Data Science for Public Policy

Aaron R. Williams - Georgetown University

Unsupervised Machine Learning – Cluster Analysis

Reading

- Hands on Machine Learning with R Chapter 20
- Catalyzing Policing Reform with Data

Unsupervised Machine Learning

Background

Cluster analysis: The process of using algorithms to assign observations to groups by calculating similarity between the observations (rows) using their variables (columns). The goal is to minimize the similarity between groups and maximize the similarity within groups.

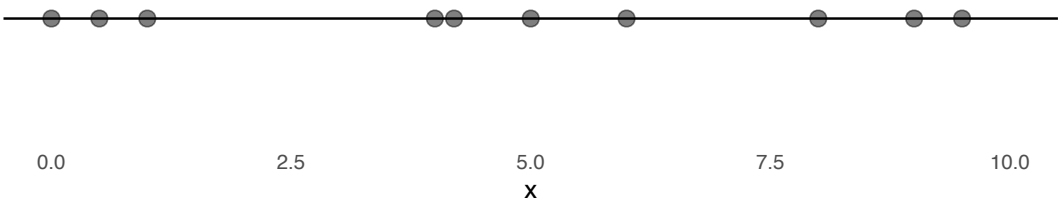
Cluster assignments are the arbitrary labels for the groups (i.e. cluster 1 and cluster 2). Cluster assignments can be used for data visualization and calculating group-wise summary statistics like cluster means.

In practice, clusters can be used to create typologies and to summarize rows in a complicated data set. At the Urban Institute, researchers have used cluster analysis to create neighborhood types and to categorize counties based on financial wellness indicators.

Intuitive Explanation

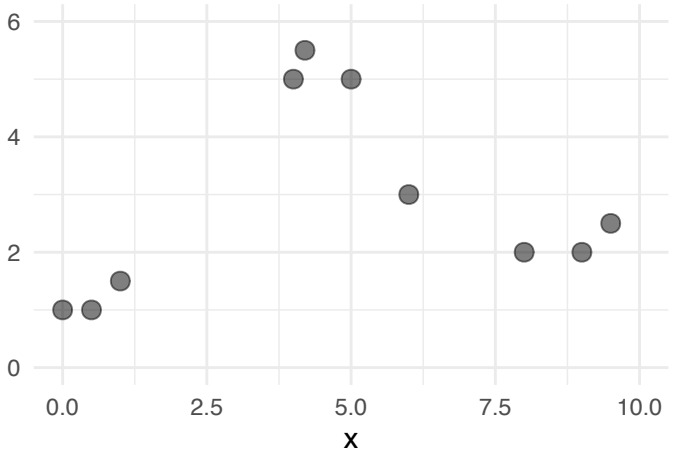
Example 1

Consider the following 10 observations. How might you meaningfully cluster the observations?



Example 2

Consider the following 10 observations. How might you meaningfully cluster the observations?



These two examples raise three important questions:

1. What is a formal method to calculate similarity between the observations?
2. How can we scale up this process to many observations and many variables?
3. How many groups should be created?

1. What is a formal method to calculate similarity between the observations?

Euclidean distance is very popular for cluster analysis:

One variable:

$$Dist_{ij} = \sqrt{(x_i - x_j)^2}$$

Multiple variables:

$$Distance_{ij} = \sqrt{\sum_{k=1}^{p}(x_{ik} - x_{jk})^2}$$

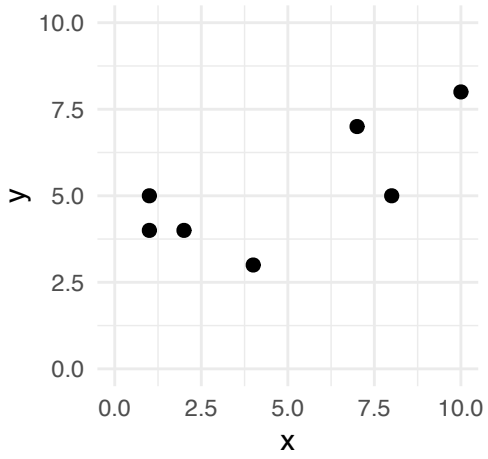
Where i is the first observation, j is the second observation, k is the k^{th} variables, and p is the number of variables.

2. How can we scale up this process to many observations and many variables?

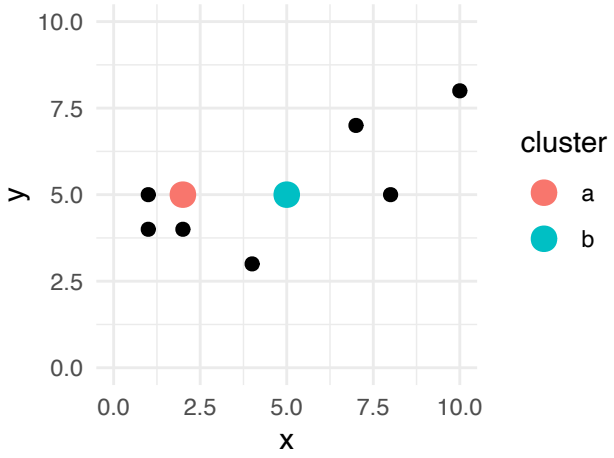
When the clusters are clear and there aren't many variables, we can intuit reasonable groups (like above). But what if there are 30 variables? Or 500 observations?

K-means clustering is a popular clustering algorithm. It is different than KNN but uses many of the same mathematical ideas.

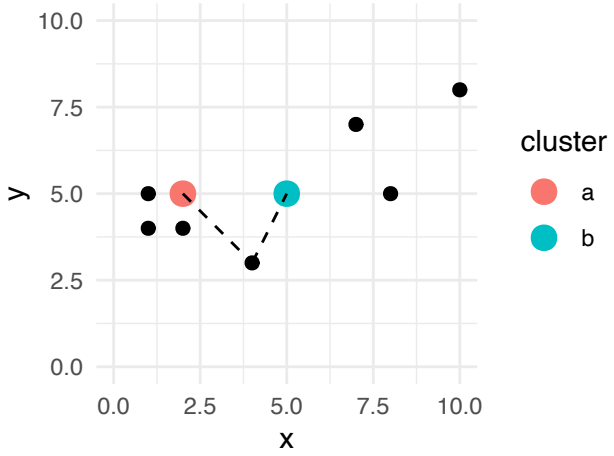
Consider the following data set.



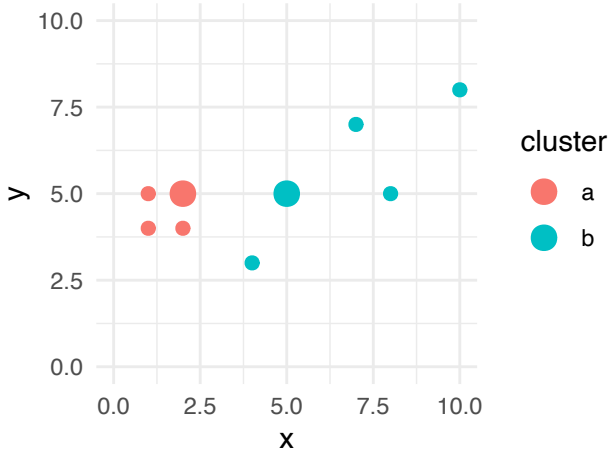
Step 1: Randomly place K centroids in your n-dimensional vector space



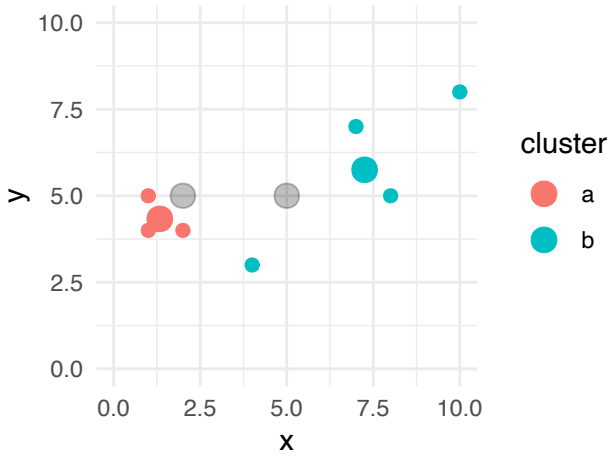
Step 2: Calculate the nearest centroid for each point using a distance measure



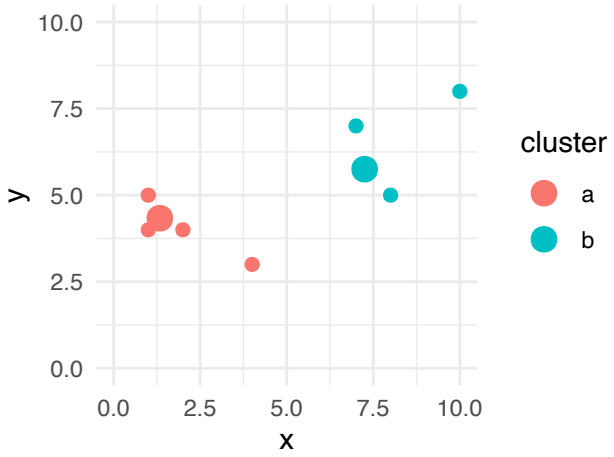
Step 3: Assign each point to the nearest centroid



Step 4: Recalculate the position of the centroids based on the means of the assigned points



Step 5: Repeat steps 2-4 until no points change cluster assignments



Note:

1. The data typically need to be on the same scale.
2. K-means clustering is not robust to outliers.
3. The cluster assignments are not deterministic.

Implementation

Tutorial 1

Consider the state data from earlier.

```r
library(tidyverse)
library(broom)

# select numeric variables of interest
state_data <- urbnmapr::statedata %>%
  select(hhpop, horate, medhhincome)

# standardize the variables
state_data_numeric <- state_data %>%
  mutate_all(.funs = ~ scales::rescale(.x))

# set a seed because the clusters are not deterministic
set.seed(20200205)

# predict four clusters
state_data_kmeans <- kmeans(
  state_data_numeric,
  centers = 4, # number of clusters
  nstart = 100 # number of random starts
)

tidy(state_data_kmeans) %>%
  knitr::kable(digits = 2)
```

hhpop	horate	medhhincome	size	withinss	cluster
0.13	0.82	0.26	28	1.00	1
0.00	0.00	0.97	1	0.00	2
0.71	0.55	0.48	4	0.32	3
0.10	0.78	0.71	18	0.91	4

Tutorial 2

Recall the votes data set from the previous tutorial.

```r
votes <- read_csv(here::here("tutorials", "12_unsupervised-ml-cluster-analysis", "votes.csv"))
```

```
## Rows: 100 Columns: 495
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr   (2): name, party
## dbl (493): v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12, v13, v14, v15,...
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Principal Components Analysis: 主成分分析

```
# select the numeric variables
votes_numeric <- votes %>%
  select_if(is.numeric)

# run PCA
votes_pca <- prcomp(votes_numeric)

# extract the principle components
votes_pcs <- votes_pca %>%
  .$x %>%
  as_tibble()

# combine the pcs to the names and parties
votes_pcs <- bind_cols(
  select(votes, name, party),
  votes_pcs
)

# fit with 2 clusters
votes_kmeans2 <- kmeans(
  votes_numeric,
  centers = 2, # number of clusters
  nstart = 100 # number of random starts
)

bind_cols(
  select(votes, name, party),
  cluster = votes_kmeans2$cluster
) %>%
  count(party, cluster)
```

```
## # A tibble: 3 x 3
##   party cluster     n
##   <chr>   <int> <int>
## 1 D           2    44
## 2 I           2     2
## 3 R           1    54
```

We see that the estimated clusters fit the party variable even though it wasn't included in the model.

Latent variable (hidden variable): An unobserved variable inferred through statistical methods

Suppose we didn't known party. Then cluster analysis is the process of trying to discover a latent variable or latent groups in the data. In general, cluster analyses will be much tidier if there is a strong latent process driving the latent groups. For example, imagine latent groups for children, women, and men when looking at heights and weights without observing age or sex.

In addition to identifying data and variables that capture latent structure in our data, we

also need to choose how many latent groups are in the data! Consider the above example with 3 groups:

```
# fit with 3 clusters
votes_kmeans3 <- kmeans(
  votes_numeric,
  centers = 3, # number of clusters
  nstart = 100 # number of random starts
)

bind_cols(
  select(votes, name, party),
  cluster = votes_kmeans3$cluster
) %>%
  count(party, cluster)
```

```
## # A tibble: 4 x 3
##   party cluster     n
##   <chr>   <int> <int>
## 1 D           2    44
## 2 I           2     2
## 3 R           1    23
## 4 R           3    31
```

3. Picking the number of clusters

The main considerations when picking a number of clusters are theory and application. Is the objective to find three archetypal communities? Does the visualization need five groups?

Beyond theory and application, there are analytic measures that can help determine the optimal number of clusters. The best number of clusters usually can be picked with the "elbow method" by choosing the number where a kink occurs in the following plots. All of these plots come from `library(factoextra)`.

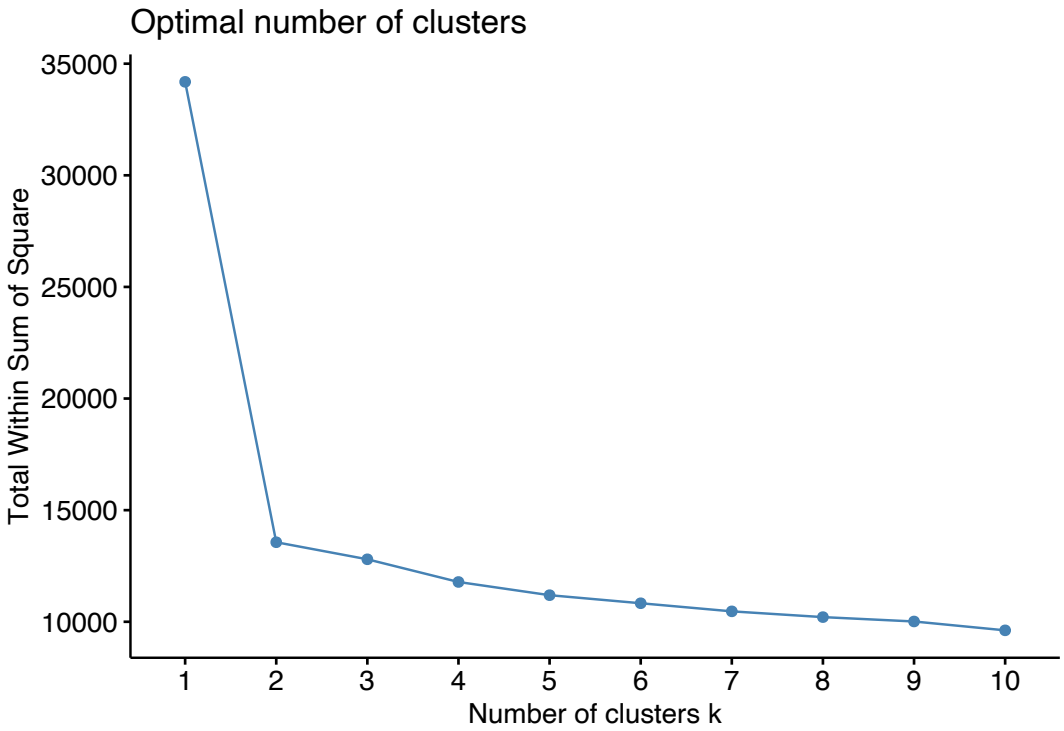
```
library(factoextra)
```

Consider the votes data where there are two clear clusters.

Total within sum of squares

The simplest measure is total within sum of squares. This is simply the sum of squared errors within groups, which should monotonically decrease as the number of clusters increases.

```
fviz_nbclust(votes_numeric, FUN = kmeans, method = "wss")
```

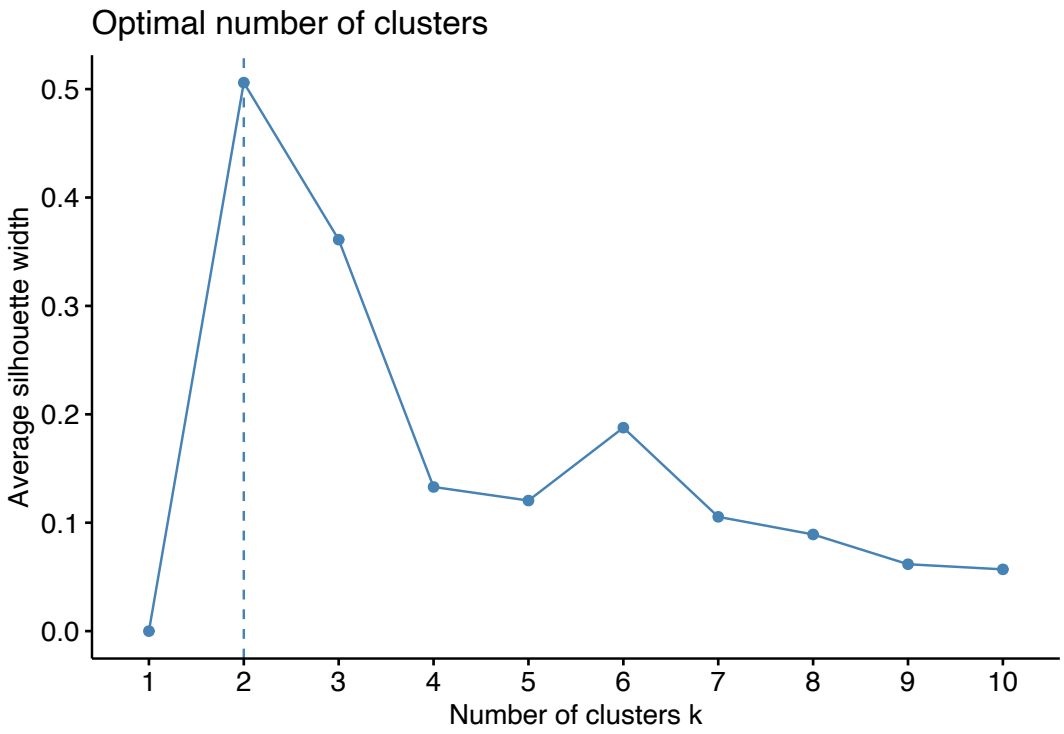


Average silhouette width

The silhouette score is an observation-level measure of cohesion within an observation's group and separation from the closest group. It ranges from -1 to +1, with higher numbers representing better clusters. Negative numbers suggest that an observation has been assigned to an incorrect cluster. 0 indicates that an observation is on the boundary. Positive numbers suggest correct assignment. The average across all observations can be used to pick cluster sizes.

```
fviz_nbclust(votes_numeric, FUN = kmeans, method = "silhouette")
```

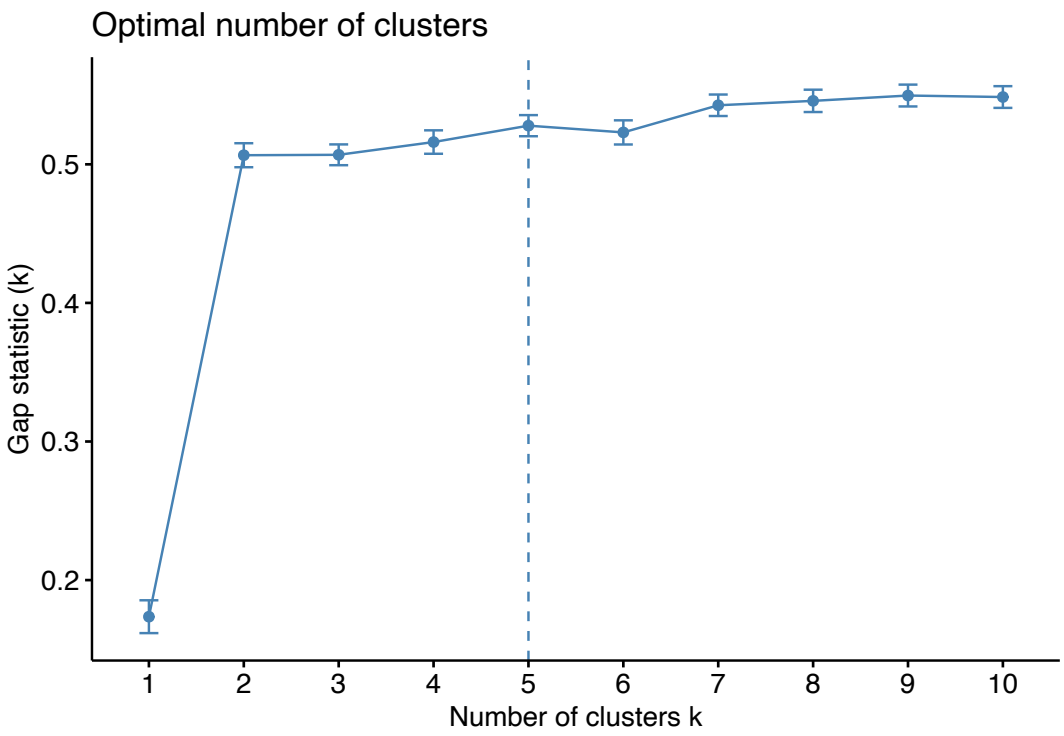
但这里没讲是怎么计算的啊



Gap statistics

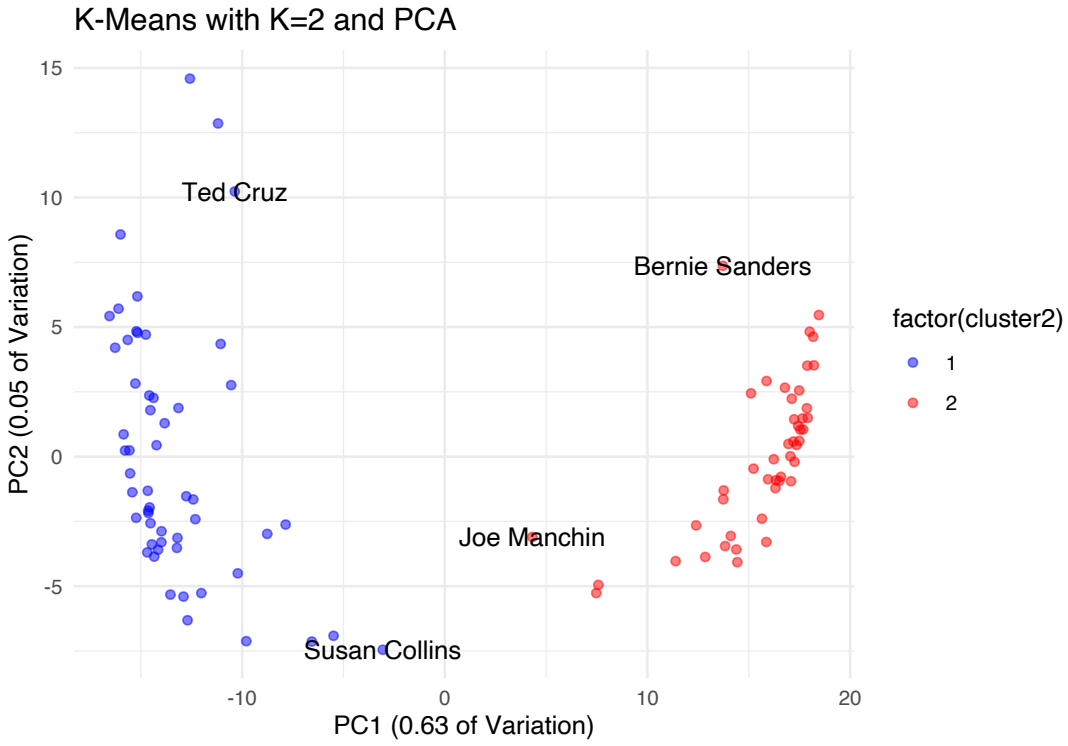
The gap statistic (Tibshirani, Walther, and Hastie 2000), compares the change in within-cluster dispersion with a reference null distribution that represents data without clear clusters. Higher numbers are better. It takes the longest to calculate because it requires bootstrapping Monte Carlo simulations to calculate the null distribution.

```
fviz_nbclust(votes_numeric, FUN = kmeans, method = "gap_stat")
```



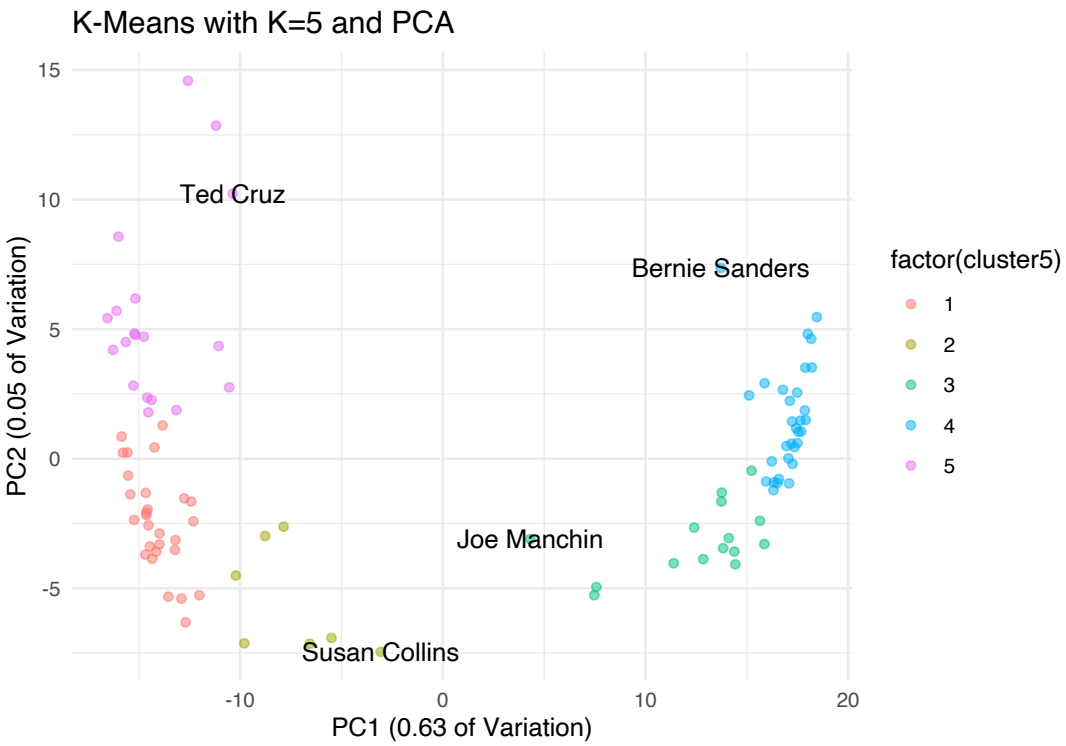

```
votes_kmeans5 <- kmeans(
  votes_numeric,
  centers = 5, # number of clusters
  nstart = 100 # number of random starts
)

votes_clusters <- bind_cols(
  select(votes, name, party),
  select(votes_pcs, PC1, PC2),
  cluster2 = votes_kmeans2$cluster,
  cluster5 = votes_kmeans5$cluster
)

names <- c("Bernie Sanders", "Ted Cruz", "Joe Manchin", "Susan Collins")

ggplot() +
  geom_point(
    data = votes_clusters,
    mapping = aes(PC1, PC2, color = factor(cluster2)),
    alpha = 0.5
  ) +
  geom_text(
    data = filter(votes_pcs, name %in% names),
    mapping = aes(PC1, PC2, label = name)
```

```
  ) +
  scale_color_manual(values = c("blue", "red")) +
  labs(
    title = "K-Means with K=2 and PCA",
    x = "PC1 (0.63 of Variation)",
    y = "PC2 (0.05 of Variation)"
  ) +
  theme_minimal() +
  guides(text = NULL)
```




```
ggplot() +
  geom_point(
    data = votes_clusters,
    mapping = aes(PC1, PC2, color = factor(cluster5)),
    alpha = 0.5
  ) +
  geom_text(
    data = filter(votes_pcs, name %in% names),
    mapping = aes(PC1, PC2, label = name)
  ) +
  labs(
    title = "K-Means with K=5 and PCA",
    x = "PC1 (0.63 of Variation)",
    y = "PC2 (0.05 of Variation)"
```

```
  ) +
  theme_minimal() +
  guides(text = NULL)
```



Additional Considerations

Picking variables

Start with a subset of numeric variables based on subject matter expertise. In general, clustering algorithms work well with all continuous variables or all recoded categorical variables. Mixing variable types is possible but requires extra care.

```
# load the tidyverse
library(tidyverse)

# load statedata from library(urbnmapr)
state_data <- urbnmapr::statedata
```

```
# select numeric variables other than year
state_data <- state_data %>%
  select(hhpop, horate, medhhincome)

knitr::kable(head(state_data), digits = 3)
```

hhpop	horate	medhhincome
1846380	0.681	44700
250183	0.631	70600
2463012	0.621	51000
1144657	0.655	42000
12895471	0.537	64600
2074517	0.639	63500

Scaling variables

The variable `hhpop` will swamp `horate` and `medhhincome` because of its scale. We need to put the variables on the same scale. The first option is to standardize variables by subtracting the mean and dividing by the standard deviation.

$$\tilde{x_i} = \frac{x_i - \bar{x}}{s_x}$$

`mutate_all()` is a convenient function to apply a transformation to all variables.

```
state_data %>%
  mutate_all(.funs = ~(.x - mean(.x)) / sd(.x))
```

```
## # A tibble: 51 x 3
##      hhpop horate medhhincome
##      <dbl>  <dbl>       <dbl>
##  1 -0.190   0.575      -1.21
##  2 -0.833  -0.312       1.53
##  3  0.0585 -0.498      -0.546
##  4 -0.472   0.101      -1.50
##  5  4.26   -1.97        0.894
##  6 -0.0979 -0.175       0.777
##  7 -0.392   0.238       1.65
##  8 -0.791   1.07        0.534
##  9 -0.820  -4.34        1.99
## 10  2.07   -0.186      -0.757
## # ... with 41 more rows
```

The second option is to normalize the variable to a 0 to 1 or 0 to 100 scale. `rescale()` normalizes a variable.

$$\tilde{x_i} = \frac{x_i - x_{min}}{x_{max} - x_{min}}$$

```
state_data %>%
  mutate_all(scales::rescale)
```

```
## # A tibble: 51 x 3
##      hhpop horate medhhincome
##      <dbl>  <dbl>       <dbl>
##  1 0.128    0.871      0.131
##  2 0.00168  0.714      0.85
##  3 0.176    0.681      0.306
##  4 0.0723   0.787      0.0556
##  5 1        0.421      0.683
##  6 0.146    0.738      0.653
##  7 0.0880   0.811      0.881
##  8 0.00976  0.959      0.589
##  9 0.00417  0          0.972
## 10 0.571    0.736      0.25
## # ... with 41 more rows
```

Standardization and normalization remove units from variables and put variables on the same scale.

Finding correlated variables

Including highly correlated variables in a clustering algorithm will generally overweight the variability of the highly correlated variables. Use `cor()` to create a correlation matrix.

```
cor(state_data)
```

```
##                   hhpop     horate medhhincome
## hhpop        1.00000000 -0.2926963  0.01883672
## horate      -0.29269634  1.0000000 -0.37477714
## medhhincome  0.01883672 -0.3747771  1.00000000
```

In the presence of highly-correlated variables, there are several options of recourse:

1. Drop all but one of the highly correlated variables.
2. Combine highly correlated variables.
    - `sum()`, `mean()`, or principle components
3. Re-weight the variables.

- Duplicate all variables `n` times and add or subtract duplicates based on if the variable is correlated (i.e. Pearson's correlation coefficient > 0.6) or highly correlated (i.e. Pearson's correlation coefficient > 0.9).

4. Use dimension reduction techniques like PCA.

Diagnostics

Consider the k-means clusters from the states data.

hhpop	horate	medhhincome	size	withinss	cluster
0.13	0.82	0.26	28	1.00	1
0.00	0.00	0.97	1	0.00	2
0.71	0.55	0.48	4	0.32	3
0.10	0.78	0.71	18	0.91	4

There are several ways that we can evaluate the quality of the clusters after the fact.

Expert check

```
bind_cols(
  state_data,
  cluster = state_data_kmeans$cluster
) %>%
  group_by(cluster) %>%
  summarize(
    mean(hhpop),
    mean(horate),
    mean(medhhincome)
  ) %>%
  knitr::kable(digits = 3)
```

cluster	mean(hhpop)	mean(horate)	mean(medhhincome)
1	1895957	0.665	49283.96
2	281788	0.402	75000.00
3	9253386	0.579	57250.00
4	1545705	0.653	65555.56

Stability (sensitivity analysis)

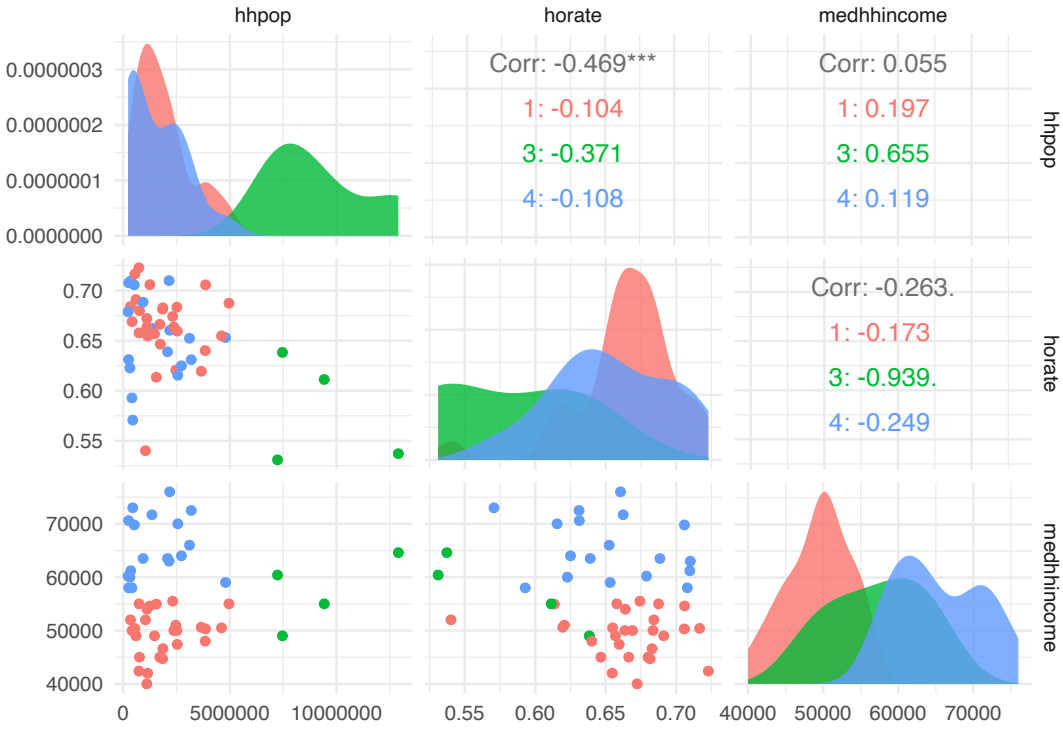
It is important to conduct a stability (sensitivity) analysis, in which the final specification of the cluster analysis is run p times, where p is the number of variables in the cluster analysis, each time removing one variable and checking how many observations remain in similar groups as a result.

Visualizing clusters with a scatter plot matrix

A scatter plot matrix allows for the pairwise comparison of all numeric variables. Adding cluster membership as colors shows the membership of each point in the scatter plots.

Note, scatter plot matrices become unwieldy with many variables or many observations. If there are too many observations or too many variables, visualize a representative subset. Also note that every cluster need not differ by every variable for a meaningful multivariate grouping to exist.

```
library(GGally)

#add cluster membership to the origin data
bind_cols(
  state_data,
  cluster = state_data_kmeans$cluster
) %>%
  filter(cluster != 2) %>%
  # create a scatter plot matrix
  ggpairs(
    columns = 1:3,
    mapping = aes(color = factor(cluster)),
    diag = list(continuous = wrap("densityDiag",
                                  alpha = 0.8,
                                  color = NA))
  ) +
  theme_minimal()
```



Visualizing clusters with PCA

A scatter plot matrix requires $\frac{p(p-1)}{2}$ plots to show p variables. This can quickly grow unwieldy.

We can use PCA to plot much of the variation in the data in two dimensions. The output shows that the first two components cumulatively explain more than 80% of variation in the data.

```
# run PCA
principle_components <- prcomp(state_data_numeric, scale. = FALSE)

# view the amount of variance explained
summary(principle_components)
```

```
## Importance of components:
##                           PC1    PC2    PC3
## Standard deviation     0.2762 0.2060 0.1412
## Proportion of Variance 0.5502 0.3060 0.1438
## Cumulative Proportion  0.5502 0.8562 1.0000
```

```r
# create and visualize the group membership of observations in the components
bind_cols(
  as_tibble(principle_components$x),
  cluster = state_data_kmeans$cluster
) %>%
  ggplot(aes(PC1, PC2, color = factor(cluster))) +
  geom_point() +
  labs(title = "State Data Clusters and PCA") +
  theme_minimal()
```

Putting it all together

1. Clearly define the question of interest
2. Pick the variables that will be considered
    - Recode categorical variables to dummy variables
    - Rescale variables
3. Check the variables for high correlations
    - Apply reweighting or variable selection to create a mostly uncorrelated subset
4. Pick an algorithm
5. Estimate the clusters and pick an optimal number of clusters

6. Diagnostics
    - Expert check
    - Stability analysis
    - Visualize with PCA

Example projects

- Disrupting Food Insecurity (feature, technical appendix)
    - This is a good example of explaining the clusters after assigning cluster membership.
- Residential Mobility and Neighborhood Change: Real Neighborhoods Under the Microscope (paper)
- Beyond Red vs. Blue: The Political Typology (blog)
- Who Profits from Amateurism? Rent Sharing in Modern College Sports Page 60

Appendix

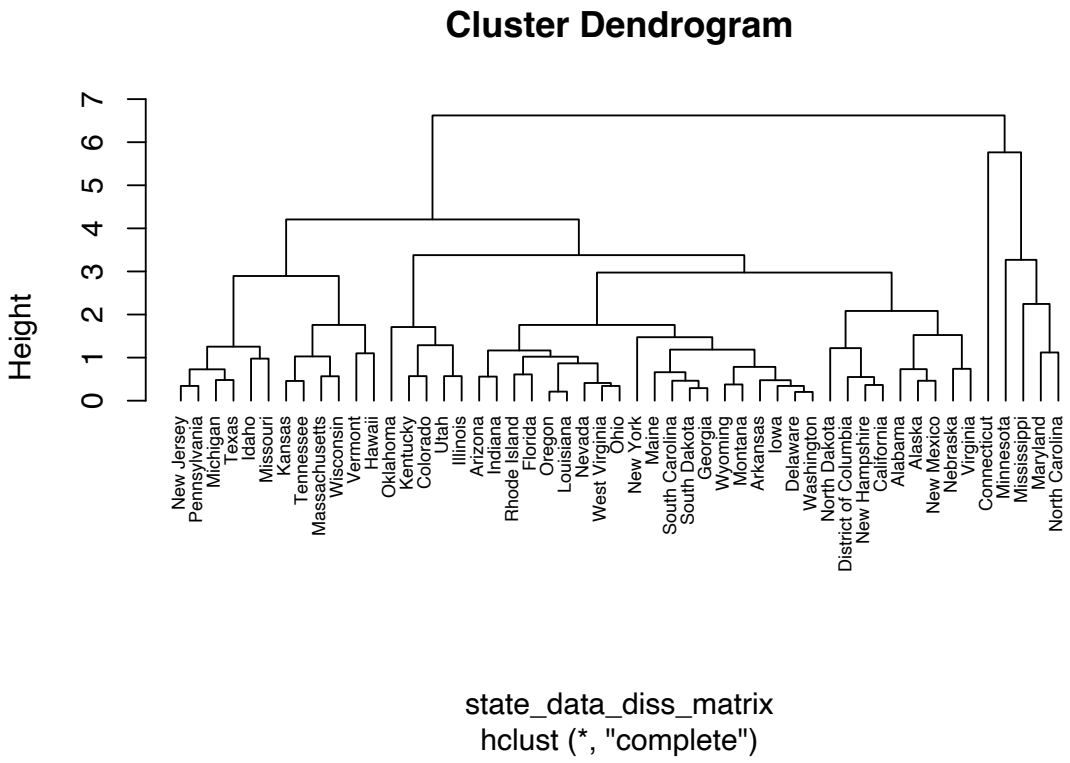
Partition-based algorithms

This blog post is a thorough background on K-means clustering.

Hierarchical models

Agglomerative hierarchical clustering: Each element starts as a single-element cluster and are combined into bigger clusters.

```
library(cluster)
library(dendextend)

# select numeric variables of interest
state_data <- urbnmapr::statedata %>%
  select_if(.predicate = is.numeric) %>%
  select(-year)

# standardize the variables
state_data <- state_data %>%
  mutate_all(.funs = ~(.x - mean(.x)) / sd(.x))

# create a dissimilarity matrix
state_data_diss_matrix <- dist(state_data,
                               method = "euclidean")

# run the hierarchical clustering algorithm
state_data_hclust <- hclust(
  d = state_data_diss_matrix,
  method = "complete"
)

# add state labels
state_data_hclust$labels <- urbnmapr::statedata$state_name[state_data_hclust$order]

# create a dendrogram of the clusters
plot(state_data_hclust,
     cex = 0.6,
     hang = -1)
```



Use `cutree()` to pick a height and assign group membership to observations based on the height. The height should be determined by the number of desired clusters.

```
# use four clusters
cutree(tree = state_data_hclust, k = 4)
```

```
##            Delaware             Vermont                Utah
##                   1                   2                   1
##             Wyoming           Minnesota      New Hampshire
##                   1                   3                   1
##       Massachusetts          New Jersey         Connecticut
##                   2                   2                   4
##            Maryland              Alaska              Hawaii
##                   3                   1                   2
##              Nevada        North Dakota        Rhode Island
##                   1                   1                   1
##             Arizona              Oregon                Iowa
##                   1                   1                   1
##               Maine             Indiana           Wisconsin
##                   1                   1                   2
##              Kansas            Nebraska               Idaho
##                   2                   1                   2
##             Montana        South Dakota       West Virginia
##                   1                   1                   1
##           Louisiana            Oklahoma            Missouri
```

```
##                  1                  1                  2
##          Tennessee           Arkansas        Mississippi
##                  2                  1                  3
##         New Mexico           Kentucky            Alabama
##                  1                  1                  1
##       South Carolina           Illinois           Virginia
##                  1                  1                  1
##           Colorado         Washington               Ohio
##                  1                  1                  1
##            Georgia      North Carolina           Michigan
##                  1                  3                  2
##       Pennsylvania District of Columbia         California
##                  2                  1                  1
##           New York            Florida              Texas
##                  1                  1                  2
```

Divisive hierarchical clustering begins with a single cluster and divides it into smaller clusters. Agglomerative hierarchical clusters is good at finding small groups. Divisive hierarchical clustering is good at finding large groups.

This blog post is a good deeper dive into hierarchical clustering.

Density-based algorithms

DBSCAN can be implemented with `library(dbscan)`.

Animation of DBScan

Model-based algorithms

Gaussian Mixture Modeling can be implemented with `library(mclust)`. This blog is a thorough introduction.