
```
# append cluster assignment to tibble
# which cluster is each point associated with
assignments <- kclusts %>%
  unnest(cols = c(augmented))
```

```
# Plot each model
p1 <- ggplot(assignments, aes(x = x1, y = x2)) +
  geom_point(aes(color = .cluster), alpha = .8) +
  scale_color_brewer(palette = "Set1") +
  facet_wrap(~k)

p1
```




```
# Use a clustering function from {factoextra} to plot  total WSSs
fviz_nbclust(points, kmeans, method = "wss")
```

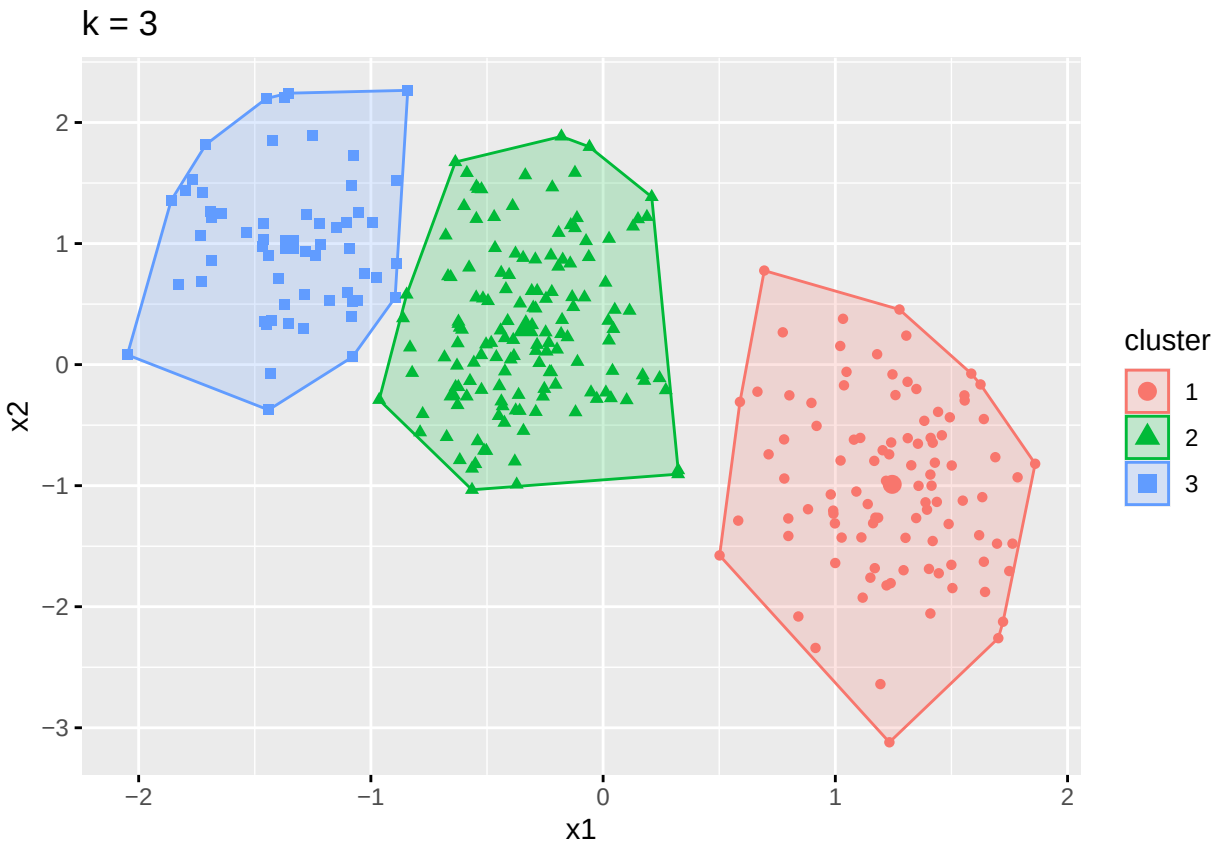



```
# Another plotting method
k3 <- kmeans(points, centers = 3, nstart = 25)

k4 <- kmeans(points, centers = 3, nstart = 5)

p3 <- fviz_cluster(k3, geom = "point", data = points) +
  ggtitle("k = 3")

p3
```



In-class assignment!

Now it's your turn to partition a dataset. For this round we'll use data from Roberts et al. 2008 on bio-contaminants in Sydney Australia's Port Jackson Bay. The data are measurements of metal content in two types of co-occurring algae at 10 sample sites around the bay.

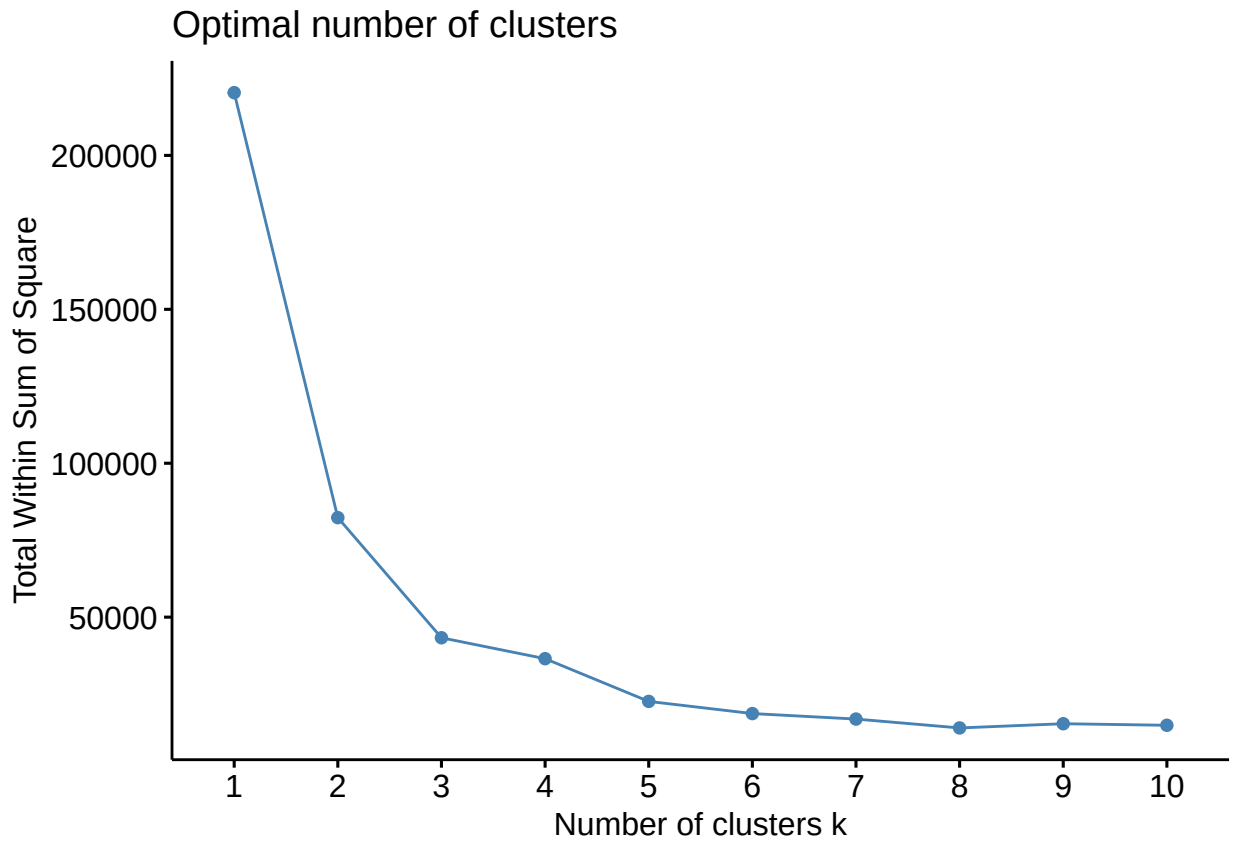
```
#Read in data
metals_dat <- read_csv(here::here("labs", "data", "Harbour_metals.csv"))

# Inspect the data

#Grab pollutant variables
metals_dat2 <- metals_dat[, 4:8]
```

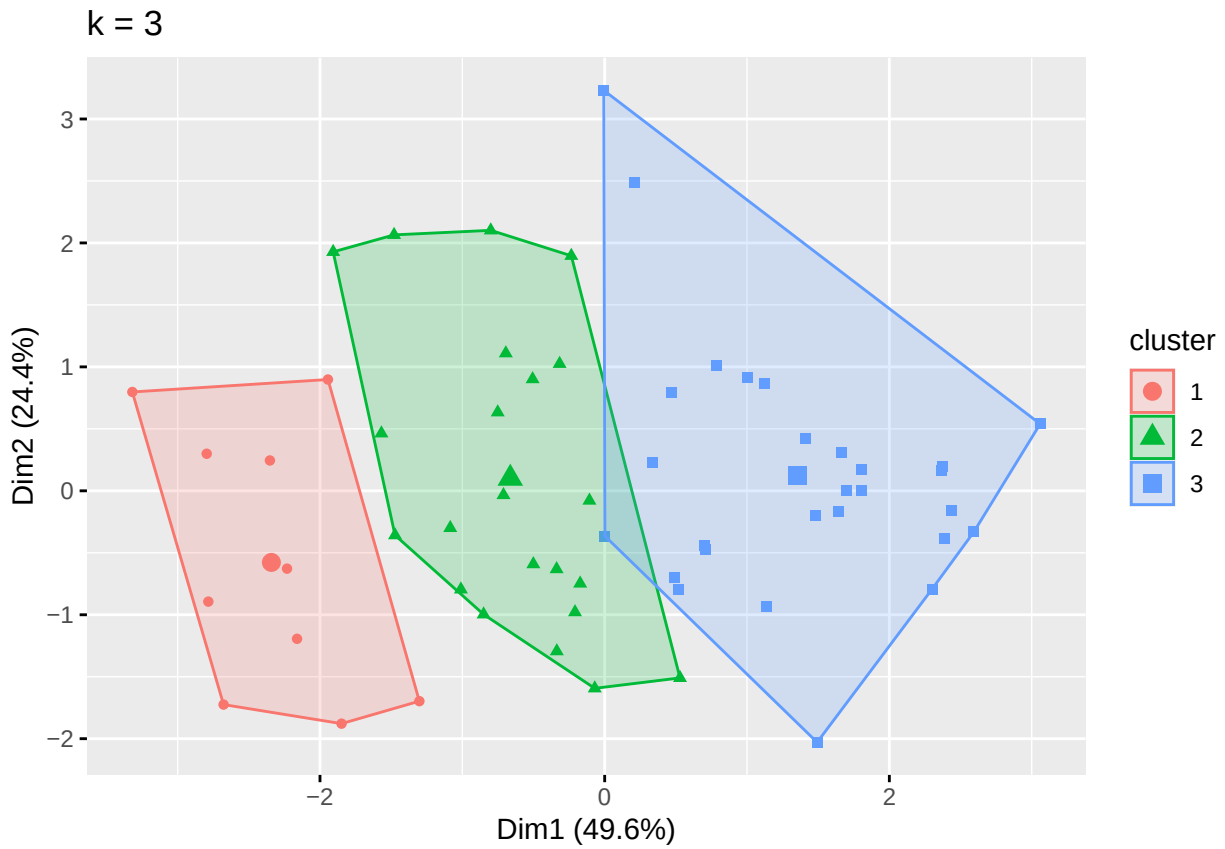
1. Start with k-means clustering - kmeans(). You can start with fviz_nbclust() to identify the best value of k. Then plot the model you obtain with the optimal value of k.

```
# visualize the best value of k for k-means
fviz_nbclust(metals_dat2, kmeans, method = "wss")
```




```
# creating a k-means model with k = 3
metals_kmeans <- kmeans(metals_dat2, centers = 3, nstart = 25)

# visualizing the clustering
fviz_cluster(metals_kmeans, geom = "point", data = metals_dat2) +
  ggtitle("k = 3")
```



Do you notice anything different about the spacing between clusters? Why might this be?

- Two of our clusters overlap, where visually it appears that a point in cluster 3 is actually closer to the center of cluster 2. This might be because we haven't done our process of reassigning points to clusters, where in the next iteration that point in cluster 3 would likely be reassigned to cluster 2.

Run summary() on your model object. Does anything stand out?

```
# summarizing our kmeans model object
metals_kmeans
```

```
## K-means clustering with 3 clusters of sizes 10, 22, 28
##
## Cluster means:
##         Cd       Cr        Cu        Mn    Ni
## 1 0.796000 6.530000 127.09000 126.28200 2.304
## 2 1.076818 6.180455  70.28455  66.79864 2.385
## 3 1.593214 3.462857  22.39893  18.85250 1.870
##
## Clustering vector:
##  [1] 2 2 1 1 1 1 3 2 2 2 2 2 2 2 1 2 2 1 3 3 3 3 3 3 3 3 2 2 2 2 3 3 3 3 3 3 1 3
## [39] 2 3 2 2 3 3 3 3 3 3 2 1 1 2 1 2 2 3 3 3 3 3
##
## Within cluster sum of squares by cluster:
## [1] 13159.695 22633.706  7511.094
```

```
##  (between_SS / total_SS =  80.4 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

- By specifying k = 3, we get 3 clusters with 10, 22, and 28 points. The cluster with 22 points has the highest within cluster sum of squares at 22,633.

2. Good, now let's move to hierarchical clustering that we saw in lecture. The first step for that is to calculate a distance matrix on the data (using dist()). Euclidean is a good choice for the distance method. Use tidy() on the distance matrix so you can see what is going on. What does each row in the resulting table represent?

```
## hierarchical clustering
# calculate distance matrix on the data using dist()
dist_matrix <- get_dist(metals_dat2, method = "euclidean")

# tidy object matrix for viewing
dist_matrix_tidy <- tidy(dist_matrix)
```

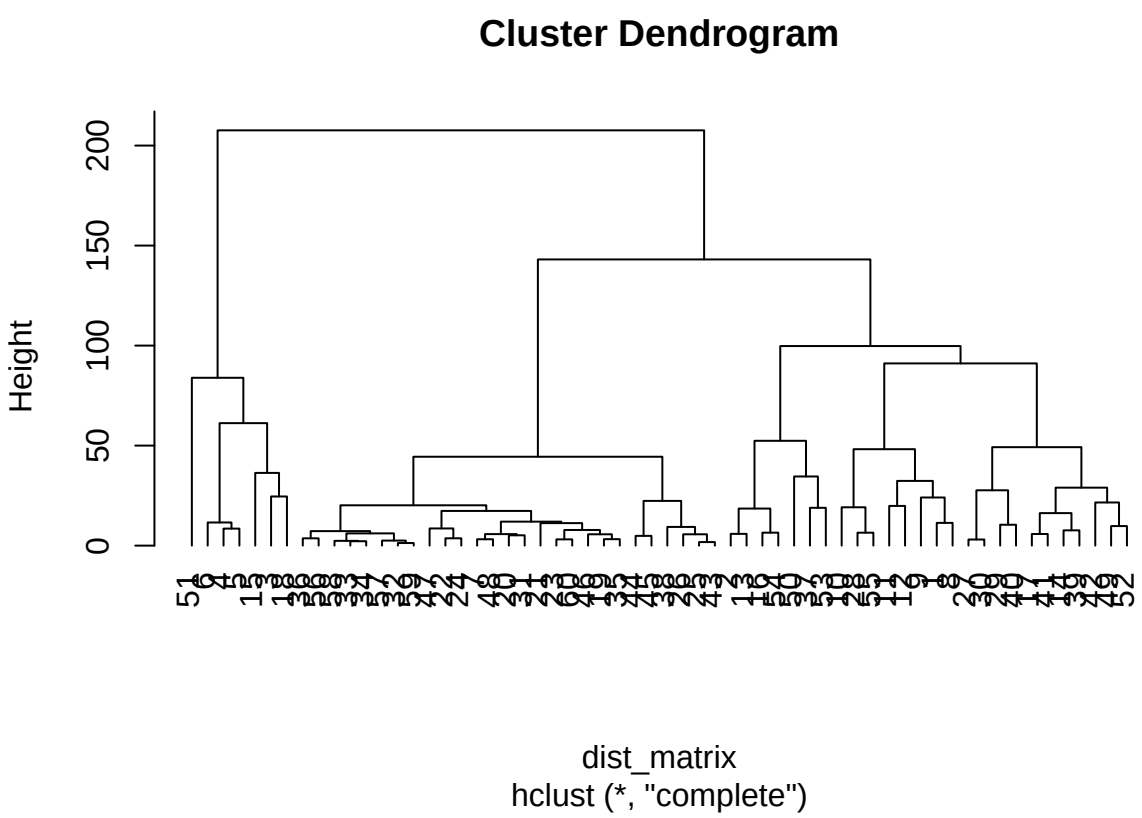
- Each row in the table shows the distance measure from a point to another point. Each point is assigned an ID from 1-60, and the whole table has a distance for every combination.

3. Then apply hierarchical clustering with hclust().

```
metals_hclust <- hclust(dist_matrix, method = "complete")
```

4. Now plot the clustering object. You can use something of the form plot(as.dendrogram()). Or you can check out the cool visual options here: https://rpubs.com/gaston/dendrograms

```
# plot our hierarchical clustering as a dendrogram
plot(metals_hclust, hang = -1)
```



How does the plot look? Do you see any outliers? How can you tell?

- If we interpret the dendrogram from the bottom up, it shows where individual observations become grouped together based on similarity until we only have one cluster. Outliers are clusters that exist as long branches before being grouped. For example, point 51 isn't grouped until height ~ 80, while points 6-18 have already arrived at one cluster after being clustered multiple times before. Once 51 and those points are clustered, they aren't clustered until they join the rest of the data.