Stat 231 - Clustering Examples

A.S. Wagaman

The text illustrates hierarchical clustering and k-means methods. k-means is very popular, so it will be our focus. (Again, you could explore more either in your final projects or in other stat courses.) Note that the kmeans function is actually in the default stats package loaded with base R, so you don't need to load any other package to use it. Here in the example, I illustrate another technique that requireds the 'mclust' package. Other packages for clustering exist as well.

Clustering Example

```
crabs <- read.table("https://awagaman.people.amherst.edu/stat240/crabs.txt", h = T)
```

For our example, we have data from Campbell & Mahon (1974). This is data on the morphology of rock crabs of genus Leptograpsus. There are 50 specimens of each sex of each of two color forms. The variables are:

- sp 'species', coded B (blue form) or O (orange form)
- sex coded M or F
- FL frontal lip of carapace (mm)
- RW rear width of carapace (mm)
- CL length along the midline of carapace (mm)
- CW maximum width of carapace (mm)
- BD body depth (mm)

We will look for natural groups in the data and see if the natural groups correspond to sex or species differences.

Visualizing clustering solutions is challenging without some dimension reduction techniques, assuming more than 2 variables are used to create the clusters. While we will not cover these methods in depth, I encourage you to explore if you choose clustering as a method for your final projects.

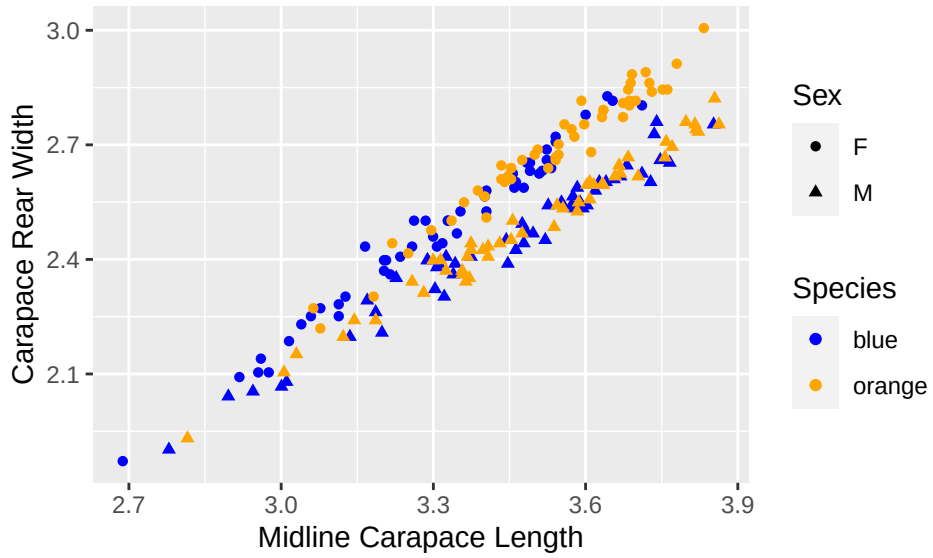
K-means Methods

Two variables

We'll use a length and width variable to look for clusters of crabs.

```
ggplot(data = crabs, aes(x = CL, y = RW, color = sp, shape = sex)) +
  geom_point() +
  scale_color_manual(values = c("blue", "orange")) +
  labs(x = "Midline Carapace Length",
       y = "Carapace Rear Width",
```

```
       color = "Species",
       shape = "Sex")
```



What happens if we look for 2 clusters?

```
set.seed(231)
clustering_vars <- c("CL", "RW")
crabs_km2 <- crabs %>%
  select(clustering_vars) %>%
  kmeans(centers = 2, nstart = 20)
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(clustering_vars)` instead of `clustering_vars` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

```
# Vector of cluster assignments
crabs_km2$cluster
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [75] 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
## [112] 1 1 1 1 1 1 1 1 1 1 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
## [149] 2 2 1 1 1 1 1 1 2 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
## [186] 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

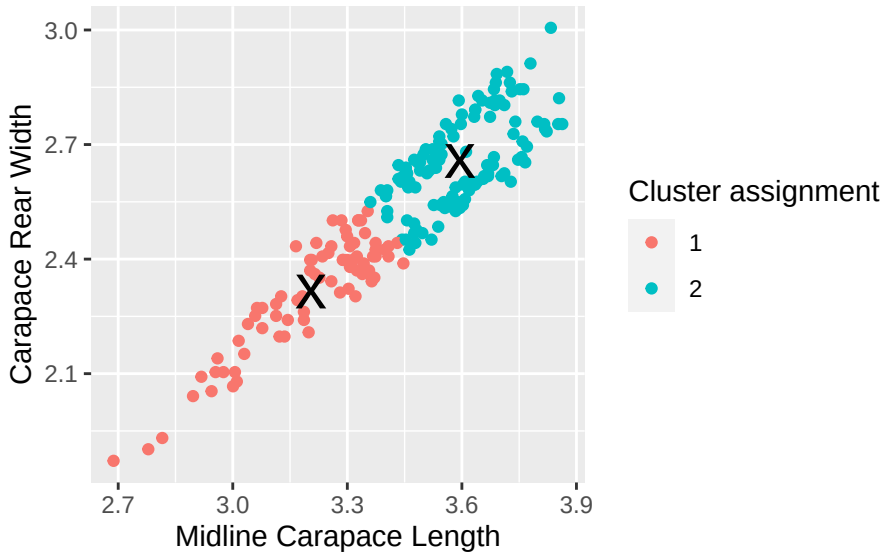
```
# The centroids for the fit
crabs_km2$centers
```

```
##         CL       RW
## 1 3.204656 2.314652
## 2 3.594750 2.656493
```

```
# Add cluster assignment to the data frame
crabs <- crabs %>%
  mutate(clusters2 = factor(crabs_km2$cluster))

# Visualize the cluster assignments and centroids
```

```
ggplot(data = crabs, aes(x = CL, y = RW)) +
  geom_point(aes(color = clusters2)) +
  coord_fixed() +
  geom_point(data = data.frame(crabs_km2$centers),
             aes(x = CL, y = RW),
             pch = "x", size = 8) +
  labs(x = "Midline Carapace Length",
       y = "Carapace Rear Width",
       color = "Cluster assignment")
```



This solution is on the unscaled variables, which can have a BIG effect on the clustering solution. Scaling matters. Let's scale all the numeric variables in the data set (even though we are currently only using 2 of them), and try the 2 cluster solution again.

```
# Standardize or scale() numeric variables (subtract mean and divide by SD)
crabs <- crabs %>%
  mutate(across(where(is.numeric),  ~ scale(.)[,1], .names = "{.col}_scaled"))
```

```
set.seed(231)
clustering_vars <- c("CL_scaled", "RW_scaled")
crabs_km2s <- crabs %>%
  select(clustering_vars) %>%
  kmeans(centers = 2, nstart = 20)

# Vector of cluster assignments
crabs_km2s$cluster
```

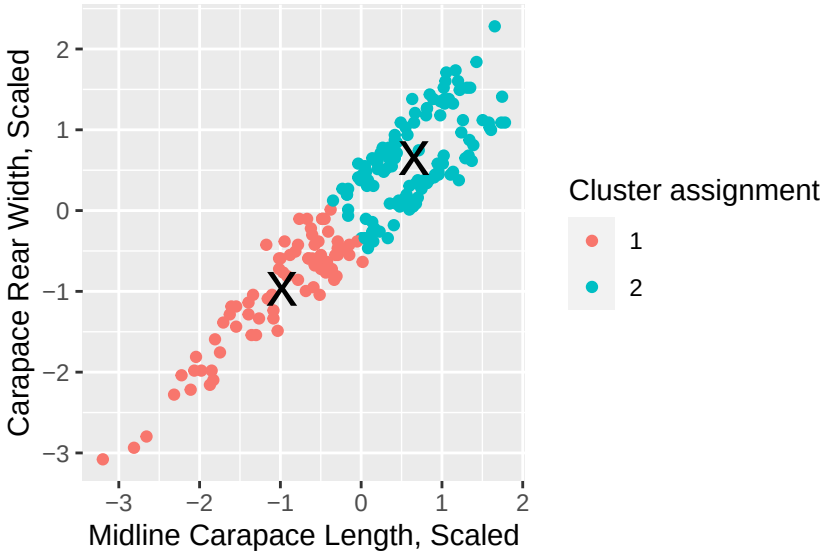
```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [38] 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [75] 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
## [112] 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [149] 2 2 1 1 1 1 1 1 2 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
## [186] 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```
# The centroids for the fit
crabs_km2s$centers
```

```
##    CL_scaled  RW_scaled
## 1 -0.9817432 -0.9711038
## 2  0.6544955  0.6474025
```

```r
# Add cluster assignment to the data frame
crabs <- crabs %>%
  mutate(clusters2s = factor(crabs_km2$cluster))

# Visualize the cluster assignments and centroids
ggplot(data = crabs, aes(x = CL_scaled, y = RW_scaled)) +
  geom_point(aes(color = clusters2s)) +
  coord_fixed() +
  geom_point(data = data.frame(crabs_km2s$centers),
             aes(x = CL_scaled, y = RW_scaled),
             pch = "x", size = 8) +
  labs(x = "Midline Carapace Length, Scaled",
       y = "Carapace Rear Width, Scaled",
       color = "Cluster assignment")
```



There isn't much change here because both measurement variables were on the same scale to begin with. However, scaling is VERY important to consider.

What about 4 clusters?

```r
set.seed(231)

crabs_km4s <- crabs %>%
  select(clustering_vars) %>%
  kmeans(centers = 4, nstart = 20)

# Vector of cluster assignments
crabs_km4s$cluster
```

```
##   [1] 1 1 1 1 1 1 3 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4
##  [38] 4 4 4 4 4 4 2 4 4 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3
##  [75] 3 3 3 3 4 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 2 2 2 2 1 1 1 1 1 1 3 3 3 3 3
## [112] 3 3 3 3 3 3 3 3 3 3 3 4 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 2 4 2 2 2 2 2 2 2
```

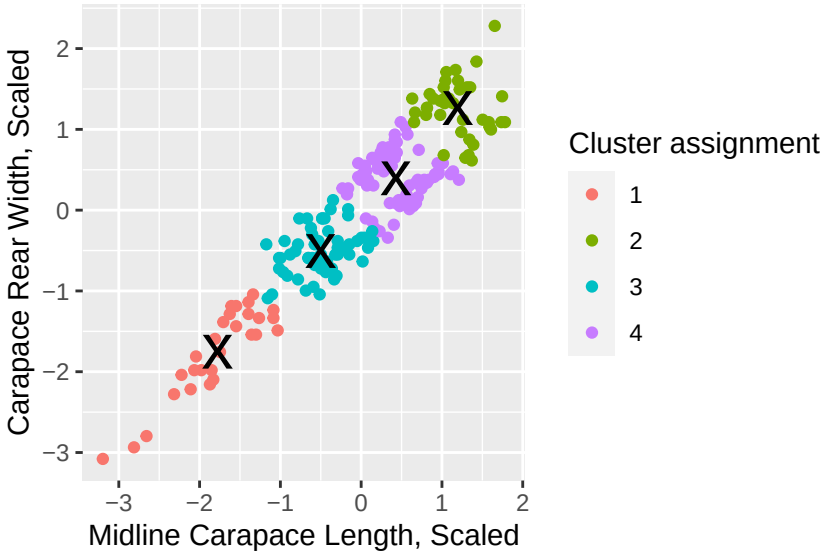
```
## [149] 2 2 1 1 3 3 3 3 3 3 4 4 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 2 4 4 2 4 4 4 2 2 2
## [186] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```
# The centroids for the fit
crabs_km4s$centers
```

```
##    CL_scaled  RW_scaled
## 1 -1.7786527 -1.7544835
## 2  1.1949420  1.2615637
## 3 -0.5013026 -0.5081765
## 4  0.4303346  0.3919839
```

```
# Add cluster assignment to the data frame
crabs <- crabs %>%
  mutate(clusters4s = factor(crabs_km4s$cluster))

# Visualize the cluster assignments and centroids
ggplot(data = crabs, aes(x = CL_scaled, y = RW_scaled)) +
  geom_point(aes(color = clusters4s)) +
  coord_fixed() +
  geom_point(data = data.frame(crabs_km4s$centers),
             aes(x = CL_scaled, y = RW_scaled),
             pch = "x", size = 8) +
  labs(x = "Midline Carapace Length, Scaled",
       y = "Carapace Rear Width, Scaled",
       color = "Cluster assignment")
```



The within group sum of squares (distance from points to the center of the cluster, summed over all points and then over all clusters) can be obtained for each solution. Generally, increasing the number of clusters will decrease the WGSS, but it's a tradeoff.

```
crabs_km2s$tot.withinss
```

```
## [1] 143.7517
```

```
crabs_km4s$tot.withinss
```

```
## [1] 50.50465
```

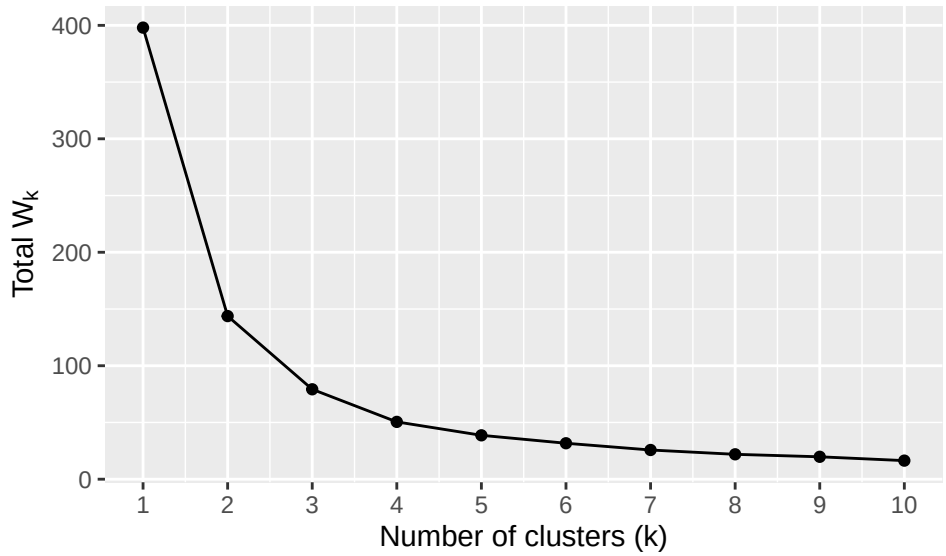
We can examine the tradeoff using an elbow plot:

```
elbow_plot <- data.frame(clusters = 1:10,
                         within_ss = rep(NA, 10))

set.seed(75)
for (i in 1:10){
  crabs_kms_out <- crabs %>%
    select(clustering_vars) %>%
    kmeans(centers = i, nstart = 20)

  elbow_plot$within_ss[i] <- crabs_kms_out$tot.withinss
}

# Construct elbow plot
ggplot(elbow_plot, aes(x = clusters, y = within_ss)) +
  geom_point() +
  geom_line() +
  scale_x_continuous(breaks = 1:10) +
  labs(x = "Number of clusters (k)", y = expression("Total W"[k]))
```



This can be used to help choose a value of k, which you want to do where the elbow is or where the plot starts to level off.

All measurement variables

Limiting the clustering to 2 variables means we lose a lot of the available data. So, let's expand our included variables.

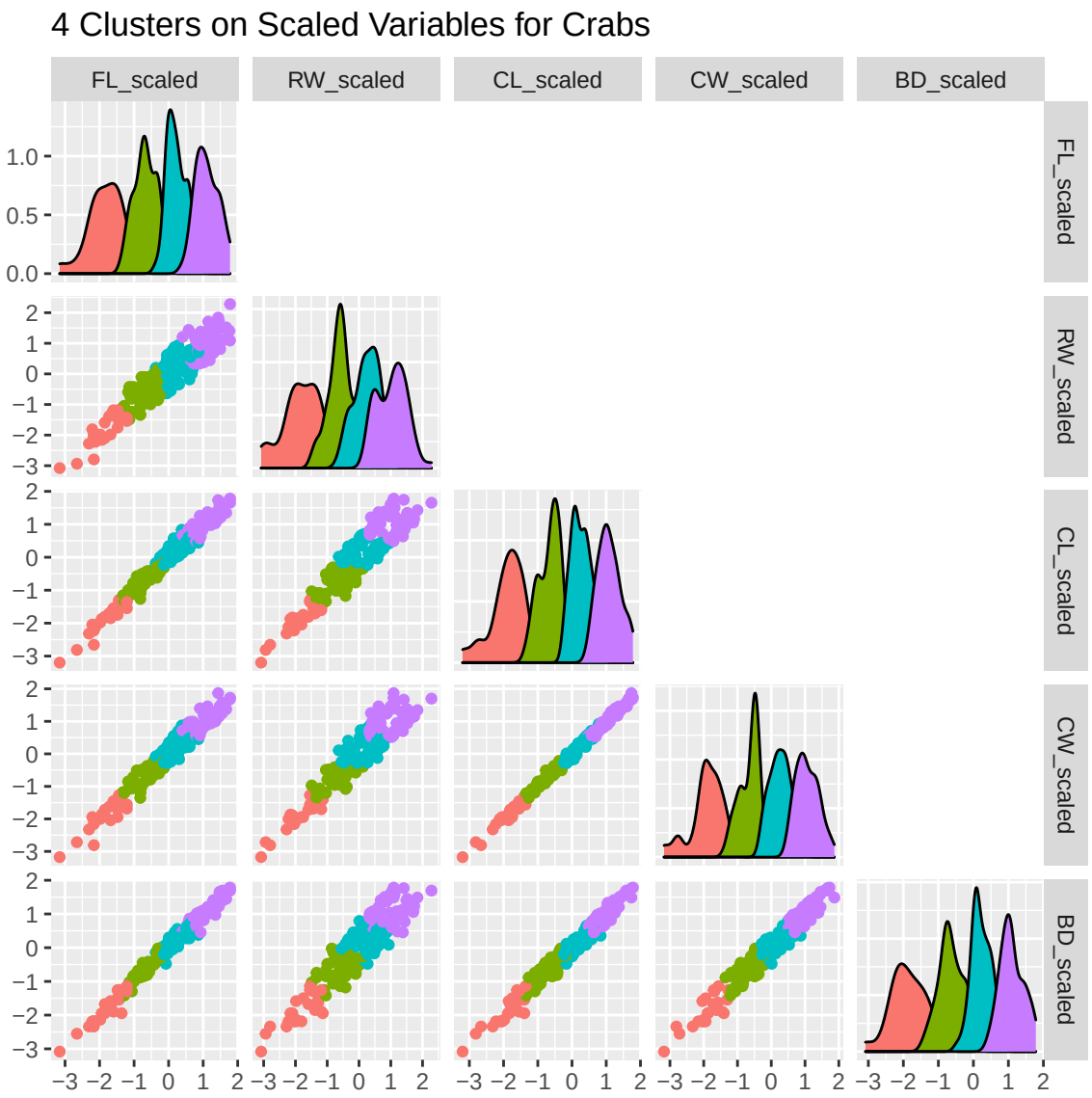
```
set.seed(231)
crabs_kms <- crabs %>%
  select(ends_with("scaled")) %>%
  kmeans(centers = 4, nstart = 20)

crabs <- crabs %>%
  mutate(clusters4_scaled = factor(crabs_kms$cluster))
```

Visualizing the solution is hard without a dimension reduction technique (such as Principal Components Analysis, PCA), so we'll resort to a scatterplot matrix.

```
# Scatterplot matrix of clusters based on scaled variables
GGally::ggpairs(data = crabs, aes(color = clusters4_scaled),
                columns = c("FL_scaled",
                            "RW_scaled",
                            "CL_scaled",
                            "CW_scaled",
                            "BD_scaled"),
                upper = list(continuous = "blank")) +
  labs(title = "4 Clusters on Scaled Variables for Crabs")
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```



Unsurprisingly, as with a number of biological applications, the clusters being found are directly related to

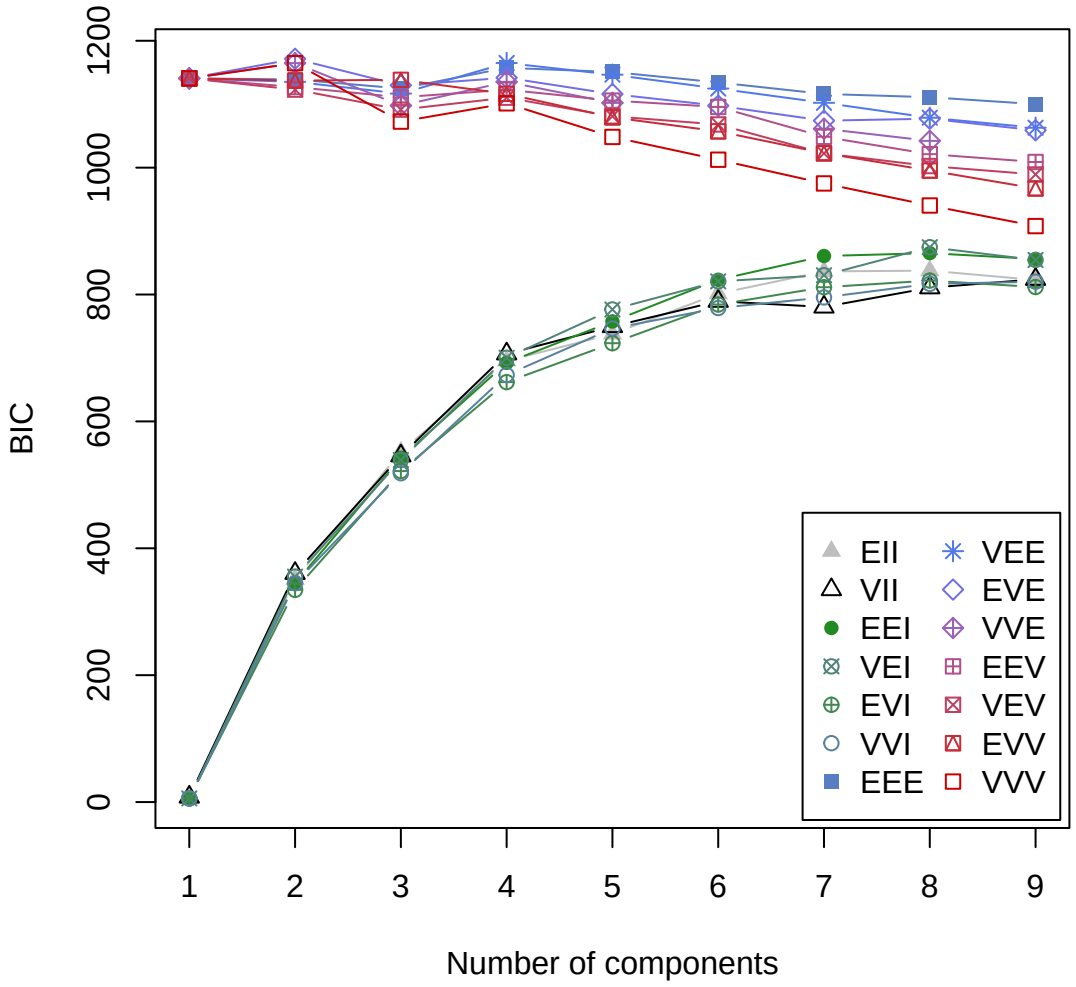
overall size of the biological specimen.

Other Algorithms

There are LOTS of other clustering algorithms. Here, I demo model-based approach.

For model-based clustering, here I try using just three of the variables in the crabs data set. This uses finite normal mixture modeling. The method doesn't need to standardize - it will choose models with different means and variances for the variables as needed on its own.

```
crabsub <- crabs %>% select(FL, RW, BD)
mclustsol <- mclustBIC(crabsub)
plot(mclustsol)
```



This plot shows the MANY different models in mclust and how they did with this data set.

```
summary(mclustsol)
```

```
## Best BIC values:
```

```
##             EVE,2       VEE,4       VVE,2
## BIC      1171.558 1164.919339 1164.757977
## BIC diff    0.000   -6.638924   -6.800285
```

Here, we see which algorithms had the best BIC (Bayesian Information Criterion) values. These are the models you might want to investigate further.

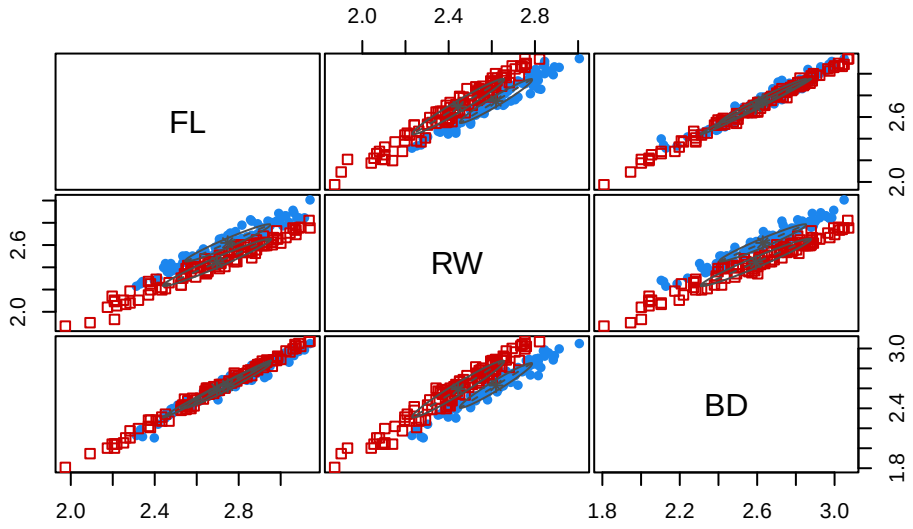
To try out the first solution, we can do:

```
mod1 <- Mclust(crabsub, x = mclustsol)
summary(mod1, parameters = TRUE)
```

```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust EVE (ellipsoidal, equal volume and orientation) model with 2 components:
##
##  log-likelihood   n df      BIC      ICL
##        625.5165 200 15 1171.558 1145.848
##
## Clustering table:
##   1   2
##  90 110
##
## Mixing probabilities:
##         1         2
## 0.4515497 0.5484503
##
## Means:
##        [,1]     [,2]
## FL 2.747560 2.696420
## RW 2.619368 2.443978
## BD 2.629102 2.592910
##
## Variances:
## [,,1]
##            FL         RW         BD
## FL 0.04104939 0.03278606 0.04449077
## RW 0.03278606 0.02803614 0.03629246
## BD 0.04449077 0.03629246 0.05019165
## [,,2]
##            FL         RW         BD
## FL 0.06604636 0.05335567 0.07278424
## RW 0.05335567 0.04539320 0.05898582
## BD 0.07278424 0.05898582 0.08105865
```

This says that in the 3-D space we had, it found 2 clusters of roughly equal size. It shows us the group means and the variances in each cluster.

```
plot(mod1, what = "classification")
```



I chose only 3 variables so I could make the plot and it still be readable. We can see that it finds two groups that are fairly well separated along some of the projections. You can do this in higher dimensions, obviously, but the plot may not look at nice.

For more on mclust (and how I did this quickly), you can check out this site, which is the package vignette from R here.