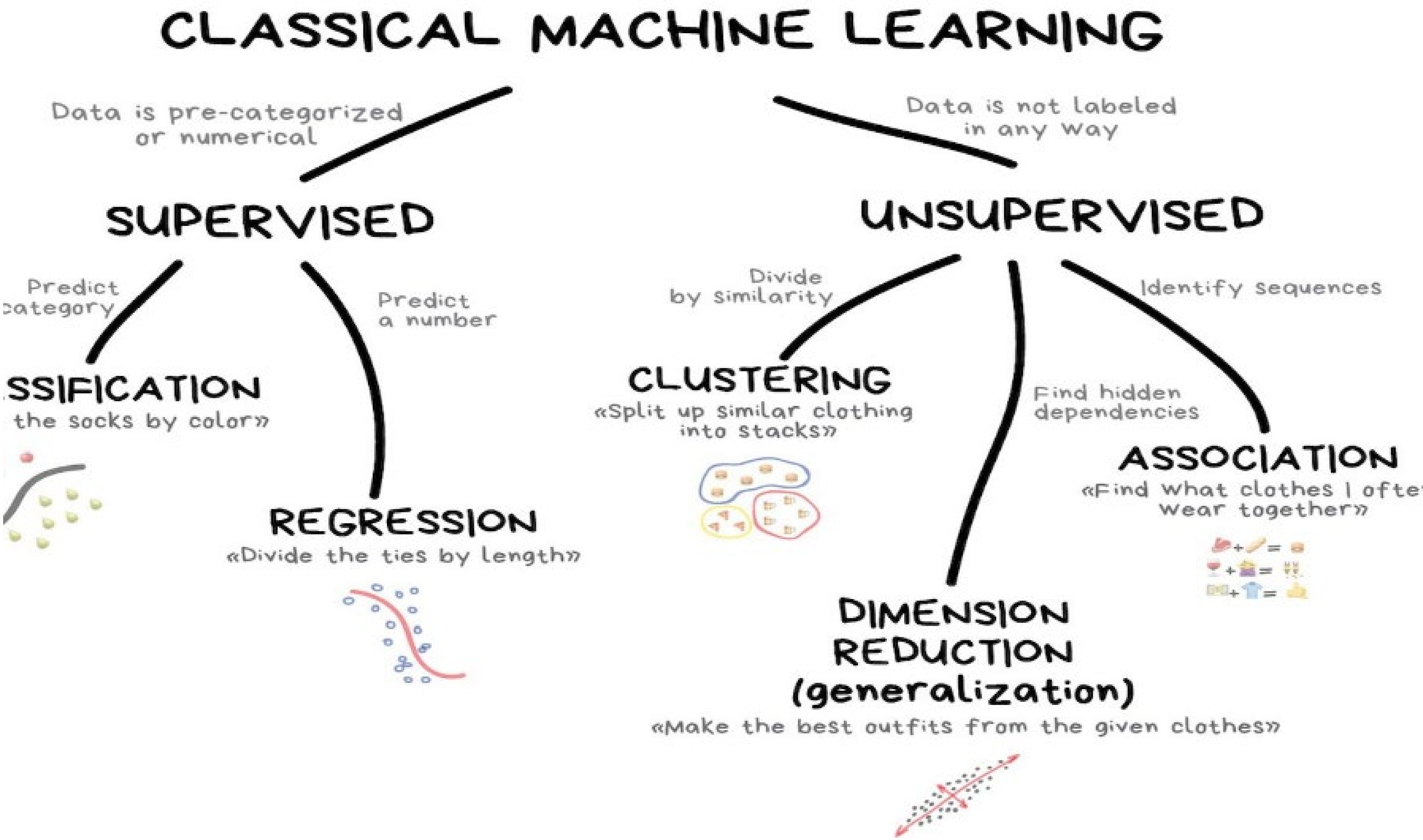
Machine Learning Overview, EDA and Clustering

Brian Wright

brianwright@virginia.edu



CLASSICAL MACHINE LEARNING
Data is pre-categorized or numerical
Data is not labeled in any way
SUPERVISED
UNSUPERVISED
Predict category
Predict a number
Divide by similarity
Identify sequences
SSIFICATION
the socks by color»
CLUSTERING
«Split up similar clothing into stacks»
Find hidden dependencies
ASSOCIATION
«Find what clothes I often wear together»
REGRESSION
«Divide the ties by length»
DIMENSION REDUCTION (generalization)
«Make the best outfits from the given clothes»

Example of K-Means

- Given *D* :{2,4,10,12,3,11,20,25,30}, and k=2 clusters
- Randomly assign the means: m_1=3, m_2=4
- K_1={2,3}, K_2={4,10,12,11,20,25,30}, m_1=2.5, m_2=16
- K_1={2,3,4}, K_2={10,12,11,20,25,30}, m_1=3, m_2=18
- K_1={2,3,4,10}, K_2={12,11,20,25,30}, m_1=4.75, m_2=19.6
- K_1={2,3,4,10,11,12}, K_2={20,25,30}, m_1=7, m_2=25
- K_1={2,3,4,10,11,12}, K_2={20,25,30}, m_1=7, m_2=25
    - **Stop**, since the clusters and the means found in all subsequent iterations will be the same.

Outline: Intro to Unsupervised ML

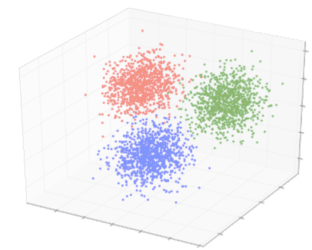
1. What is Machine Learning?
2. What is exploratory data analysis?
3. k-means clustering
   – Does Congress vote in patterns?
4. Multi-dimensional k-means clustering
   – Are NBA players compensated according to performance?

What is exploratory data analysis?

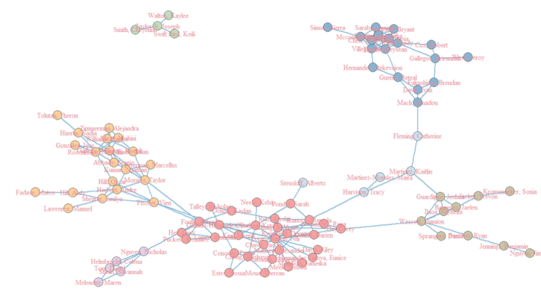
- Exploratory data analysis or “EDA” is an approach where the intent is to see what the data can tell us beyond modeling or hypothesis testing
  - Data visualization is one of the most common forms of EDA

Types of exploratory data analysis

When data is too big or complex to be analyzed just by visualizing it, these types of analysis can help:



1. Clustering: compare pieces of data by measuring similarity among them

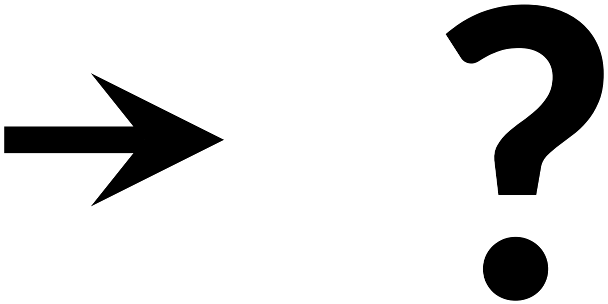


2. Network analysis: analyze how people, places and entities are connected to evaluate the properties and structure of a network
3. Text mining: analyze what large bodies of unstructured or structured text say

Unsupervised machine learning

The data inputs have ***(x)*** no target outputs ***(y)***

Input *x*:
Voter



Output *y*:
Not given
(To be discovered)

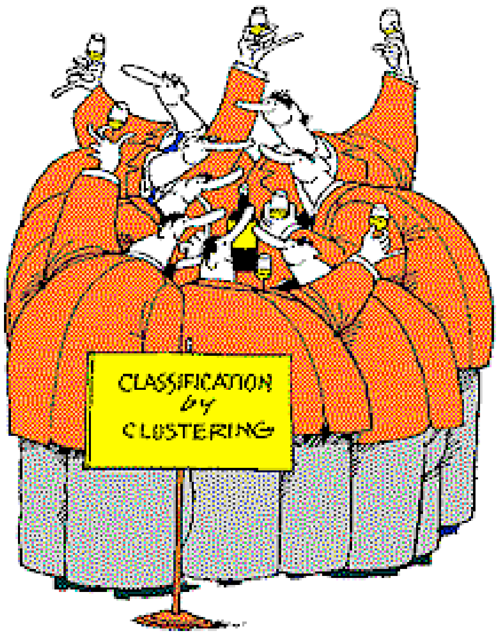
We want to impose structure on the inputs *(x)* to say something meaningful about the data

What is clustering?

1. Technique for finding similarity between groups

2. Type of unsupervised machine learning
   - Not the only class of unsupervised learning algorithms

3. Similarity needs to be defined
   - Will depend on attributes of data
   - Usually a distance metric



Key assumption: data points that are "closer" together are related or similar

GE Capital case study: grouping clients

- Haimowitz and Schwarz 1997 paper on clustering for credit line optimization
  – http://www.aaai.org/Papers/Workshops/1997/WS-97-07/WS97-07-006.pdf
- Cluster existing GE Capital customers based on similarity in use of their credit cards to pay bills and customers' profitability to GE Capital
- Resulted in five clusters of consumer credit behavior
- Created classification model to predict customer type and offer tailored products

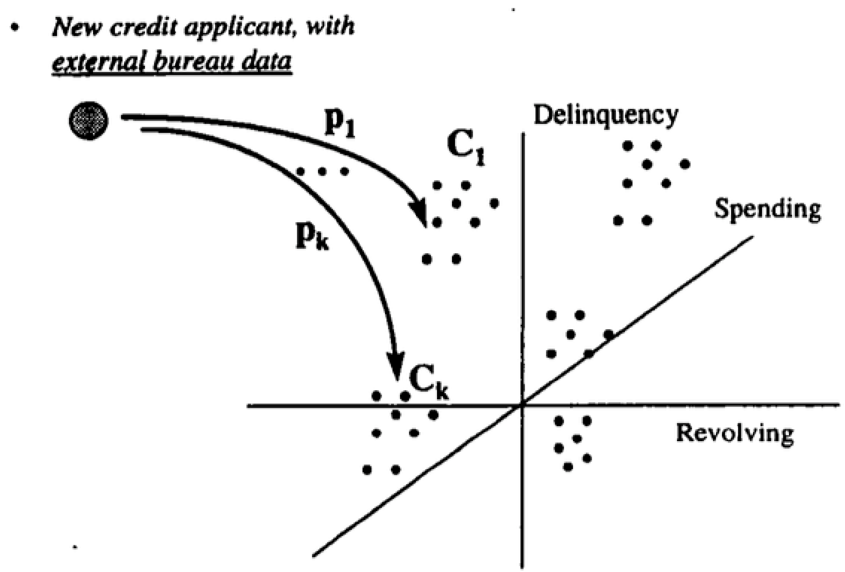


Figure 1: Clustering-based framework for optimizing a credit policy.

Concept summary

Example use case	General question	Concept
Does Congress vote in patterns?	Is there a pattern? Is there structure in unstructured data?	k-means clustering
Are basketball players "priced" efficiently (based on performance)?	How to uncover trends with many variables that you can't easily visualize?	k-means clustering in many dimensions

Political clustering

Goal: to understand how polarized the US Congress is

1. Data set consists of 427 members (observations)
2. Members served a full year in 2013
3. Three vote types:
   - "Aye"
   - "Nay"
   - "Other"

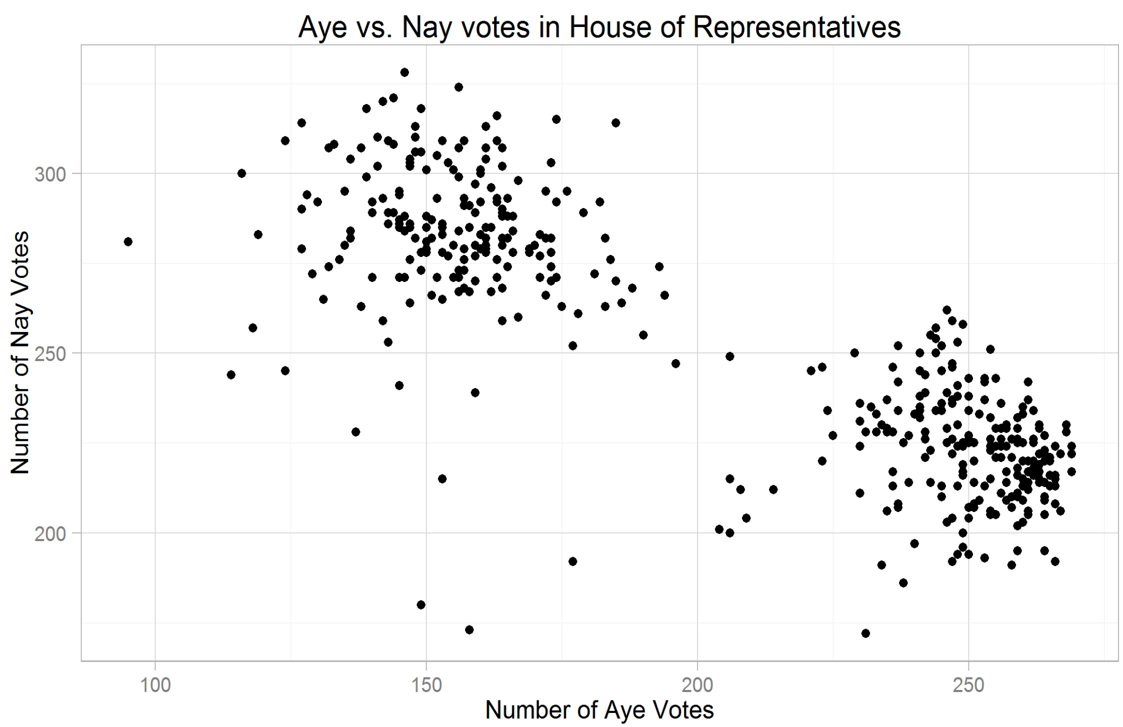
The joint session of Congress on Capitol Hill in Washington

Finding voting patterns

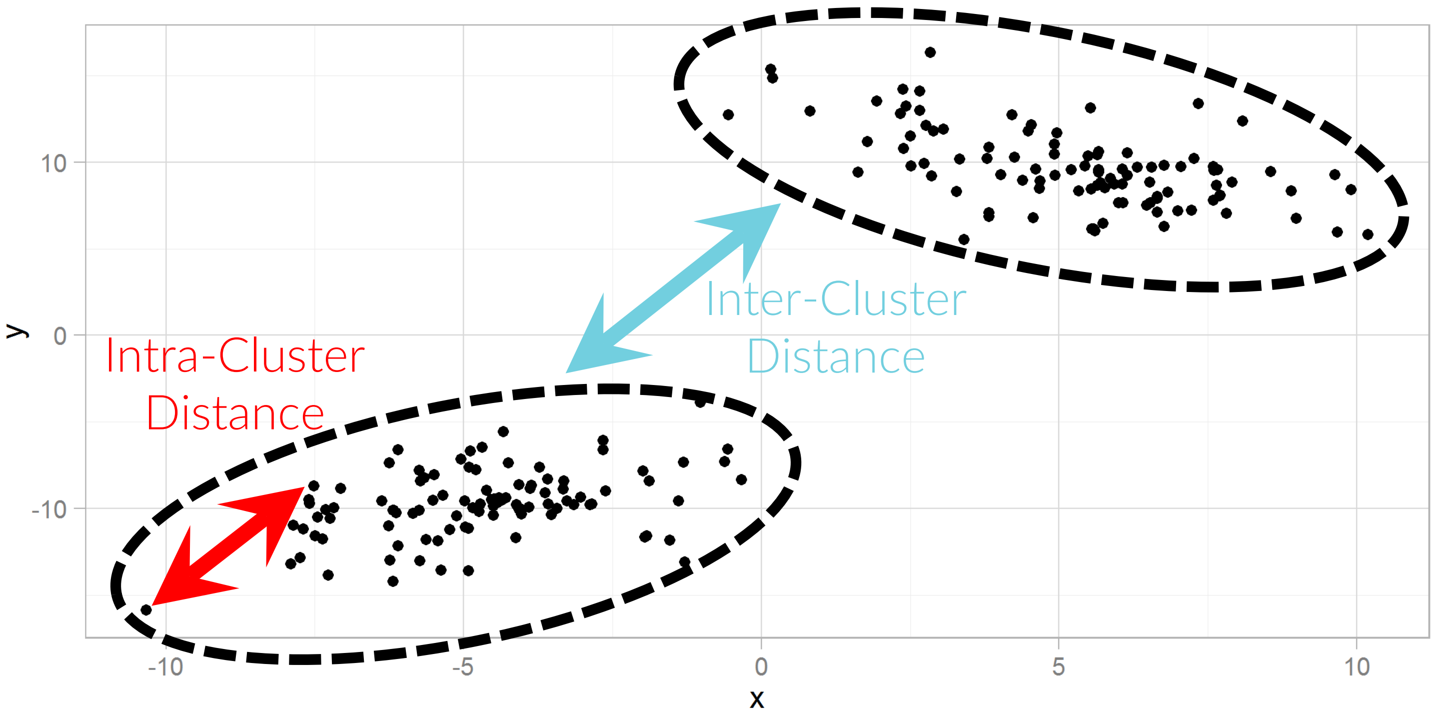
Each data point represents a member of Congress

- How do we identify swing votes?
  - Lobbying
  - Bridging party lines

- Assumption:
  - Democrats and Republicans vote among partisan lines, which generates clusters



Intra vs. inter-cluster distance

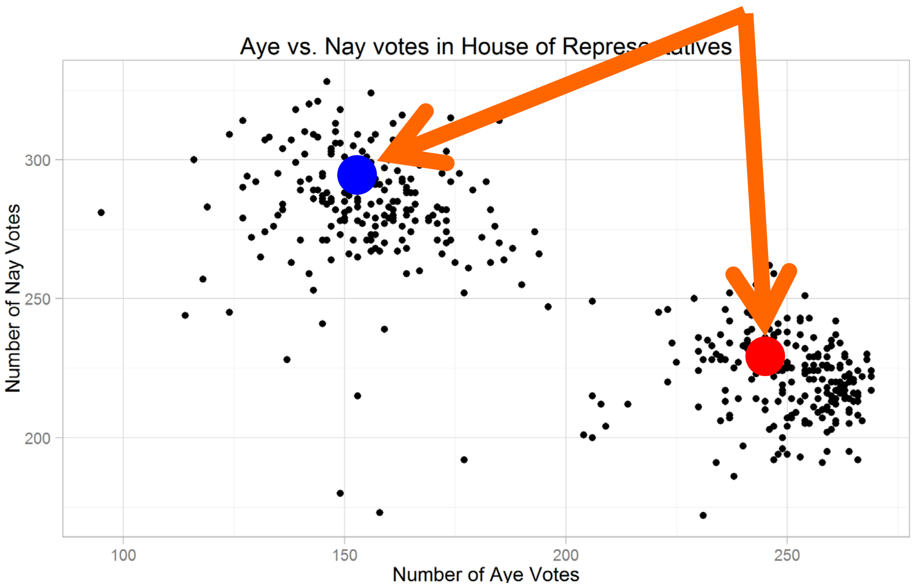


Objective: minimize intra-cluster distance, maximize inter-cluster distance

k-means clustering is based on centroids

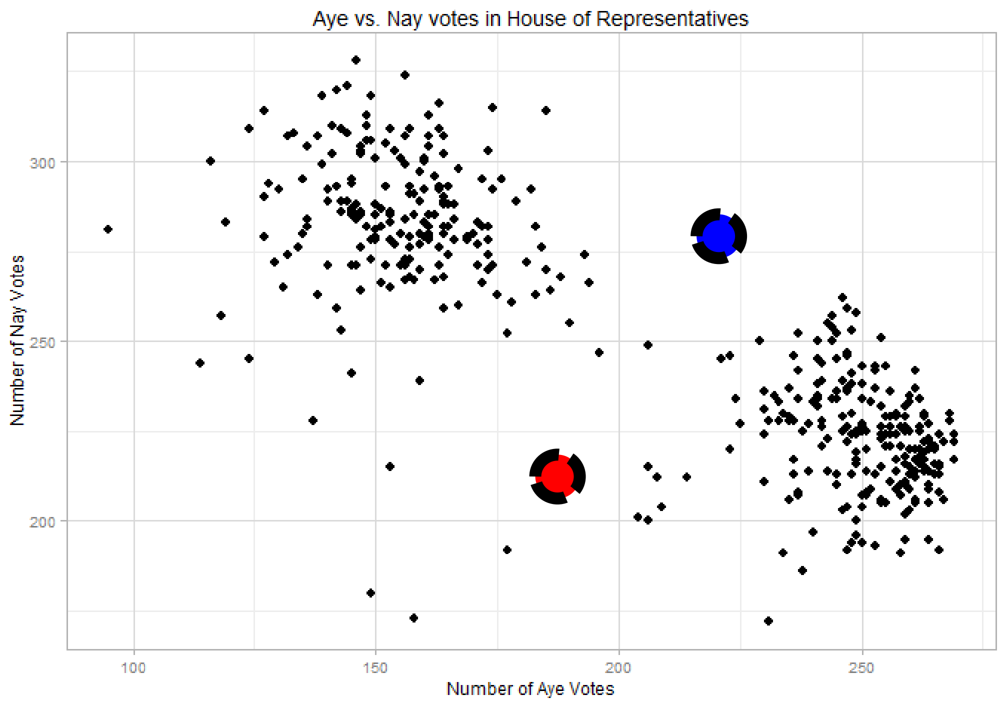
- The centroid is the average location of all points in the cluster
- Another definition: the centroid minimizes the distance between a central location and all the data points in the cluster

Note: Centroids are generally not existing data points, rather locations in space



k-means in 4 steps

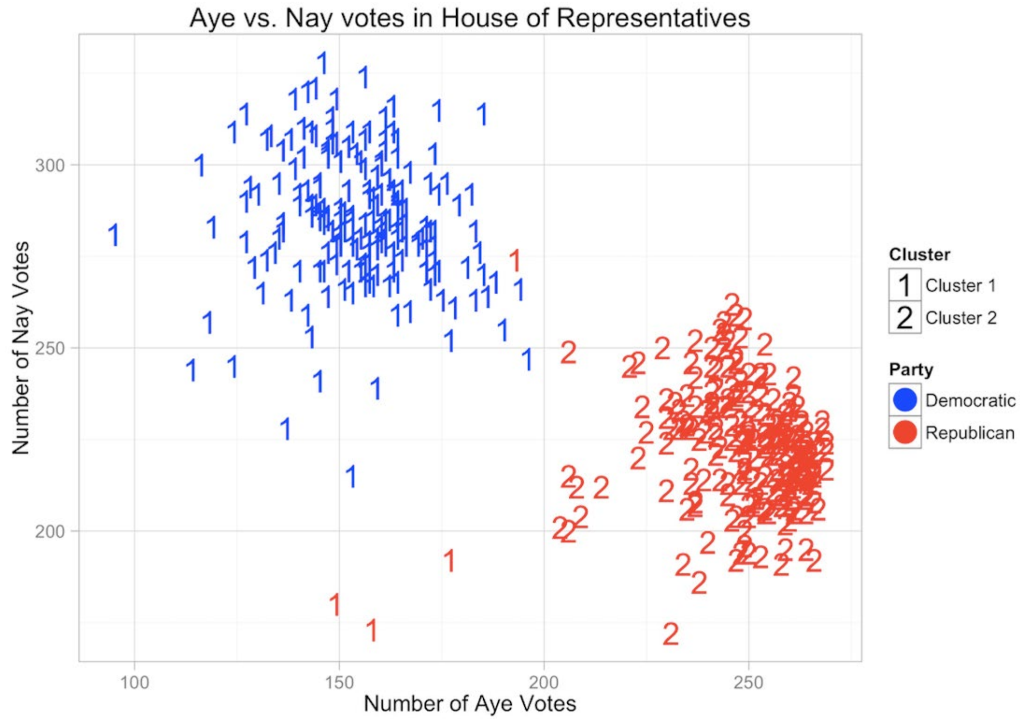
1. Randomly choose k data poi
   to be centroids



k-means in 4 steps

1. Randomly choose k data points to be centroids

2. Assign each point to closest centroid

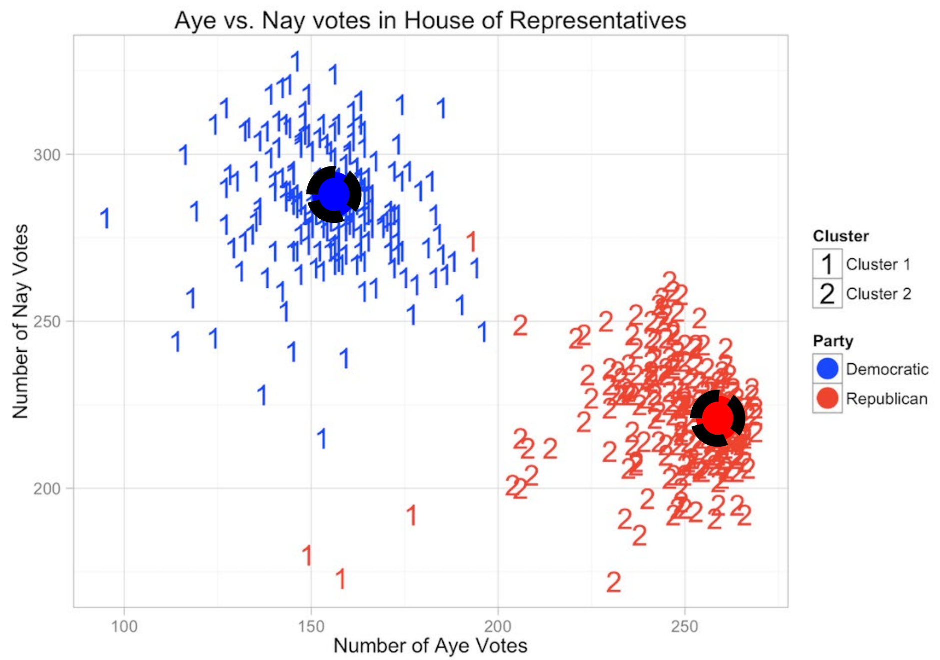


k-means in 4 steps

1. Randomly choose k data points to be centroids

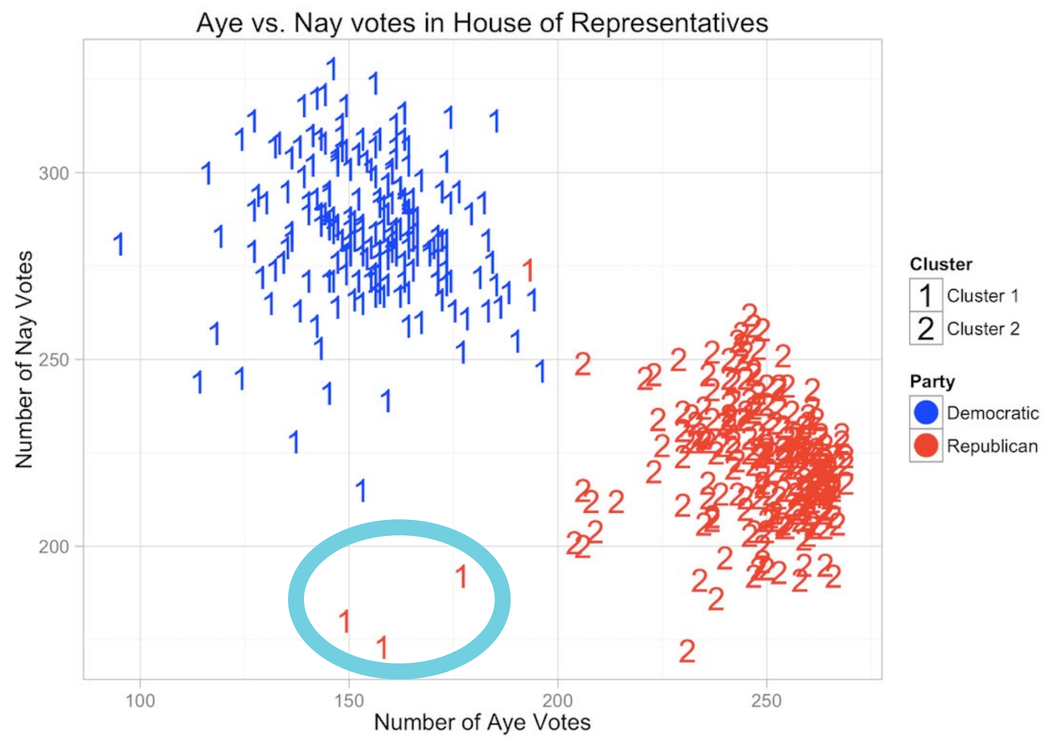
2. Assign each point to closest centroid

3. Recalculate centroids based on current cluster membership



k-means in 4 steps

1. Randomly choose k data points to be centroids
2. Assign each point to closest centroid
3. Recalculate centroids based on current cluster membership

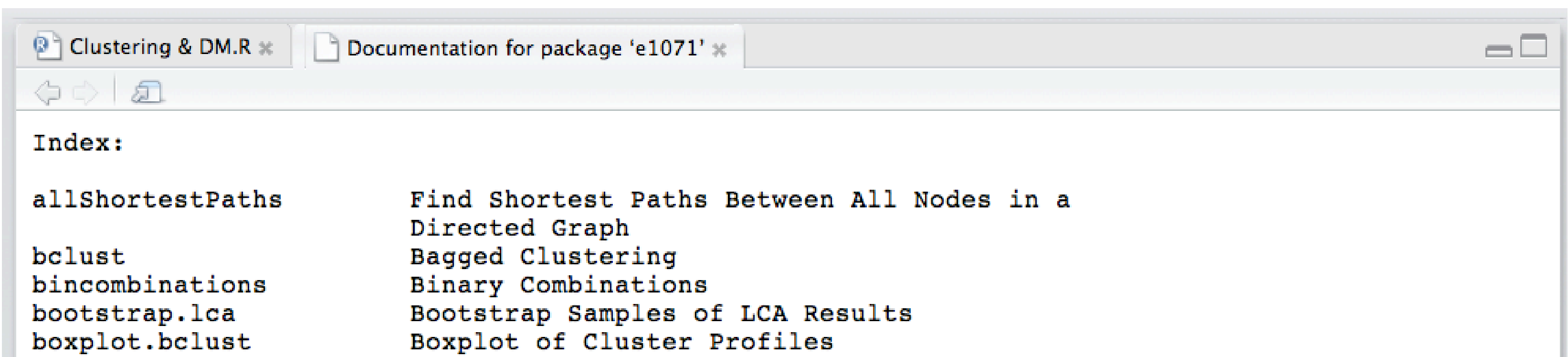


4. Repeat steps 2-3 with the new centroids until the centroids don't change anymore

Step 1: load packages and data

```r
# Install packages
install.packages("e1071")
install.packages("ggplot2")

# Load libraries
library(e1071)
library(ggplot2)

library(help = e1071)
```

Learn about all the functionality of the package, be well informed about what you're doing!




```
Index:

allShortestPaths        Find Shortest Paths Between All Nodes in a
                        Directed Graph
bclust                  Bagged Clustering
bincombinations         Binary Combinations
bootstrap.lca           Bootstrap Samples of LCA Results
boxplot.bclust          Boxplot of Cluster Profiles
```

Step 1: load packages and data

Script

```
# Loading house data

house_votes_Dem = read_csv("house_votes_Dem.csv")

# What does the data look like?
View(house_votes_Dem)
```

Clustering & DM.R | house_votes_Dem

Filter

	Last.Name	party.labels	yea	nay	other
1	Courtney	Democrat	66	163	91
2	Lewis	Democrat	59	145	116
3	Bera	Democrat	84	141	95
4	McCollum	Democrat	74	154	92
5	Olson	Republican	127	103	90

Step 2: run k-means

Script

```
# Define the columns to be clustered by subsetting the data
clust_data_Dem = house_votes_Dem[, c("aye", "nay", "other")]


# Run an algorithm with 2 centers
set.seed(1)

kmeans_obj_Dem = kmeans(clust_data_Dem, centers = 2,
                        algorithm = "Lloyd")

# What does the new variable kmeans_obj contain?
kmeans_obj_Dem

# View the results of each output of the kmeans
# function
head(kmeans_obj_Dem)
```

1. By placing the set of data we want after the comma, we tell R we're looking for columns
2. kmeans uses a different starting data point each time it runs. To make the results reproducible make R start from the same point every time with `set.seed()`
3. We're not specifying the number of iterations so R defaults to 10
4. We'll see that kmeans produces a list of vectors of different lengths. As a result, we cannot use the `View()` function

Step 2: run k-means

```
Console ~/Desktop/
> kmeans_obj_Dem
K-means clustering with 2 clusters of sizes 202, 225

Cluster means:
        yea      nay     other
1  70.32673 145.6337 104.03960
2 122.56889 106.9956  90.43556

Clustering vector:
  [1] 1 1 1 1 2 2 2 1 2 1 2 2 2 2 1 2 1 1 2 2 2 2 1 1 2 1 2 1 2 1 1 1 1 2 2 1 2 2 1 2 1 1 1 1 1 2 1 1 2 1 1 1 1 1 2 2 1 2 1 1 1 1 1 2 2 2 2 1 1 2 1 1 1 1 1 2 2 1 2 2 2
 [82] 1 2 2 2 2 1 1 1 1 2 2 2 1 1 1 2 1 1 1 1 2 2 1 2 1 2 2 2 2 2 2 1 1 1 2 2 1 2 2 1 1 2 2 2 2 2 2 1 1 1 1 1 2 1 2 1 2 2 1 2 1 1 1 1 1 2 2 1 1 2 2 2 1 2 1 2 1 2 1 2 2
[163] 1 2 1 2 1 2 2 2 1 1 2 1 1 1 1 2 2 2 2 1 2 1 1 2 2 2 2 1 2 1 2 1 1 1 2 2 1 1 2 1 1 1 2 1 2 2 2 1 2 2 2 2 1 1 2 2 1 2 1 2 1 1 1 2 2 2 2 2 2 1 1 2 2 2 1 1 2 1 1 1 1
[244] 2 2 2 2 2 1 1 1 1 2 2 1 1 1 2 2 2 1 2 1 2 1 2 1 2 1 2 2 2 2 1 2 2 2 1 1 1 1 1 1 1 2 2 1 1 2 2 2 2 2 1 1 1 1 2 1 2 2 2 2 1 2 1 1 2 1 1 2 2 2 1 1 2 1 2 2 1 2 2 1 1
[325] 2 2 1 1 2 2 2 1 2 1 2 2 2 1 1 1 2 2 2 2 2 2 2 1 2 2 1 1 1 2 2 2 1 2 2 2 2 1 2 1 2 2 1 1 1 1 2 2 2 1 2 1 1 1 2 1 2 2 1 2 2 1 2 1 2 2 2 2 2 1 2 1 1 2 1 1 2 2 2 2 2
[406] 1 2 2 1 1 2 1 2 2 2 2 2 1 2 1 2 1 2 1 1 2 1

Within cluster sum of squares by cluster:
[1] 77671.01 43093.49
 (between_SS / total_SS =  79.5 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"         "iter"         "ifault"
>
```

1. Number of points each cluster contains
2. The “location” of each cluster center is specified by 3 coordinates, one for each column we’re clustering
3. The list assigning either cluster 1 or 2 to each data point

1. 79.5% of the variance between the data points is explained by our clustering, we will discuss this in detail later
2. List of other types of data included in `kmeans_obj`

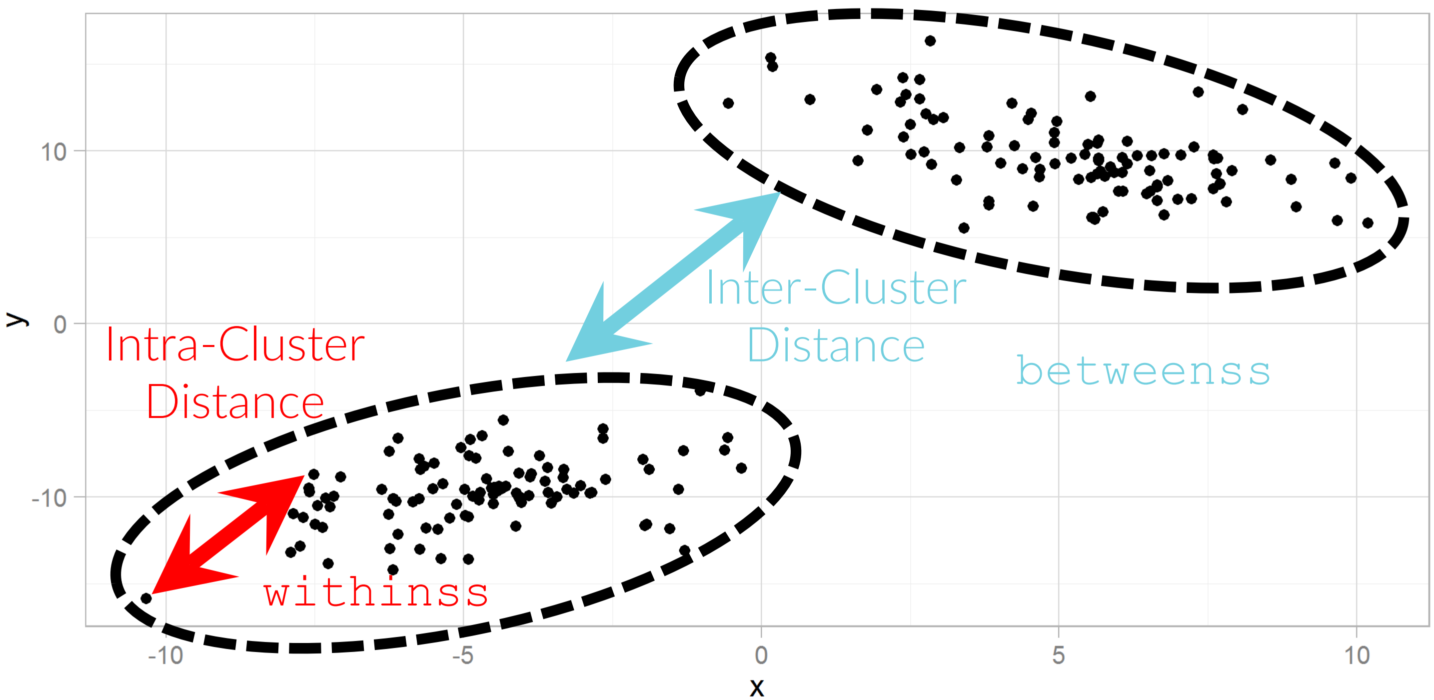
kmeans outputs

- cluster: a vector indicating the cluster to which each point is allocated
- centers: a matrix of cluster centers
- totss: the total sum of squares (sum of distances between all points)
- withinss: vector of within-cluster sum of distances, one number per cluster
- tot.withinss: total within-cluster sum of distances, i.e. sum of withinss
- betweenss: the between-cluster sum of squares, i.e. totss - tot.withinss
- size: the number of points in each cluster

To learn more about the kmeans function run ?kmeans

Intra vs. inter-cluster distance

`totss = withinss + betweenss`



Step 3: visualize plot

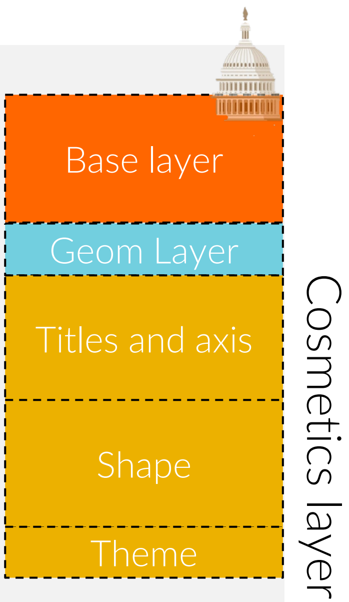
Script

```
# Tell R to read the cluster labels as factors so that ggplot2 (the
# graphing  package) can read them as category labels instead of
# continuous variables (numeric variables).
party_clusters_Dem = as.factor(kmeans_obj_Dem$cluster)

# What does party_clusters look like?
View(party_clusters_Dem)
View(as.data.frame(party_clusters_Dem))

# Set up labels for our data so that we can compare Democrats and
# Republicans.
party_labels_Dem = house_votes_Dem$party
```

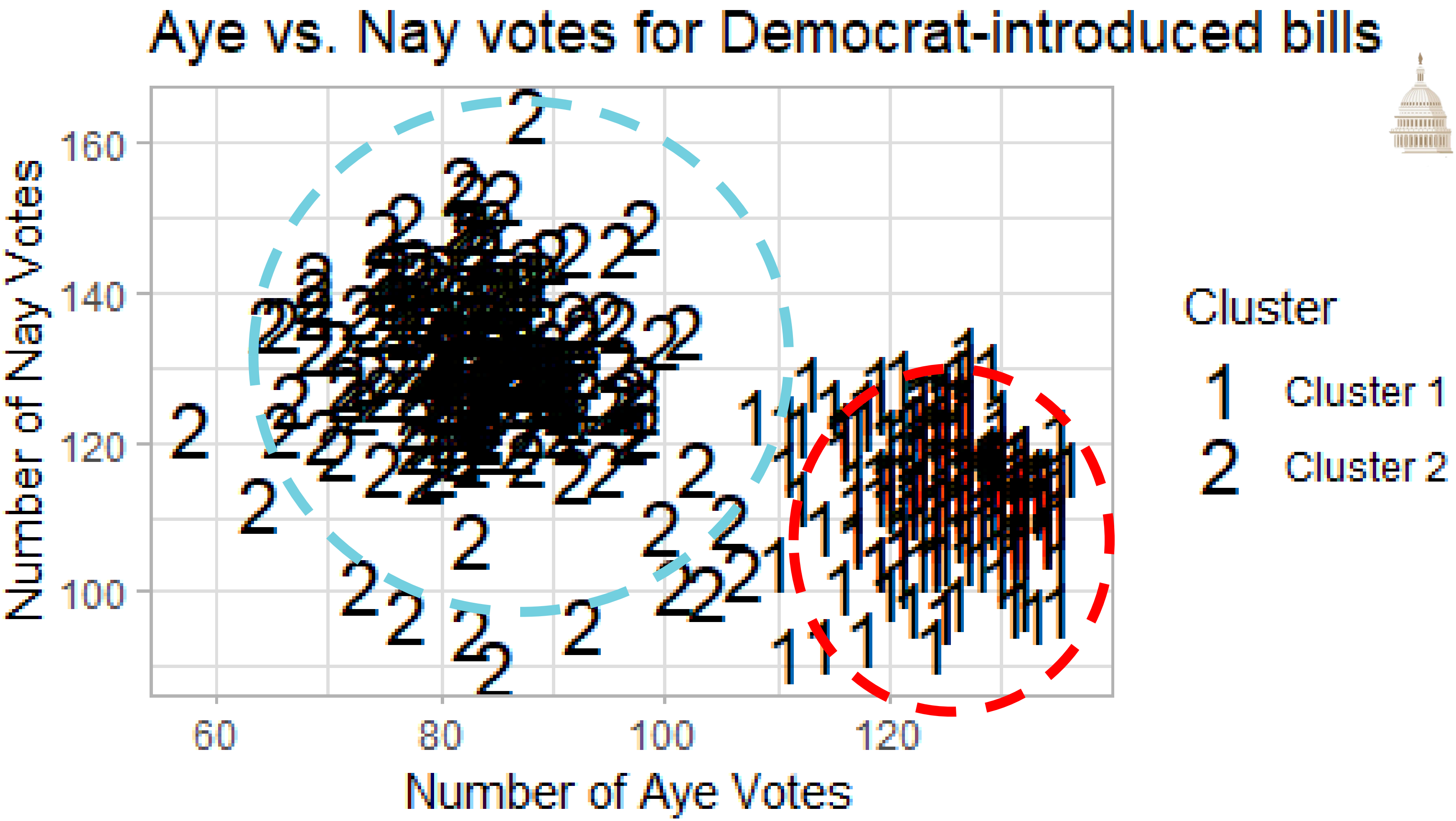
Step 3: visualize plot

Script

```
ggplot(house_votes_Dem, aes(x = aye,
                            y = nay,
                            shape = party_clusters_Dem)) +
  geom_point(size = 6) +
  ggtitle("Aye vs. Nay votes for Democrat-introduced bills") +
  xlab("Number of Aye Votes") +
  ylab("Number of Nay Votes") +
  scale_shape_manual(name = "Cluster",
                     labels = c("Cluster 1", "Cluster 2"),
                     values = c("1", "2")) +
  theme_light()
```



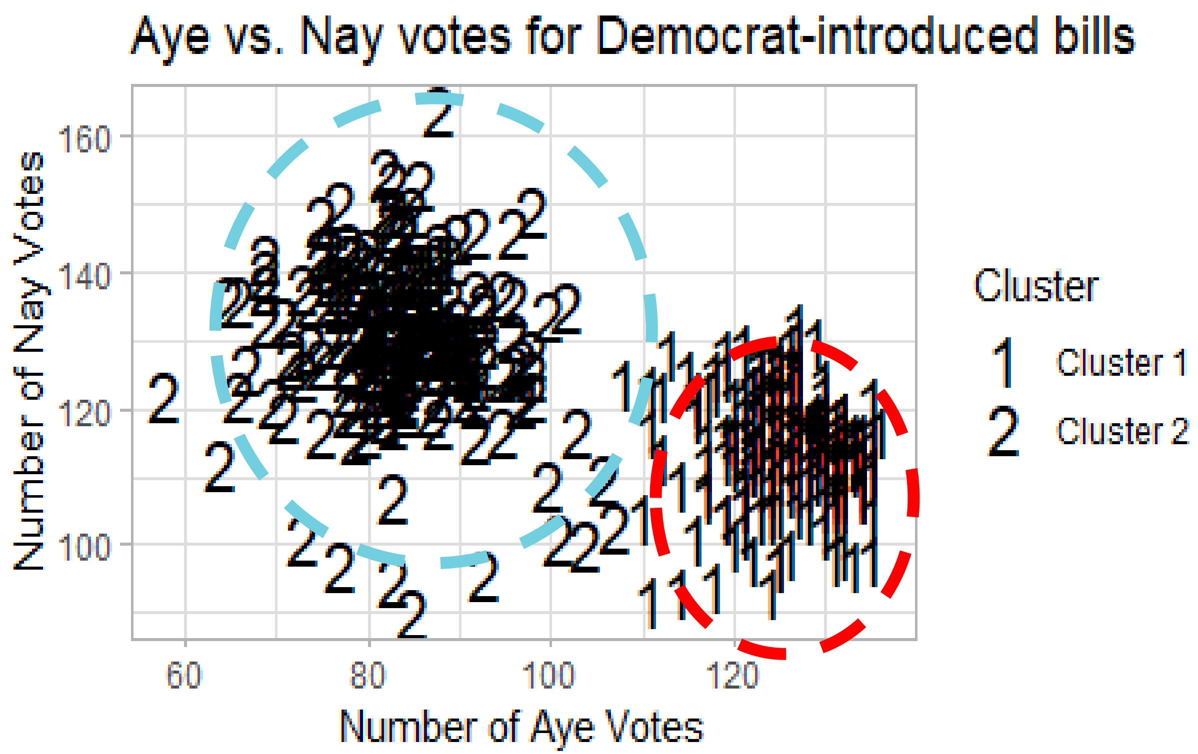
Step 3: visualize plot



Step 4: analyze results

What can we infer about the different clusters?

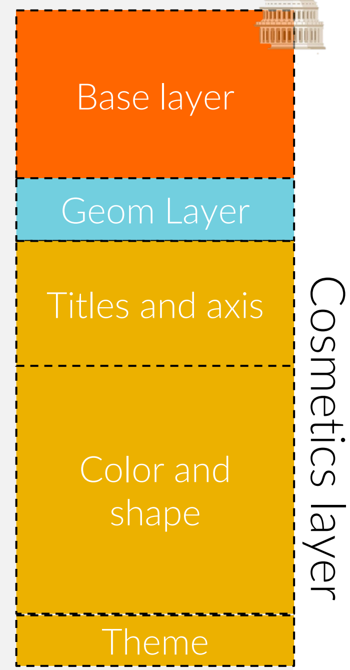
- Two groups exist
- Algorithm identifies voting patterns



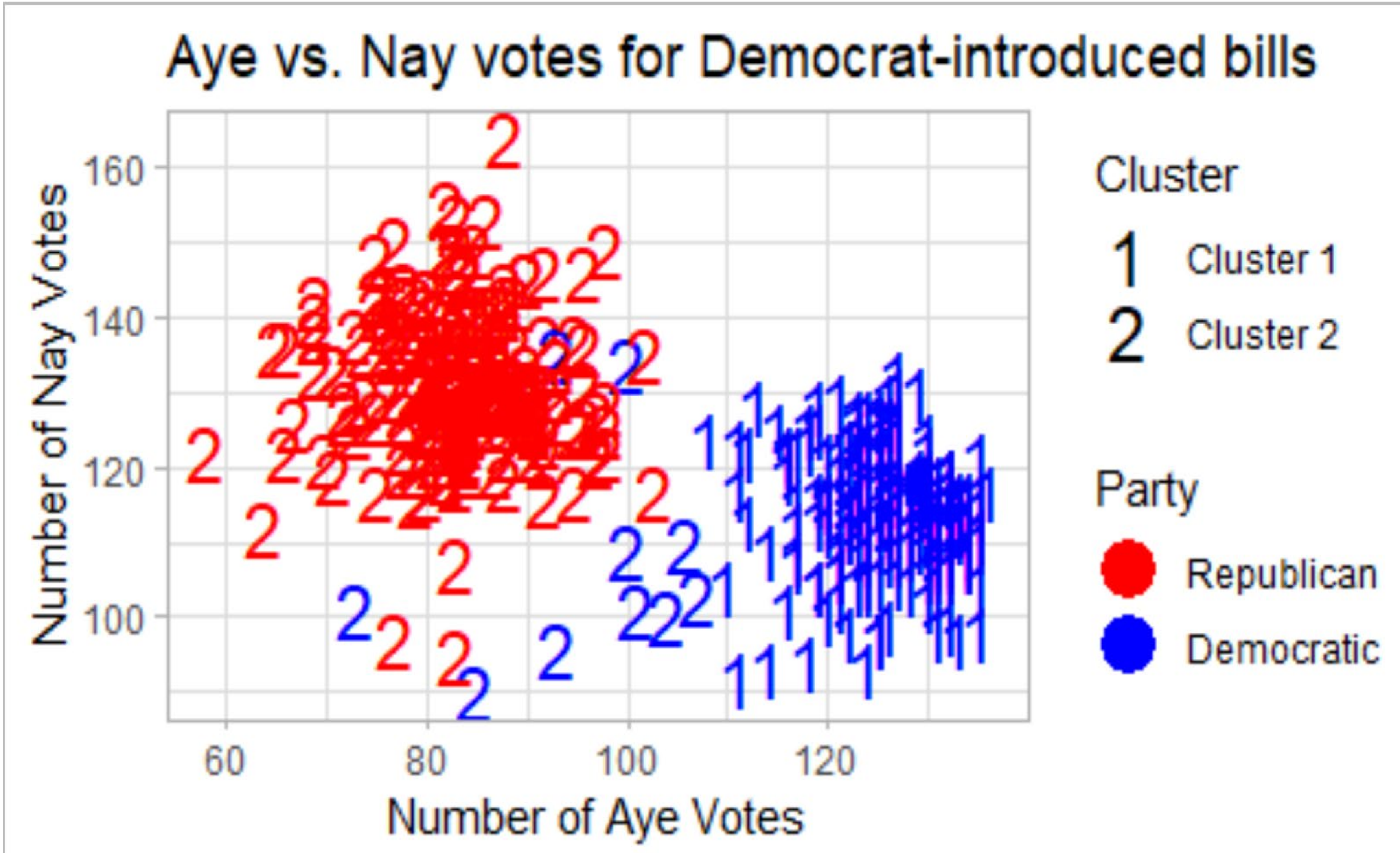
Step 5: validate results

Script

```
ggplot(house_votes_Dem, aes(x = yea,
                            y = nay,
                            color = party_labels_Dem,
                            shape = party_clusters_Dem)) +
  geom_point(size = 6) +
  ggtitle("Aye vs. Nay votes for Democrat-introduced bills")
  xlab("Number of Aye Votes") +
  ylab("Number of Nay Votes") +
  scale_shape_manual(name = "Cluster",
                     labels = c("Cluster 1", "Cluster 2"),
                     values = c("1", "2")) +
  scale_color_manual(name = "Party",
                     labels = c("Democratic", "Republican"),
                     values = c("blue", "red")) +
  theme_light()
```

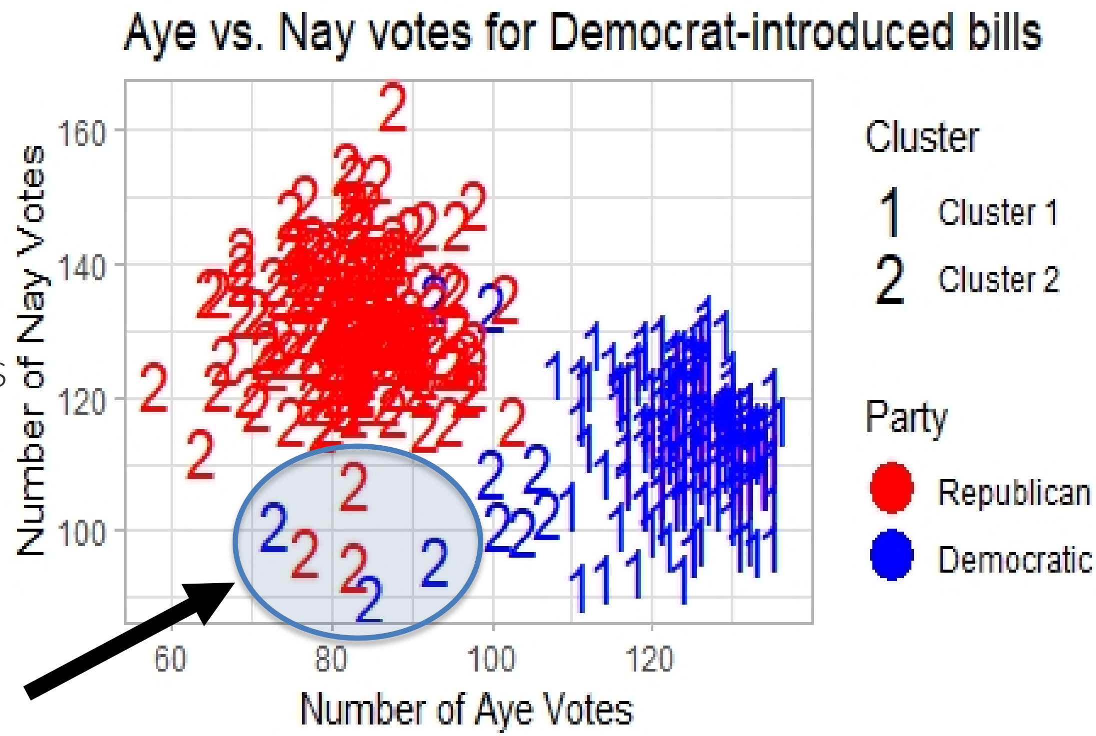


Step 5: validate results



Step 6: interpret results

- Diffuse among Democrats
- Republicans more dense
- Can gauge “outliers”
- Can see the polarization between the two political parties

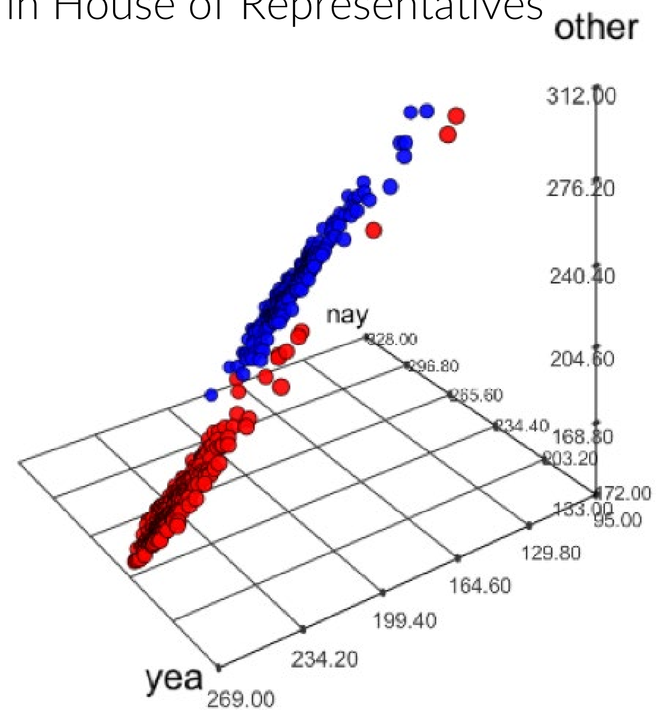


Outliers?

Clustering vs. visualizing

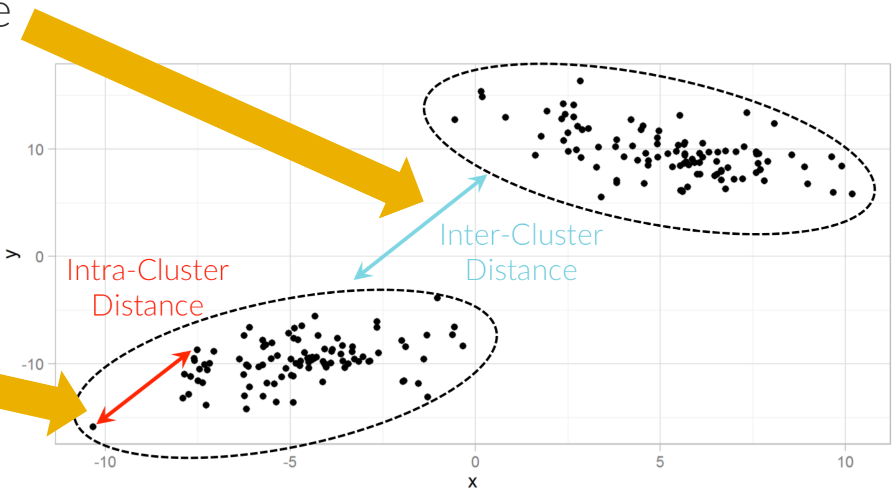
- Clustering is more powerful than the human eye in 3D
- Clustering mathematically defines which cluster the peripheral points should be in when it's not obvious to the human eye
- Clustering is helpful when many dimensions / variables exist that you can't visualize at once
  – Whiskey similarity example from classification lecture

Aye, Nay and Other Votes
in House of Representatives



How good is the clustering?

- Goals of clustering:
  - Maximize the separation between clusters
    - i.e. Maximize inter-cluster distance
  - Keep similar points in a cluster close together
    - i.e. Minimize intra-cluster distance



How good is the clustering?

Assessing how well an algorithm performs

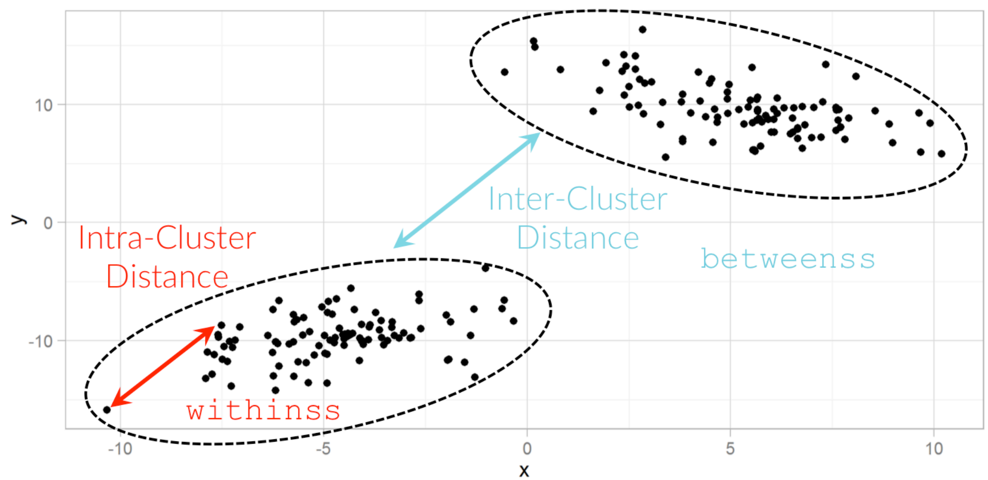
- Look at the variance explained by clusters
  - In particular, the ratio of inter-cluster variance to total variance

- How much of the total variance is explained by the clustering?

Variation explained by clusters

=

inter-cluster variance / total variance



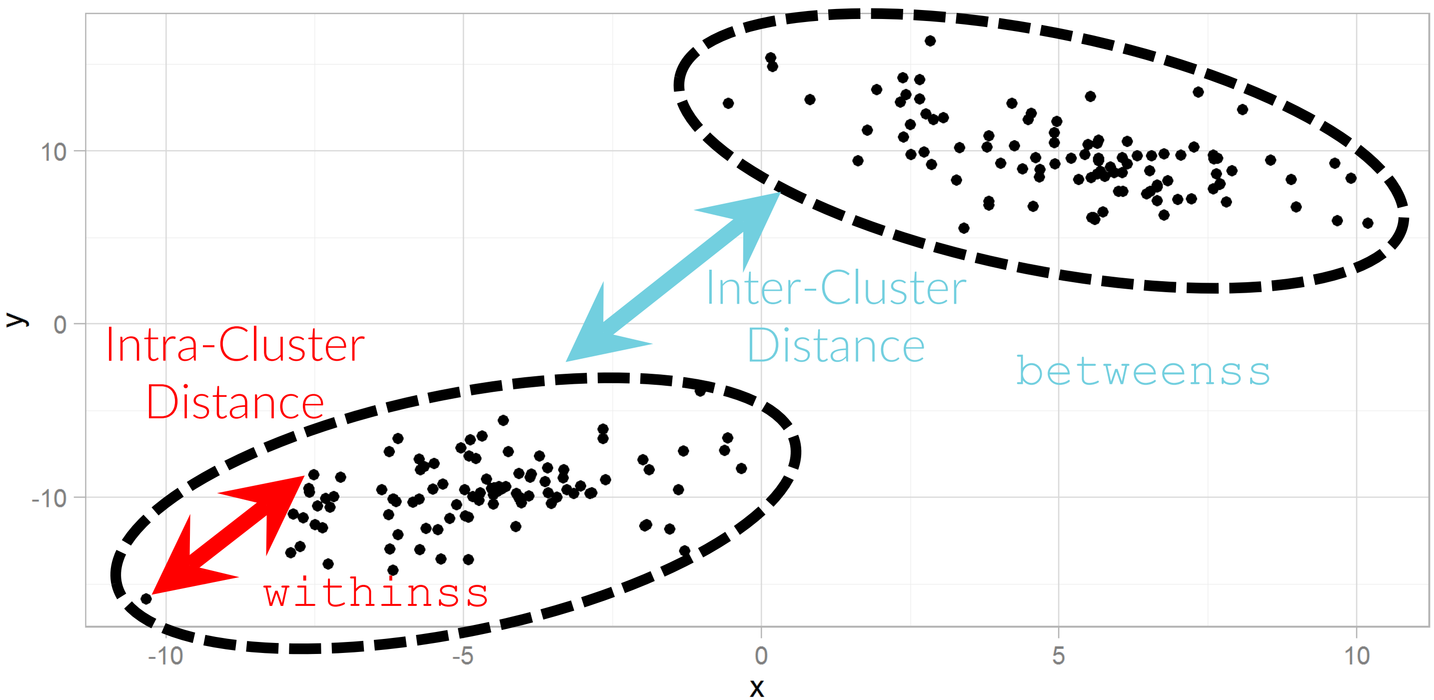
kmeans outputs

- cluster: a vector indicating the cluster to which each point is allocated
- centers: a matrix of cluster centers
- totss: the total sum of squares (sum of distances between all points)
- withinss: vector of within-cluster sum of distances, one number per cluster
- tot.withinss: total within-cluster sum of distances, i.e. sum of withinss
- betweenss: the between-cluster sum of squares, i.e. totss - tot.withinss
- size: the number of points in each cluster

To learn more about the **`kmeans`** *function run* **`?kmeans`**

Intra vs. inter-cluster distance

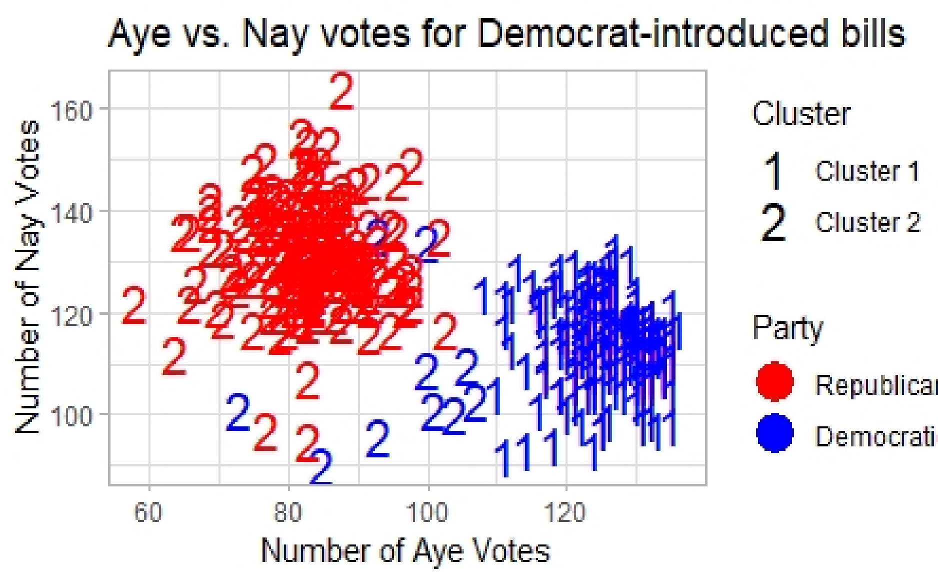
totss = withinss + betweenss



How good is the clustering?

Script

```
# Inter-cluster variance,
# "betweenss" is the sum of the
# distances between points from
# different clusters
num_Dem = kmeans_obj_Dem$betweenss

# Total variance
# "totss" is the sum of the distances
# between all the points in
# the data set
denom_Dem = kmeans_obj_Dem$totss

# Variance accounted for by
# clusters
var_exp_Dem = num_Dem / denom_Dem
var_exp_Dem
[1] 0.7193405
```



How good is the clustering?

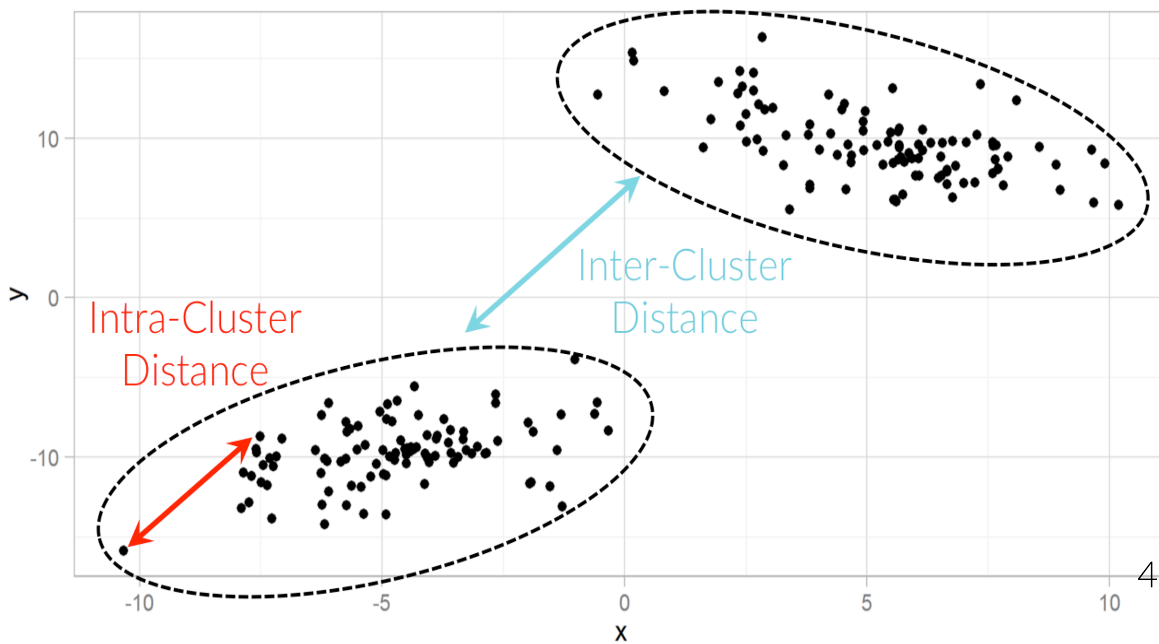
How do we choose the number of clusters (i.e. k)?

- It's easier when the number of clusters is known ahead of time, but what if we don't know how many clusters we should have?
- Since different starting points may generate different clusters, we need a way to assess cluster quality as well.

How to select k: two methods

1. Elbow method
   - Computes the percentage of variance explained by clusters for a range of cluster numbers
   - Plots a graph so results are easier to see
   - Not guaranteed to work! It depends on the data in question

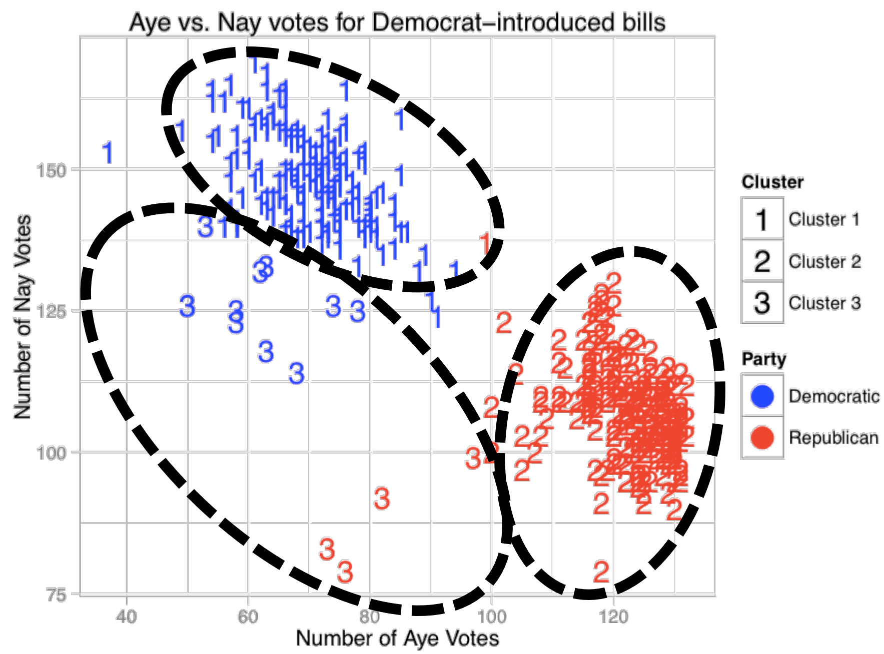
2. `NbClust`
   - Runs 30 different tests and provides “majority vote” for the best number of clusters (k’s) to use



Elbow method: measure variance

Script

```
# Run algorithm with 3 centers
set.seed(1)
kmeans_obj_Dem = kmeans(clust_data_Dem,
                        centers = 3,
                        algorithm = "Lloyd")

# Inter-cluster variance

num_Dem = kmeans_obj_Dem$betweenss

# Total variance

denom_Dem = kmeans_obj_Dem$totss

# Variance accounted for by clusters

var_exp_Dem = num_Dem / denom_Dem
var_exp_Dem
[1] 0.7949741
```



Automating a step we want to repeat

- We want to repeat the variance calculation from the previous slide for several numbers of clusters automatically

- We can create a function that contains all the steps we want to automate

```
function(data, item to iterate through)
```

Automating a step we want to repeat

Script

```
# The function explained_variance wraps our code from previous slides.

explained_variance = function(data_in, k){

  # Running k-means algorithm

  set.seed(1)
  kmeans_obj = kmeans(data_in, centers = k,
                      algorithm = "Lloyd")

  # Variance accounted for by clusters

  var_exp = kmeans_obj$betweenss /
            kmeans_obj$totss
  var_exp
}
```

1. A new variable is created and set equal to our `function()`
2. The commands inside the function are wrapped in `curly braces {}`
3. Inside the parentheses, we specify the variables that the user will input and that will then be used inside the function where they appear

Automating a step we want to repeat

Script

```
# Recall the variable we are using for the
# data that we're clustering.

clust_data_Dem = house_votes_Dem[, c("aye", "nay", "other")]
View(clust_data_Dem)

# The sapply() function plugs several values
# into explained_variance.

explained_var_Dem = sapply(1:10, explained_variance,
                           data_in = clust_data_Dem)

View(explained_var_Dem)

# Data for ggplot2

elbow_data_Dem = data.frame(k = 1:10,
                            explained_var_Dem)

View(elbow_data_Dem)
```

Function we created

1. `sapply()` applies a function to a vector
2. We have to tell `sapply()` that the we want the `explained_variance` function to use the `clust_data` data
3. Next, we create a data frame that contains both the new variance variable (`explained_var_Dem`) and the different numbers of k that we used in the previous function (1 through 10)

Elbow method: plotting the graph

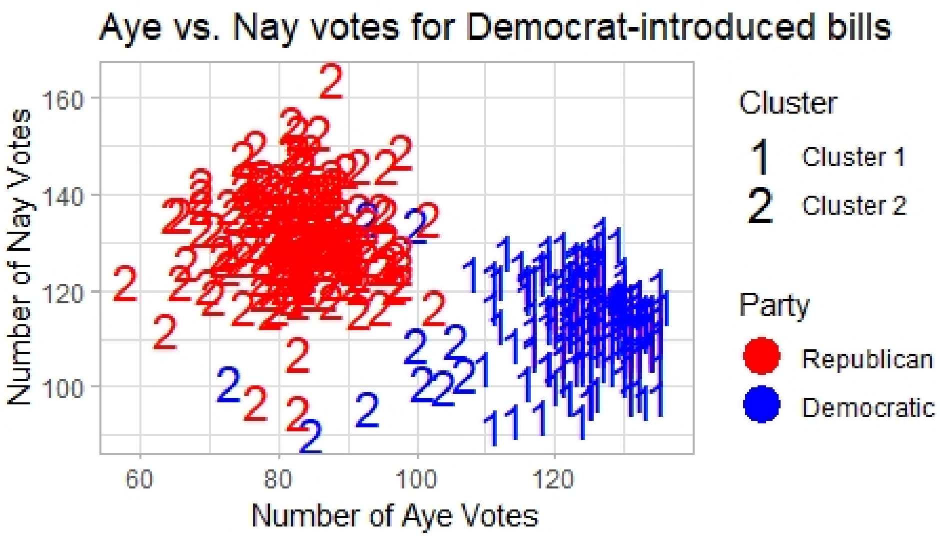
Script

```
# Plotting data

ggplot(elbow_data_Dem,
       aes(x = k,
           y = explained_var_Dem)) +
  geom_point(size = 4) +
  geom_line(size = 1) +
  xlab("k") +
  ylab("Intercluster Variance/Total Variance") +
  theme_light()
```

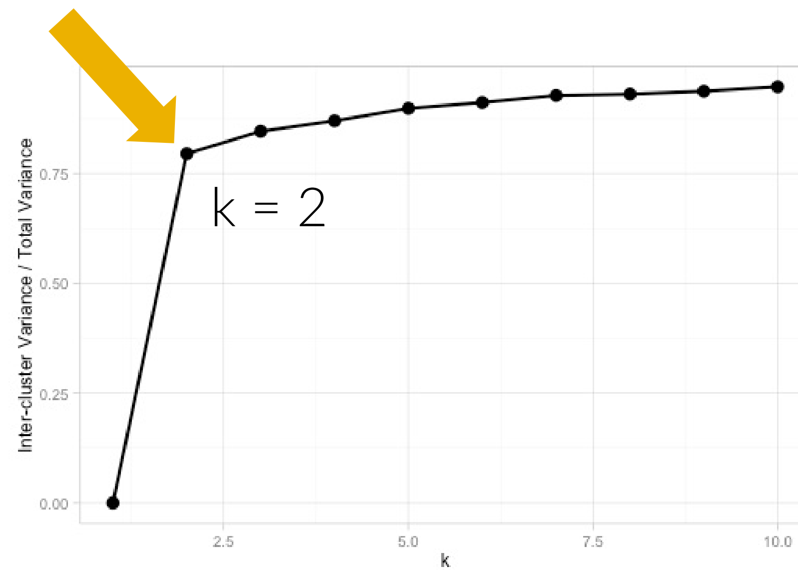
1. `geom_point()` sets the size of the data points
2. `geom_line()` sets the thickness of the line

Elbow method: measure variance

Looking for the kink in graph of inter-cluster variance / total variance



Original data



Elbow method

`NbClust`: k by majority vote

There are a number of ways to choose the right k.

`NbClust` runs 30 tests and selects k based on majority vote

- Library: `"NbClust"`
  - Functions: `"NbClust"`

```
NbClust(data, max.nc, method = "kmeans")
```

Inputs:

- `data` – data array or data frame
- `min.nc / max.nc` – minimum/maximum number of clusters
- `method` – `"kmeans"`
- There are other, more advanced arguments that can be customized but are outside of the scope of this course and are note necessary to for `NbClust` to work

NbClust: k by majority vote

Script

```
# Install the package.
install.packages("NbClust")
library(NbClust)

# Run NbClust.
nbclust_obj_Dem = NbClust(data = clust_data_Dem, method = "kmeans")

# View the output of NbClust.
nbclust_obj_Dem

# View the output that shows the number of clusters each
# method recommends.
View(nbclust_obj_Dem$Best.nc)
```

`NbClust`: k by majority vote

Console
```
> nbclust_obj_Dem = NbClust(data = clust_data_Dem, method = "kmeans")
   ...

*******************************************************************
* Among all indices:
* 14 proposed 2 as the best number of clusters
* 3 proposed 3 as the best number of clusters
* 1 proposed 4 as the best number of clusters
* 3 proposed 6 as the best number of clusters
* 1 proposed 9 as the best number of clusters
* 1 proposed 10 as the best number of clusters
* 1 proposed 15 as the best number of clusters

                   ***** Conclusion *****

* According to the majority rule, the best number of clusters is  2
```

Note: additional information appears; the above information is most relevant to us for now

`NbClust`: k by majority vote

- `nbclust_obj_Dem` shows the outputs of NbClust
  - One of the outputs is `Best.nc`, which shows the number of clusters recommended by each test

```
> nbclust_obj_Dem
```

Console

```
Console ~/Desktop/
$Best.nc
                    KL       CH Hartigan     CCC    Scott  Marriot     TrCovW   TraceW     Friedman     Rubin   Ci
ndex
Number_clusters   2.0000    2.000   15.000  2.000  15.0000   8.0000          3     3.00 4.00000e+00  13.0000 15.
0000
Value_Index     124.7563 1650.897   90.934 76.795 538.9619 276.7943 2049902656 16117.57 3.42403e+16 -85.7654  0.
1306
                    DB Silhouette   Duda PseudoT2  Beale Ratkowsky     Ball PtBiserial Frey McClain   Dunn Huber
t SDindex
Number_clusters 2.0000     2.0000 2.0000   2.0000 2.0000    2.0000     3.00     3.0000    1  2.0000 2.0000
0  2.0000
Value_Index     0.5038     0.7149 0.6358 115.1243 0.9702    0.5589 30173.51     0.8961   NA  0.2779 0.1005
0  0.0838
```

`NbClust`: k by majority vote

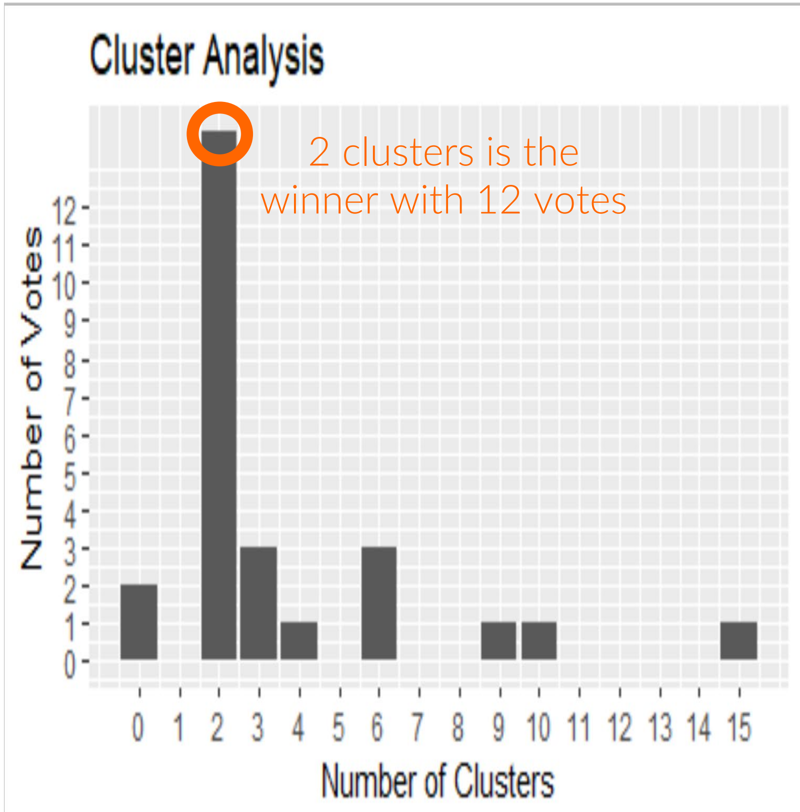
	KL	CH	Hartigan	CCC	Scott	Marriot	TrCovW	TraceW	Friedman	Rubin	Cindex
Number_clust...	2.0000	2.000	15.000	2.000	15.0000	8.0000	3	3.00	4.00000e+...	13.0000	15.0000
Value_Index	124.75...	1650.8...	90.934	76.7...	538.96...	276.7943	20499026...	16117.57	3.42403e+...	-85.76...	0.1306

- We want to visualize a histogram to make it obvious how many votes there are for each number of clusters

NbClust: k by majority vote

Script

```
# Subset the 1st row from Best.nc and convert it
# to a data frame, so ggplot2 can plot it.

freq_k_Dem = nbclust_obj_Dem$Best.nc[1,]
freq_k_Dem = data.frame(freq_k_Dem)
View(freq_k_Dem)

# Check the maximum number of clusters.

max(freq_k_Dem)

# Plot as a histogram.

ggplot(freq_k_Dem,
       aes(x = freq_k_Dem)) +
  geom_bar() +
  scale_x_continuous(breaks = seq(0, 15, by = 1)) +
  scale_y_continuous(breaks = seq(0, 12, by = 1)) +
  labs(x = "Number of Clusters",
       y = "Number of Votes",
       title = "Cluster Analysis")
```

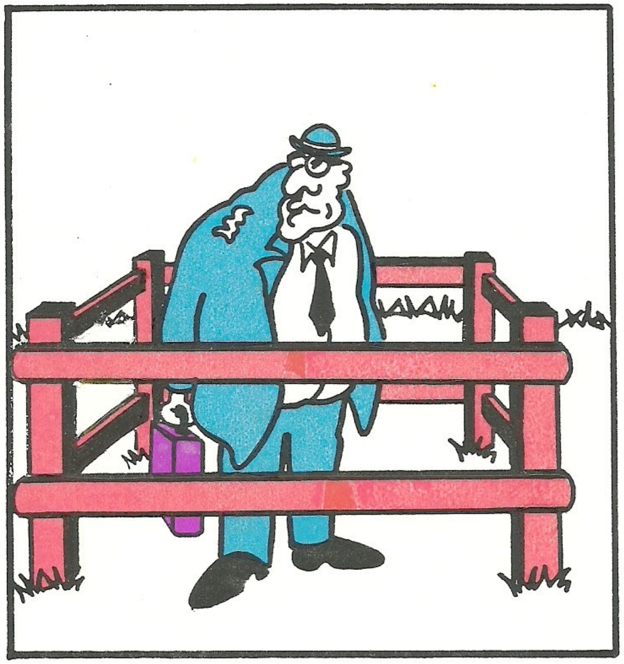


Application of results

- If you're a lobbyist, which congressperson can you influence for swing votes?
- If you're managing a campaign and your competitor is always voting along party lines, how can you use that information?
- If your congressperson is not an active voter, is she representing your interests?
- What do the voting patterns look like for Republican-introduced bills?

Implications of results

- Could see differences between the patterns of Reb lead bills and Democrat lead bills
- Could provide information on congressmen that might be see has swing votes.



DURLINGTON WAS AN EXPERT IN HIS FIELD. UNFORTUNATELY, HIS FIELD WAS A TEN FOOT SQUARE PLOT OF PASTURE IN IOWA.

Limitations of results

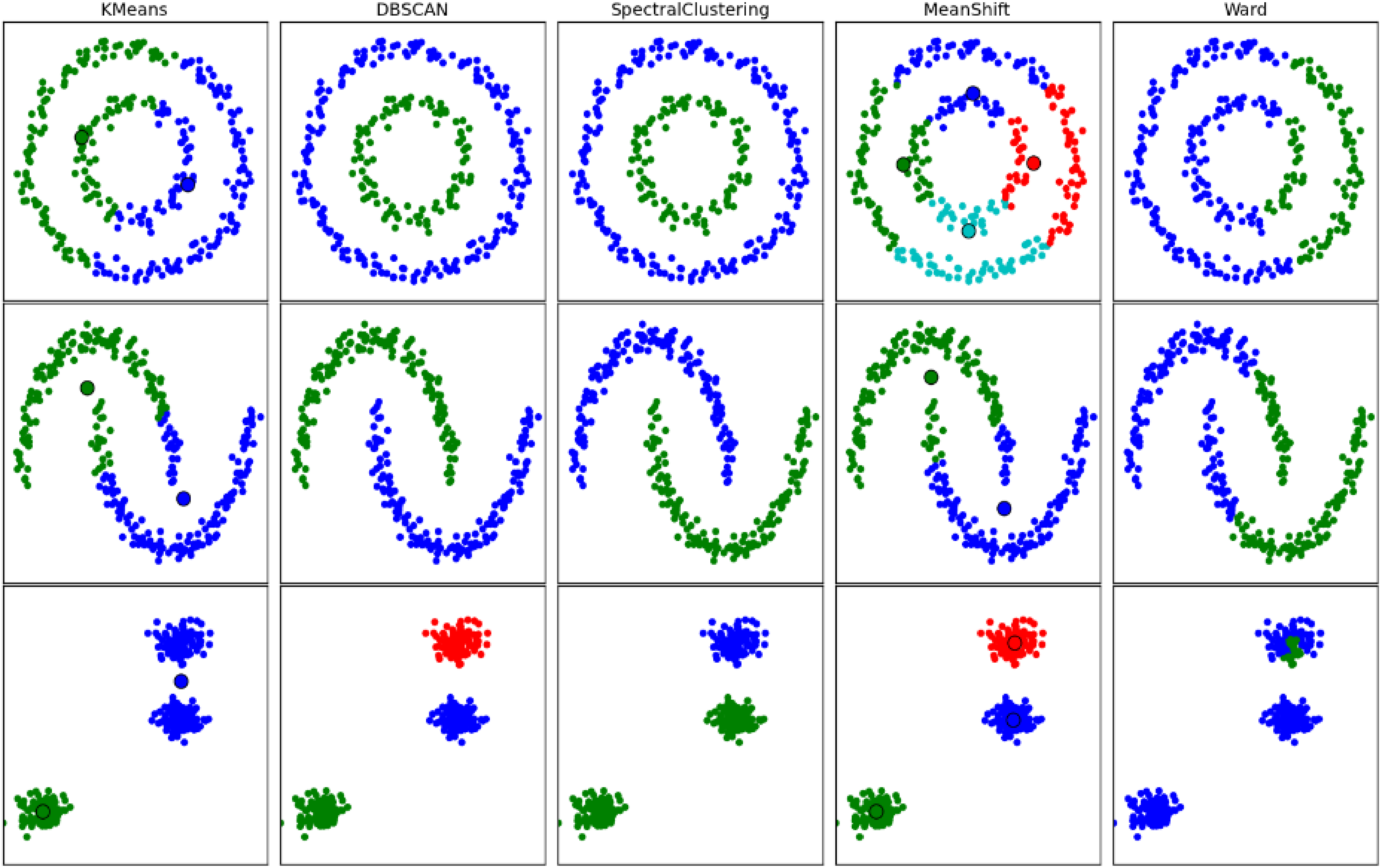
- We are assuming that the patterns correspond with the same bills being voted on – perhaps some Congressmen have the same number of 'aye' and 'nay' votes, but voted on different bills
- Network analysis can help determine additional connections between Congressmen
- We haven't taken extenuating factors into account – political initiatives, current events, etc.

This is a preliminary analysis that gives us initial insights and can help us direct further research

The good, bad, and evil

- The good and bad
  - + cheap – NO LABELS, labels are expensive to create and maintain
  - +/- clustering always works
  - - Many methods to choose from and knowing the right one can be nontrivial and the differences between many are almost zero, so you need to understand what you're doing

- The evil
  - Curse of dimensionality
  - Clusters may result from poor data quality
  - Non-deterministic (e.g. k-means) subject to local minimum. Since it works with averages, k-means does not get much better with Big Data (marginal improvements)
  - Non spherical data may result in poor clustering (depending on method used)
  - Unequal cluster sizes may result in poor clustering (depending on method used)


KMeans
DBSCAN
SpectralClustering
MeanShift
Ward

The good, bad, and evil

- Analysts need to ask the following questions
  - Do you want overlapping or non-overlapping clusters?
  - Does your data satisfy the assumptions of the clustering algorithm?
  - How was the distance measure identified?
  - How many clusters and why? Identifying the number of clusters is a difficult task if the number of class labels is not known beforehand
  - Does your method scale to the size of the data?
  - Is the compute time congruent with the temporal budget of your business need (i.e. do you get answers back in time to make meaningful decisions)