Clustering

Unsupervised Learning

So far in this class, we've only done **supervised learning**.

Meaning: We have a **response variable** and we observe its value for all or some of our observations.

Clustering is a type of **unsupervised learning**.

We want to sort our observations into **clusters** based on the **predictors**...

... but we don't have a pre-conceived notion of what those clusters represent!

Clustering

The general goal of clustering is typically to make clusters such that points **within** a cluster are closer to each other than to the points **outside** the cluster.

What is our definition of **close**?

How **many** clusters do we think exist?

What **algorithm** do we use to select the clusters?

K-means clustering

Idea: Iteratively update the centers of the clusters until convergence.

1. Plop 3 random points down in space. These are the *centroids*.
2. Determine which centroid each observation is closest to. Assign it to that cluster.
3. Mind the mean of each cluster. These are the *new* centroids.
4. Continue until the centroids don't change.

K-means Clustering

The `kmeans()` function needs to be given a **matrix of data** and a **number of clusters**.

It gives back **centroids**, **cluster assignments**, and **sum-of-squares**.

```
fed_km <- kmeans(fed_matrix, 3)
fed_km
```

```
## K-means clustering with 3 clusters of sizes 39, 5, 26
##
## Cluster means:
##         a        do        is        or      this       all       down      it
## 1 0.02264 0.0004926 0.011389 0.006299 0.007694 0.003692 0.00006095 0.01290
## 2 0.01133 0.0005557 0.006617 0.011255 0.004472 0.002808 0.00000000 0.01659
## 3 0.01991 0.0003927 0.011035 0.006990 0.007490 0.003988 0.00020684 0.01388
##         our      to      also      even      its     shall          up       an
## 1 0.0020956 0.03882 0.0003545 0.0009570 0.003617 0.001507 0.00037792 0.005632
## 2 0.0044983 0.03462 0.0013692 0.0005221 0.002432 0.001175 0.00000000 0.002015
## 3 0.0007316 0.03991 0.0004887 0.0008273 0.003761 0.001594 0.00006386 0.005143
##       every      may   should      upon      and      for.     more       so
## 1 0.0018063 0.004356 0.001963 0.0027342 0.02559 0.006711 0.002886 0.002195
## 2 0.0004988 0.003866 0.002959 0.0001198 0.04885 0.006796 0.005968 0.003247
## 3 0.0015005 0.004252 0.002164 0.0024354 0.02481 0.006119 0.002760 0.001935
##         was      any     from      must      some      were      are      had
```

```
fed_km$centers
```

```
##         a        do        is        or      this       all       down      it
## 1 0.02264 0.0004926 0.011389 0.006299 0.007694 0.003692 0.00006095 0.01290
## 2 0.01133 0.0005557 0.006617 0.011255 0.004472 0.002808 0.00000000 0.01659
## 3 0.01991 0.0003927 0.011035 0.006990 0.007490 0.003988 0.00020684 0.01388
##         our       to      also      even      its    shall         up       an
## 1 0.0020956 0.03882 0.0003545 0.0009570 0.003617 0.001507 0.00037792 0.005632
## 2 0.0044983 0.03462 0.0013692 0.0005221 0.002432 0.001175 0.00000000 0.002015
## 3 0.0007316 0.03991 0.0004887 0.0008273 0.003761 0.001594 0.00006386 0.005143
##       every      may   should      upon      and      for.      more       so
## 1 0.0018063 0.004356 0.001963 0.0027342 0.02559 0.006711 0.002886 0.002195
## 2 0.0004988 0.003866 0.002959 0.0001198 0.04885 0.006796 0.005968 0.003247
## 3 0.0015005 0.004252 0.002164 0.0024354 0.02481 0.006119 0.002760 0.001935
##        was      any      from      must      some      were      are      had
## 1 0.001539 0.003022 0.005779 0.002555 0.0014136 0.001507 0.005327 0.001454
## 2 0.001699 0.002560 0.006205 0.001444 0.0015569 0.001991 0.005777 0.001105
## 3 0.001450 0.002925 0.004930 0.001991 0.0009518 0.001211 0.004672 0.001243
##          my     such     what        as      has        no     than      when
## 1 0.0003375 0.002146 0.001287 0.008645 0.003715 0.002593 0.003044 0.0010076
## 2 0.0003618 0.003601 0.001148 0.011498 0.001965 0.001162 0.004317 0.0016872
## 3 0.0001322 0.002073 0.001191 0.008948 0.002657 0.002629 0.002846 0.0008404
##         at      have      not     that    which      be       her       now
## 1 0.003359 0.007324 0.006319 0.01493 0.011130 0.01995 0.0008013 0.0004449
## 2 0.002607 0.005895 0.007625 0.01751 0.006758 0.01880 0.0010409 0.0004457
## 3 0.002745 0.006013 0.006159 0.01470 0.010791 0.02168 0.0001255 0.0004326
##       the      who     been       his      of    their     will      but
## 1 0.08707 0.002502 0.004712 0.0021895 0.06232 0.005920 0.006149 0.003631
```

```
fed_km$totss
```

```
## [1] 0.0512
```

```
fed_km$withinss
```

```
## [1] 0.017125 0.002869 0.013263
```

```
fed_km$betweenss
```

```
## [1] 0.01794
```

```
res <- tibble(
  clust = fed_km$cluster,
  auth = fed_ex$auths)

res
```

```
## # A tibble: 70 x 2
##    clust auth
##    <int> <chr>
##  1     1 AH
##  2     2 JJ
##  3     2 JJ
##  4     2 JJ
##  5     2 JJ
##  6     1 AH
##  7     1 AH
##  8     1 AH
##  9     1 AH
## 10     1 JM
## # ... with 60 more rows
```

```
res %>% count(clust, auth)
```

```
## # A tibble: 5 x 3
##   clust auth      n
##   <int> <chr> <int>
## 1     1 AH       31
## 2     1 JM        8
## 3     2 JJ        5
## 4     3 AH       20
## 5     3 JM        6
```

K-means + PCA

Did we really need all 200 variables to find those clusters?

Did we maybe "muddy the waters" by weighing all variables equally?

It is very common to do a PCA reduction before running k-means!

```
pc <- prcomp(fed_matrix, center = TRUE, scale = TRUE)

fed_reduced <- pc$x[, 1:2]

fed_pca_km <- kmeans(fed_reduced, 3)
```

```
res <- tibble(
  clust = fed_pca_km$cluster,
  auth = fed_ex$auths)

res %>% count(clust, auth)
```

```
## # A tibble: 5 x 3
##   clust auth      n
##   <int> <chr> <int>
## 1     1 AH       43
## 2     2 AH        8
## 3     2 JJ        1
## 4     2 JM       14
## 5     3 JJ        4
```

What if we'd done four centroids?

```
pc <- prcomp(fed_matrix, center = TRUE, scale = TRUE)

fed_reduced <- pc$x[, 1:2]

fed_pca_km <- kmeans(fed_reduced, 4)

res <- tibble(
  clust = fed_pca_km$cluster,
  auth = fed_ex$auths)

res %>% count(clust, auth)
```

```
## # A tibble: 7 x 3
##   clust auth      n
##   <int> <chr> <int>
## 1     1 JJ        4
## 2     2 AH        3
## 3     2 JJ        1
## 4     2 JM       13
## 5     3 AH       22
## 6     4 AH       26
## 7     4 JM        1
```

K-means

Pros:

- Simple algorithm, easy to understand
- Plays nice with PCA
- SUPER fast to compute

Cons:

- Very sensitive to the random locations of initial centroids
- The user has to pick how many clusters.

Try it!

Open a Activity-Clustering.Rmd

Apply k-means clustering to the cannabis data using *all* the word predictors.

What was the within and between sum of squares?

Did the clusters match up with the Type?

Refer back to your PCA analysis of the cannabis data.

Apply k-means clustering to the second and third PC only

Plot these clusters. What do you think they capture?

Hierarchical Clustering

(also called **agglomerative** clustering)

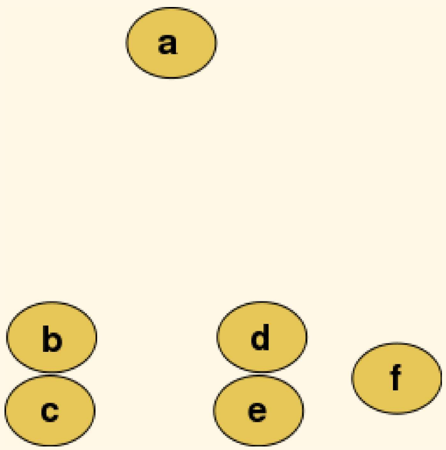
Idea: Merge observations that are close together.

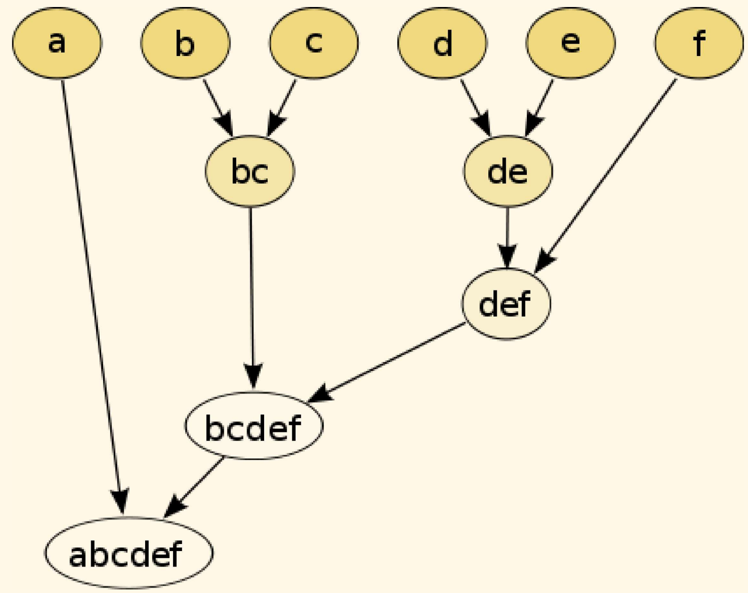
1. Find the closest two observations. Replace them with their centroid.

2. Find the next two closest observations. (One might be the centroid!) Replace them with their centroid.

3. Continue until all observations have been merged.

Hierarchical clustering





Hierarchical clustering

The `hclust` function needs to be given a *distance matrix*.

```
fed_hc <- fed_matrix %>% hclust()
```

```
## Error in if (is.na(n) || n > 65536L) stop("size cannot be NA nor exceed 65536"): missing value where TRUE/FALSE n
```

19 / 28

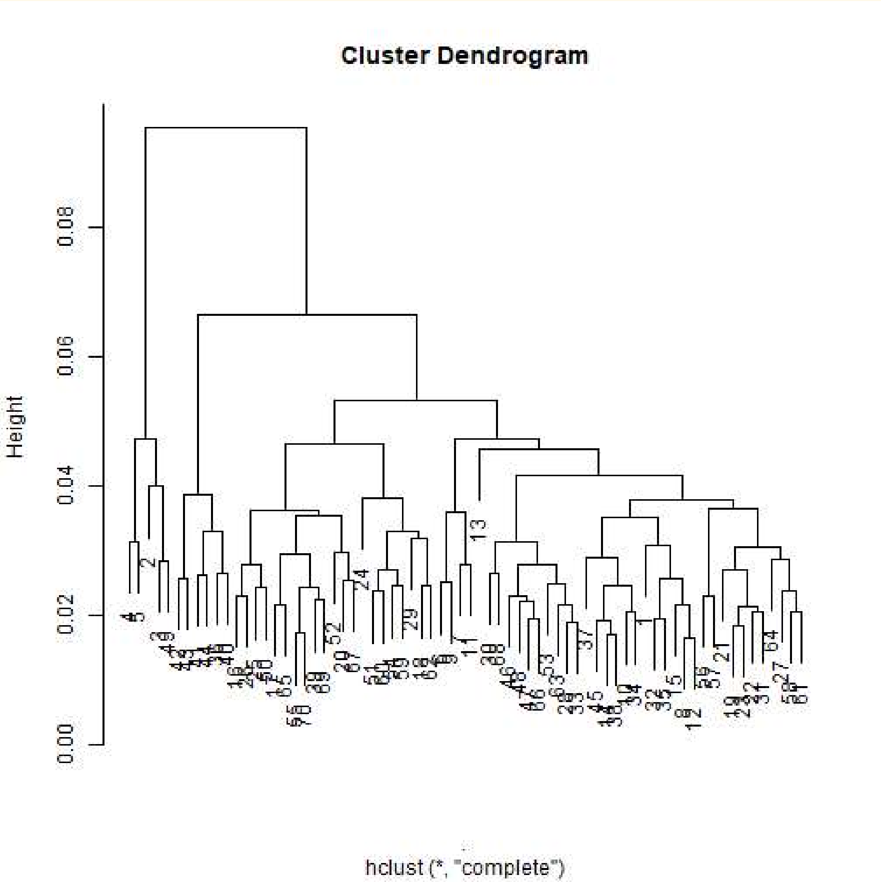
Hierarchical clustering

The `hclust` function needs to be given a *distance matrix*.

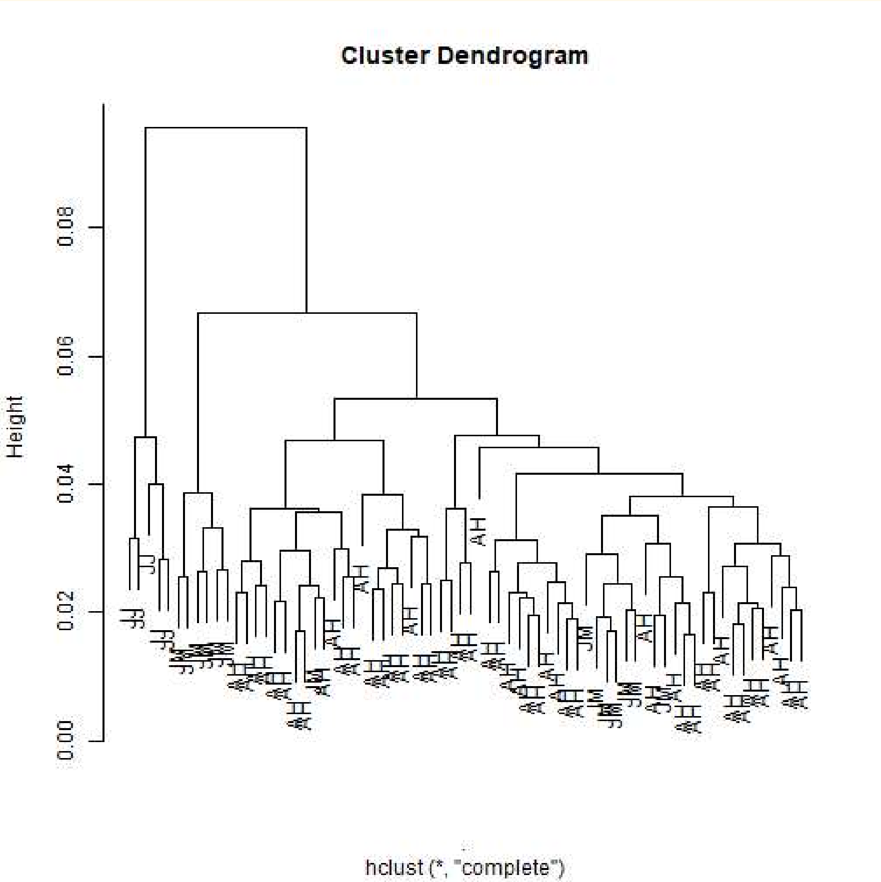
It gives back a **dendrogram**.

```
fed_hc <- fed_matrix %>% dist() %>% hclust()
```

```
plot(fed_hc)
```


Cluster Dendrogram
Height
0.00
0.02
0.04
0.06
0.08
hclust (*, "complete")

```
plot(fed_hc, labels = fed_ex$auths)
```



Dendrograms

To decide how to assign clusters, we can:

1. Choose how many clusters we want...

```
res_hc <- cutree(fed_hc, k = 3)

res_hc
```

```
##  [1] 1 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 1 1 1 1 1 1 1 1 3 1 1
## [39] 1 3 3 3 3 3 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
tibble(
  clust = res_hc,
  auth = fed_ex$auths
) %>%
  count(clust, auth)
```

```
## # A tibble: 4 x 3
##   clust auth      n
##   <int> <chr> <int>
## 1     1 AH       51
## 2     1 JM        8
## 3     2 JJ        5
## 4     3 JM        6
```

Dendrograms

To decide how to assign clusters, we can:

1. Choose a **height cutoff** for the dendrogram

```
res_hc_2 <- cutree(fed_hc, h = 0.05)
res_hc_2
```

```
##  [1] 1 2 2 2 2 1 1 1 1 1 1 1 1 1 1 3 3 3 1 3 1 1 1 3 3 3 1 1 3 1 1 1 1 1 1 4 1 1
## [39] 3 4 4 4 4 4 1 1 1 1 2 3 3 3 1 3 3 1 1 1 3 3 1 3 1 1 3 1 3 1 3 3
```

```
tibble(
  clust = res_hc_2,
  auth = fed_ex$auths
) %>%
  count(clust, auth)
```

```
## # A tibble: 6 x 3
##   clust auth      n
##   <int> <chr> <int>
## 1     1 AH       31
## 2     1 JM        7
## 3     2 JJ        5
## 4     3 AH       20
## 5     3 JM        1
## 6     4 JM        6
```

Hierarchical Clustering

Pros:

- Fast computation for moderate sized data
- Gives back full information in dendrogram form

Cons:

- User has to decide how to go from dendrogram to cluster assignments

Try it!

Open Activity-Clustering.Rmd

Apply hierarchical clustering to the cannabis data

Compare your results to k-means. Which do you prefer? Why?