Clustering

Tony Ni, Antonella Basso, Jose Lopez

6/21/2020

Libraries

Reading in Data

```
setwd("~/harvard-summer-biostats")
df <- read_csv("data/wide_illinois.csv") #read in data
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   well_id = col_character(),
##   site = col_character(),
##   disposal_area = col_character(),
##   type = col_character(),
##   gradient = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
df1 <- na.omit(df) #get rid of na's
```

Hierarchical Methods

If we want to look for cereal groups via hierarchical clustering, we need to construct a distance matrix. Distances are constructed with the *dist* function, and you need to choose whether you compute them on scaled or unscaled variables (standardize or not).

```
df.dist <- dist(df1[, -c(1:5)])
```

Now we look at how hierarchical clustering is applied. The relevant function is *hclust*.

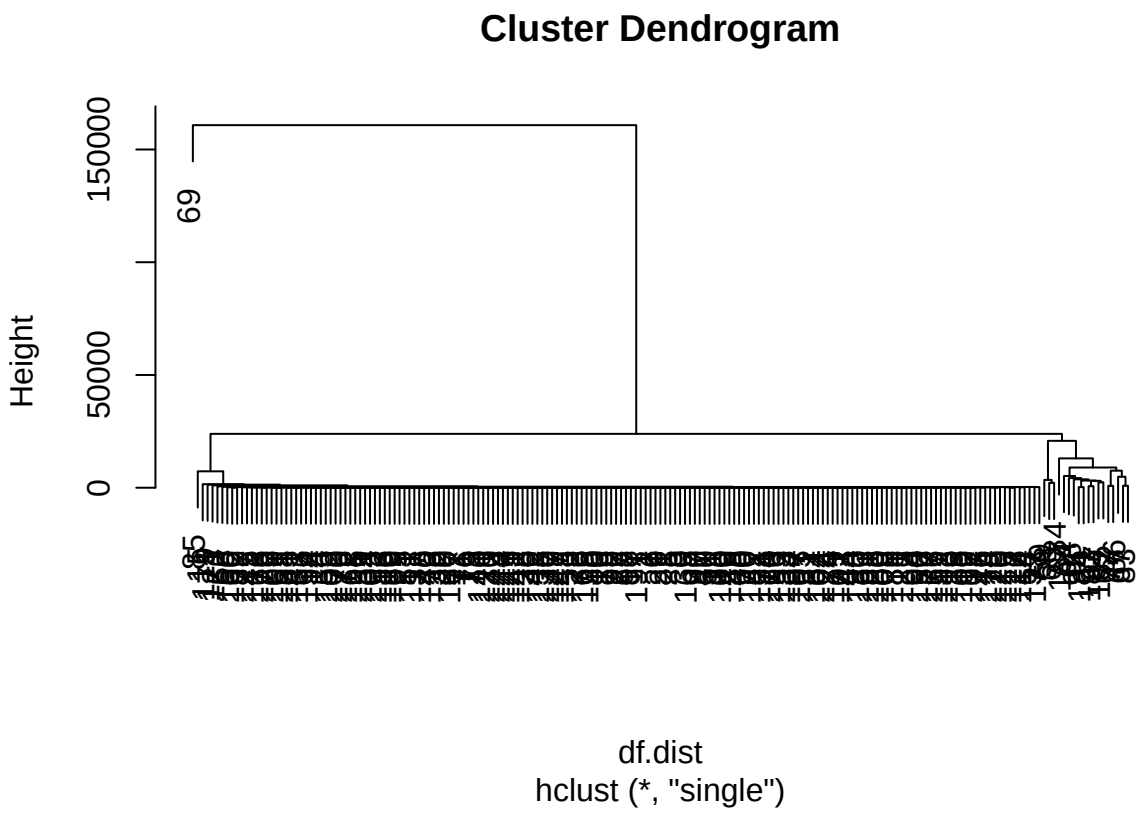
```
hcsingle <- hclust(df.dist, method = "single")
list(hcsingle) # reminds you of properties of the solution, if desired
```

```
## [[1]]
##
## Call:
```

```
## hclust(d = df.dist, method = "single")
##
## Cluster method   : single
## Distance         : euclidean
## Number of objects: 191
```

This creates the solution, and we can look at the dendrogram as:

```
plot(hcsingle)
```



The options for hclust in terms of linkages are provided in the help under options for method. The following options are listed: "ward.D", "ward.D2", "single", "complete", "average", "mcquitty", "median" or "centroid".

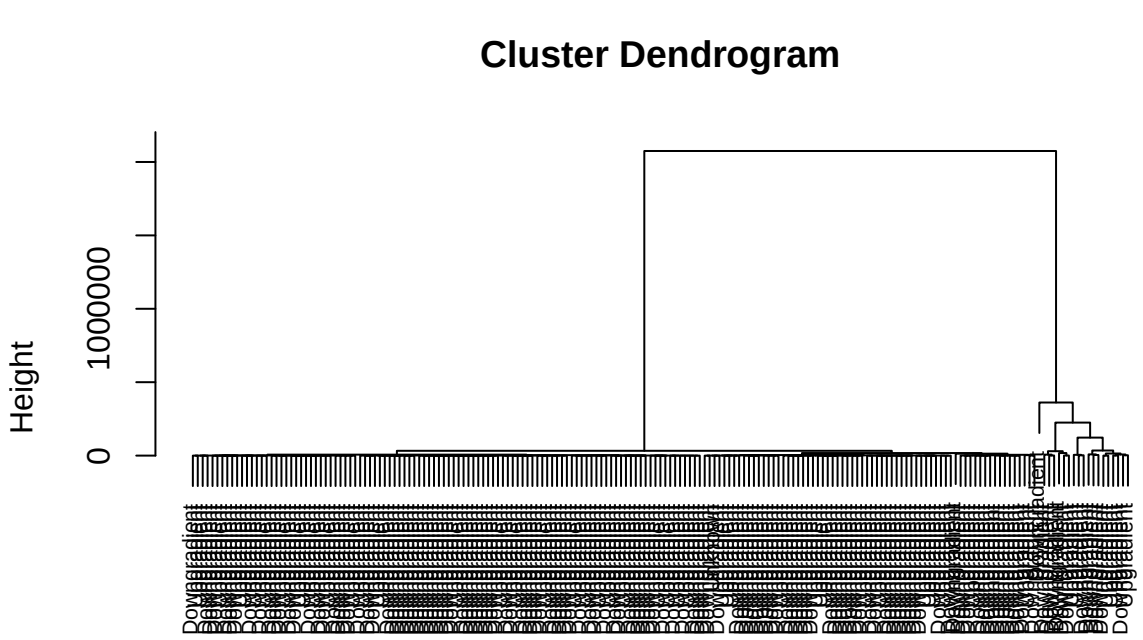
In order to obtain cluster labels, we need to *cut* our dendrograms.

```
singleSol <- (cutree(hcsingle, k = 3)) #cluster labels are numeric, k= # clusters
summary(as.factor(singleSol)) #as factor to get table
```

```
##   1   2   3
## 172  18   1
```

To learn more details about the clusters we found:

```
hcward <- hclust(df.dist, method = "ward.D")
plot(hcward, labels = df1$gradient, cex = 0.7) #cex adjusts size of label
```



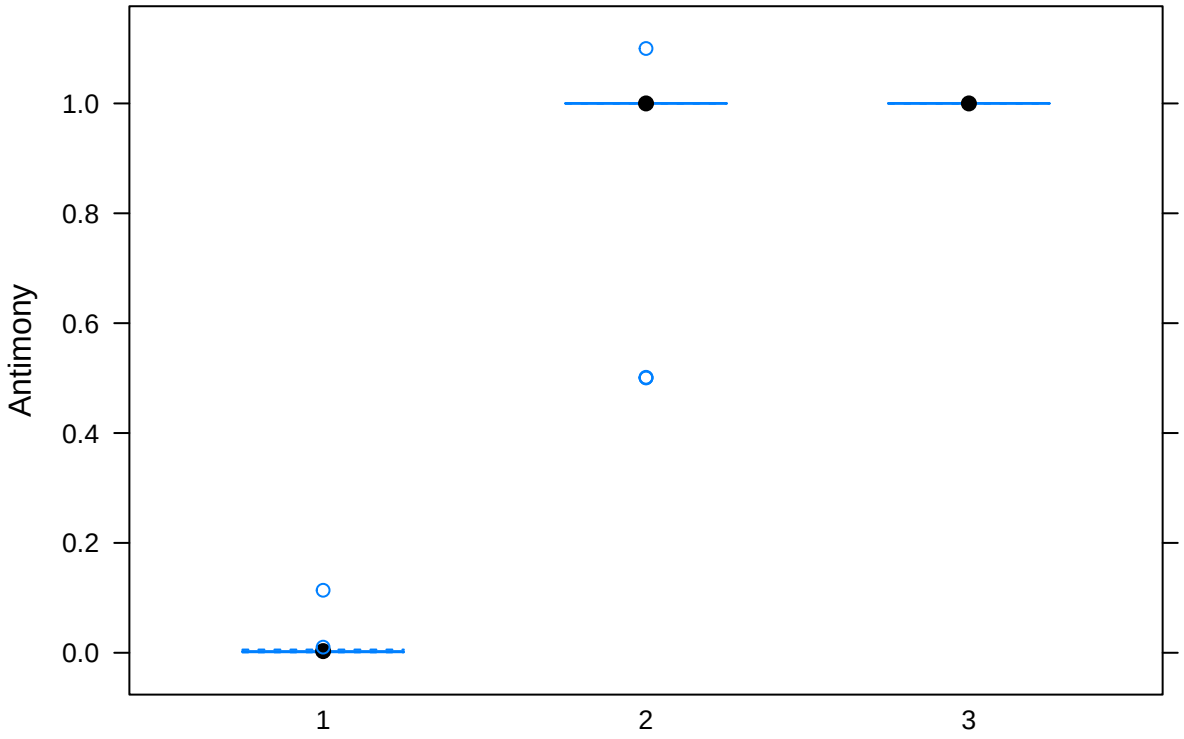

```
wardSol <- (cutree(hcward, k = 3)) #cluster labels are numeric, k= # clusters
summary(as.factor(wardSol)) #as factor to get table
```

```
##   1   2   3
## 172  18   1
```

```
favstats(Antimony ~ wardSol, data = df1) #can choose any variable
```

```
##   wardSol       min    Q1 median    Q3       max        mean         sd   n
## 1       1 0.0007625 0.001  0.003 0.003 0.1137778 0.002903086 0.00858458 172
## 2       2 0.5005000 1.000  1.000 1.000 1.1000000 0.922361111 0.19538836  18
## 3       3 1.0000000 1.000  1.000 1.000 1.0000000 1.000000000         NA   1
##   missing
## 1       0
## 2       0
## 3       0
```

```
bwplot(Antimony ~ as.factor(wardSol), data = df1)
```



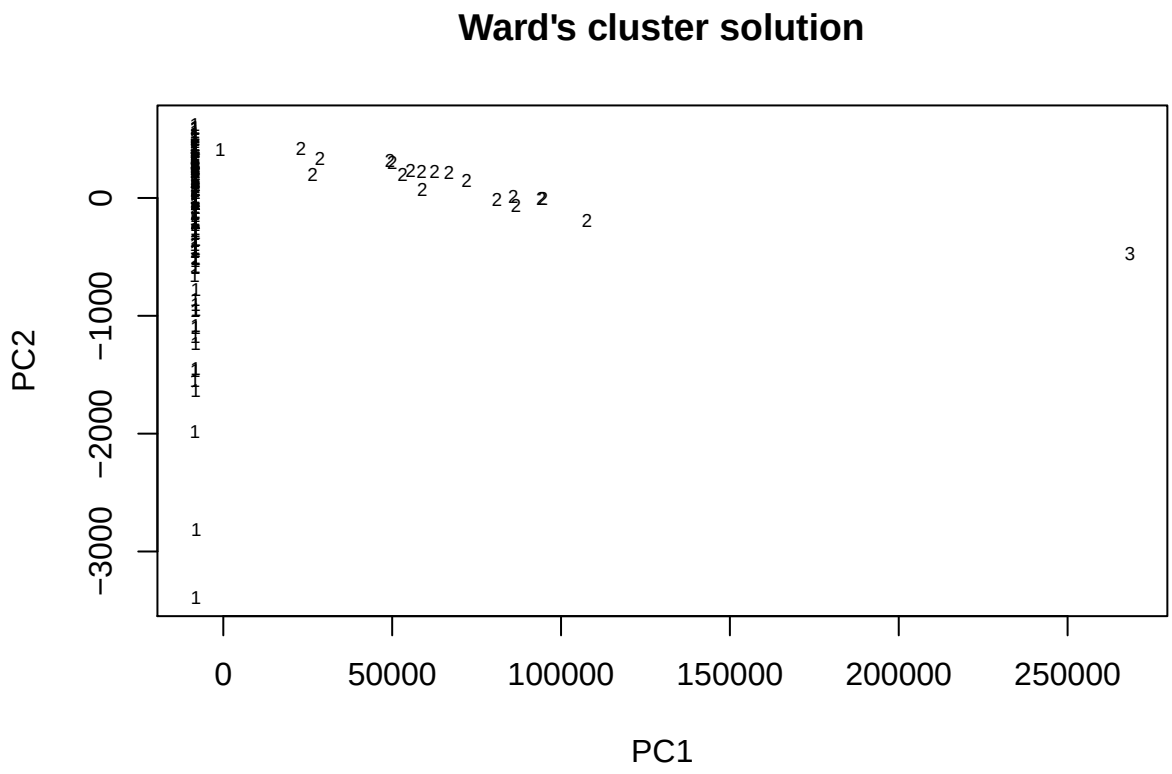
Our cluster sizes are extremely uneven... Our first clsuter has 172 wells and the second has 18 and third has 1.

```
favstats(#chemical here ~ wardSol, data = df1)
bwplot(#chemical here ~ as.factor(wardSol), data = df1)
```

We can view the solution in the PC space (say 2-D) to see how well-separated the clusters are in that space. Because we used an unstandardized distance, I will run the PCA on the covariance matrix.

```
dfPCA <- princomp(df1[, -c(1:5)], cor = FALSE)

plot(dfPCA$scores[, 1:2], type = "n", xlab = "PC1", ylab = "PC2", main = "Ward's cluster solution") #bl
text(dfPCA$scores[, 1:2], labels = wardSol, cex = 0.6) #add the text
```



K-means Methods

For k-means, we don't need to compute the distance matrix yourself. We feed the function the data set to operate on:

```
Ksol1 <- kmeans(scale(df1[, -c(1:5)]), centers = 3) #centers is the # of clusters desired
list(Ksol1) #so you can see what it gives you
```

```
## [[1]]
## K-means clustering with 3 clusters of sizes 17, 173, 1
##
## Cluster means:
##     Antimony    Arsenic     Barium  Beryllium      Boron    Cadmium    Calcium
## 1  3.0076716  2.3261871  2.4192000  2.6961948  2.8248526  2.6926073  2.2277125
## 2 -0.3140143 -0.2530565 -0.2671769 -0.3030062 -0.3029845 -0.3028129 -0.2710679
## 3  3.1940575  4.2335941  5.0951954  6.5847552  4.3938309  6.6123012  9.0236344
##      Chloride   Chromium     Cobalt    Fluoride       Lead    Lithium
## 1 -0.38437120  0.6586752  0.8657475 -0.13330953  0.5544924  2.2718424
## 2  0.04026449 -0.1376811 -0.1571124  0.01357571 -0.1305658 -0.2689775
## 3 -0.43144705 12.6213529 12.4627380 -0.08233520 13.1615161  7.9117935
##      Mercury Molybdenum          pH Radium 226+228   Selenium     Sulfate
## 1  2.9710278  2.1866659  0.51079489     0.36703379  2.9049504 -0.55303625
## 2 -0.3138067 -0.2404121 -0.05372966    -0.03928004 -0.3056403  0.05677967
## 3  3.7810955  4.4179677  0.61171847     0.55587260  3.4916074 -0.42126674
```

```
##     Thallium Total Dissolved Solids
## 1  2.9089714            -0.49322486
## 2 -0.3122229             0.05024777
## 3  4.5620477            -0.30804231
##
## 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 1 2 1 1 2 1 1 2 1 1 3 2 2 2 2 2
##  [75] 2 2 1 2 2 2 2 2 1 1 2 1 1 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [112] 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 2 2
## [149] 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 1 1 1
## [186] 2 1 2 2 2 2
##
## Within cluster sum of squares by cluster:
## [1]  340.2996 1140.6765    0.0000
##  (between_SS / total_SS =  62.9 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

The list option provides us with lots of information. We can pull out the cluster means as:

```
Ksol1$centers
```

```
##     Antimony    Arsenic     Barium  Beryllium      Boron    Cadmium    Calcium
## 1  3.0076716  2.3261871  2.4192000  2.6961948  2.8248526  2.6926073  2.2277125
## 2 -0.3140143 -0.2530565 -0.2671769 -0.3030062 -0.3029845 -0.3028129 -0.2710679
## 3  3.1940575  4.2335941  5.0951954  6.5847552  4.3938309  6.6123012  9.0236344
##      Chloride   Chromium     Cobalt    Fluoride       Lead    Lithium
## 1 -0.38437120  0.6586752  0.8657475 -0.13330953  0.5544924  2.2718424
## 2  0.04026449 -0.1376811 -0.1571124  0.01357571 -0.1305658 -0.2689775
## 3 -0.43144705 12.6213529 12.4627380 -0.08233520 13.1615161  7.9117935
##      Mercury Molybdenum          pH Radium 226+228   Selenium     Sulfate
## 1  2.9710278  2.1866659  0.51079489     0.36703379  2.9049504 -0.55303625
## 2 -0.3138067 -0.2404121 -0.05372966    -0.03928004 -0.3056403  0.05677967
## 3  3.7810955  4.4179677  0.61171847     0.55587260  3.4916074 -0.42126674
##     Thallium Total Dissolved Solids
## 1  2.9089714            -0.49322486
## 2 -0.3122229             0.05024777
## 3  4.5620477            -0.30804231
```

We can also get the clustering vector (with the cluster labels) as:

```
Ksol1$cluster
```

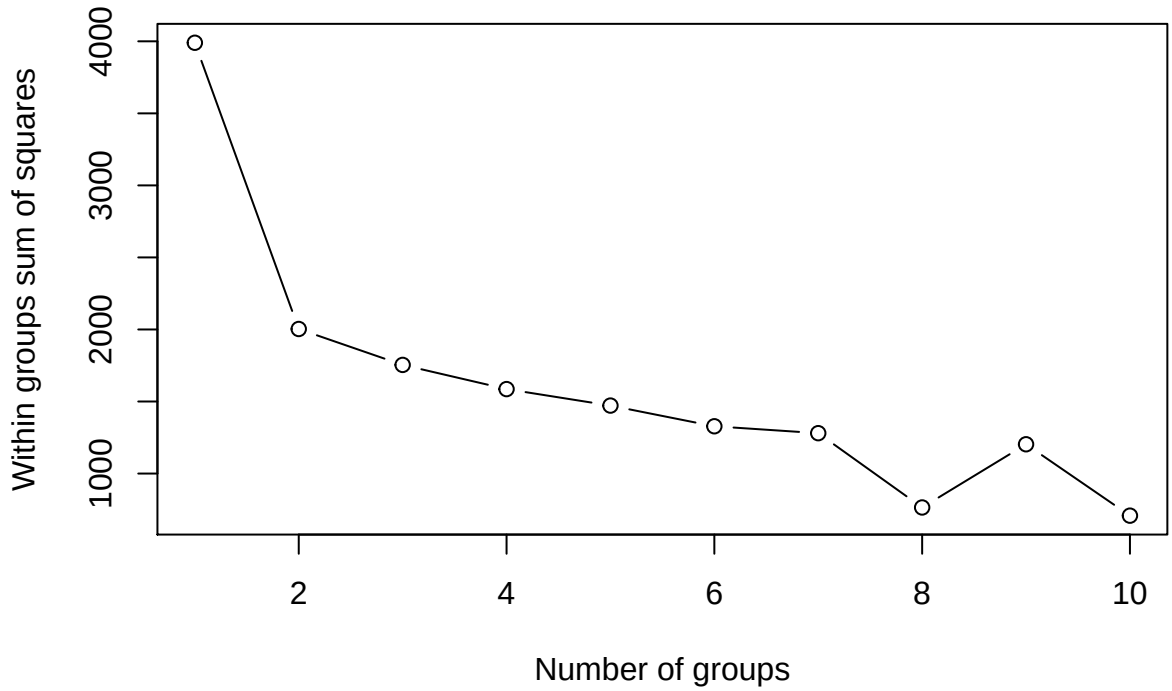
```
##   [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [38] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 1 2 1 1 2 1 1 3 2 2 2 2 2
##  [75] 2 2 1 2 2 2 2 2 1 1 2 1 1 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [112] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [149] 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 1 1 1
## [186] 2 1 2 2 2 2
```

In order to determine if we have chosen a "good" value of the number of clusters, we can look at the within cluster sum of squares for this solution and a few other options for k, the number of clusters. This runs the solution from 1 to 3 clusters and pulls the within group sum of squares from each.

```
n <- nrow(df1) #number of observations

wss <- rep(0, 10) #creates 8 copies of 0 to create an empty vector
for(i in 1:10){
  wss[i] <- sum(kmeans(scale(df1[, -c(1:5)]), centers = i)$withinss)
}

plot(1:10, wss, type = "b", xlab = "Number of groups", ylab = "Within groups sum of squares")
```



We look for elbows in the plot - here there are elbows at 2 and 8, maybe these values will be good to use?

With two clusters, we should see if there is any relationship with sites...

```
tally(Ksol1$cluster ~ type, data = df1, format = "count")
```

```
##                type
## Ksol1$cluster   L   M  SI
##             1  16   0   1
##             2  31   5 137
##             3   1   0   0
```

There does seem to be something of interest here... The first cluster has 30 wells that are "SI," the second

cluster has much more of a mix with 32 “L”, 5 “M”, and 108 “SI” wells. The third cluster has 16 wells that are “L”.

We can compare clustering solutions with similar tables. How do the K-means and Ward’s solutions overlap?

```
tally(Ksol1$cluster ~ wardSol, data = df1, format = "count")
```

```
##              wardSol
## Ksol1$cluster   1   2   3
##             1   0  17   0
##             2 172   1   0
##             3   0   0   1
```

Our solution with k-means puts many wells into cluster 2 which the ward’s solution put into cluster 1...