JSC 270 - Lecture 7
Clustering

https://jsc270.github.io/

ANNOUNCEMENTS

Perusall will be available at 2pm today

Next lecture is a guest lecture on NLP by Alistair Johnson

Supervised vs Unsupervised Learning

Supervised learning:

Data: (x_1,y_1), (x_2,y_2),...,(x_N,y_N)

Goal: find f(x), such that f minimizes a loss, eg $L(y, \hat{y}) = (y - \hat{y})^2$

Examples: classification, regression

Supervised vs Unsupervised Learning

Supervised learning:

Data: (x_1,y_1), (x_2,y_2),...,(x_N,y_N)

Goal: find f(x), such that f minimizes a loss, eg $L(y,\hat{y}) = (y-\hat{y})^2$

Examples: classification, regression

Unsupervised:

Data: $(x_1, x_2,..., x_N)$

Goal: P(X) or characteristics of the set of data

Examples: clustering

Clustering

How would you define clustering intuitively?

Clustering

Identifying subsets of similar items, where similar can be defined in a variety of ways

Examples of clustering

What applications of clustering can you think of?

Examples of clustering

Healthcare: identifying sets of similar patients (called *subphenotyping*)

Marketing: identifying groups of clients that could be targeted with the same messaging

Text: grouping together emails/documents by topic

What clustering methods do you know?

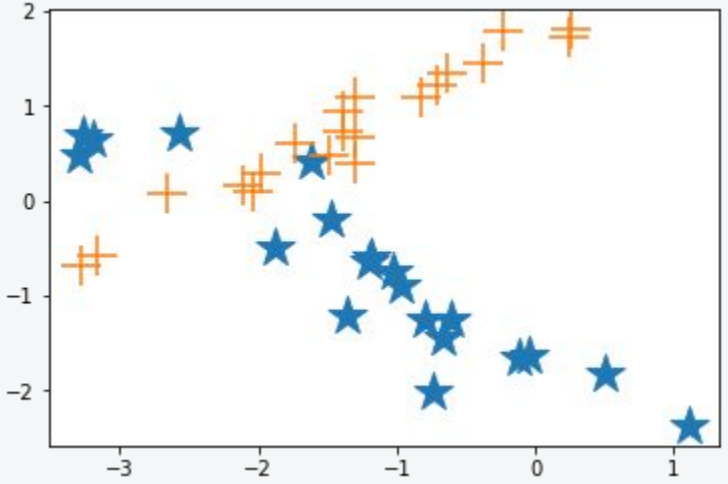
Clustering methods out there...

- Hierarchical clustering
  - Agglomerative
- K-means (X-means)
- Model based (e.g. Mixture of Gaussians)
- Affinity propagation
- Spectral
- LDA (Latent Dirichlet Allocation)
- Network/Graph clustering
  - Graph cuts (e.g. min cut - cut min # of edges)
  - Highly connected subgraphs

many more....

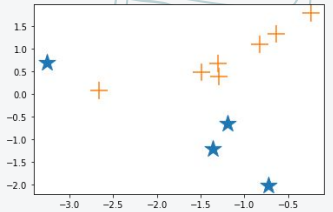
DATA

DATA

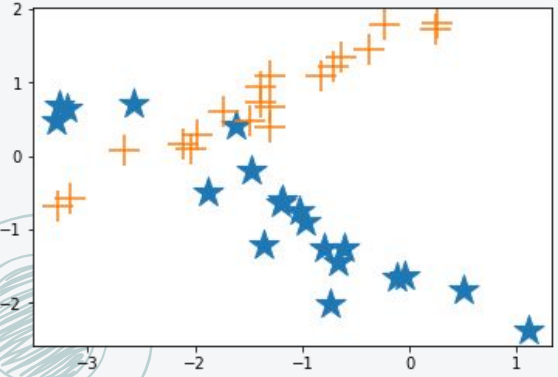


40 points
2 features
2 clusters

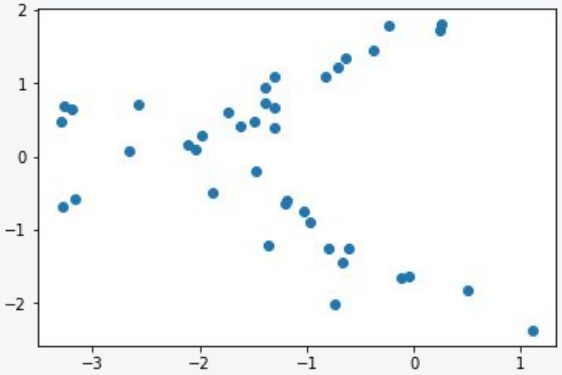
DATA



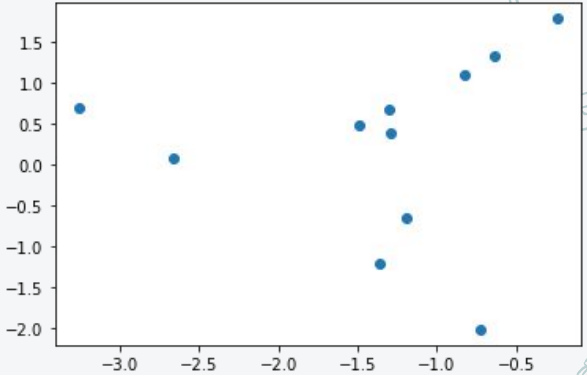
Original data



What we see



A subsample of 11 points



Clustering methods out there...

- Hierarchical clustering
  - **Agglomerative**
- **K-means (X-means)**
- Model based (e.g. Mixture of Gaussians)
- Affinity propagation
- Spectral
- LDA (Latent Dirichlet Allocation)
- Network/Graph clustering
  - Graph cuts (e.g. min cut - cut min # of edges)
  - Highly connected subgraphs

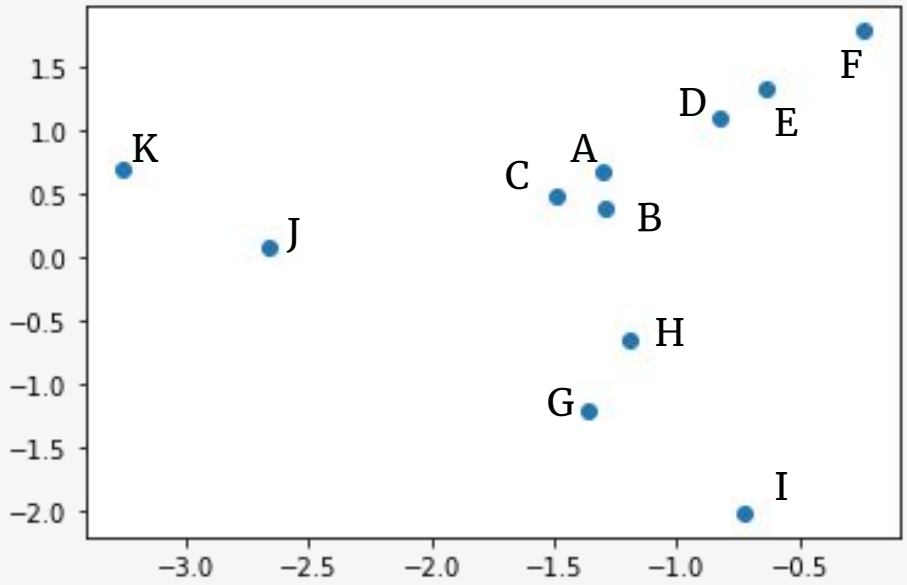
many more....

Agglomerative hierarchical clustering

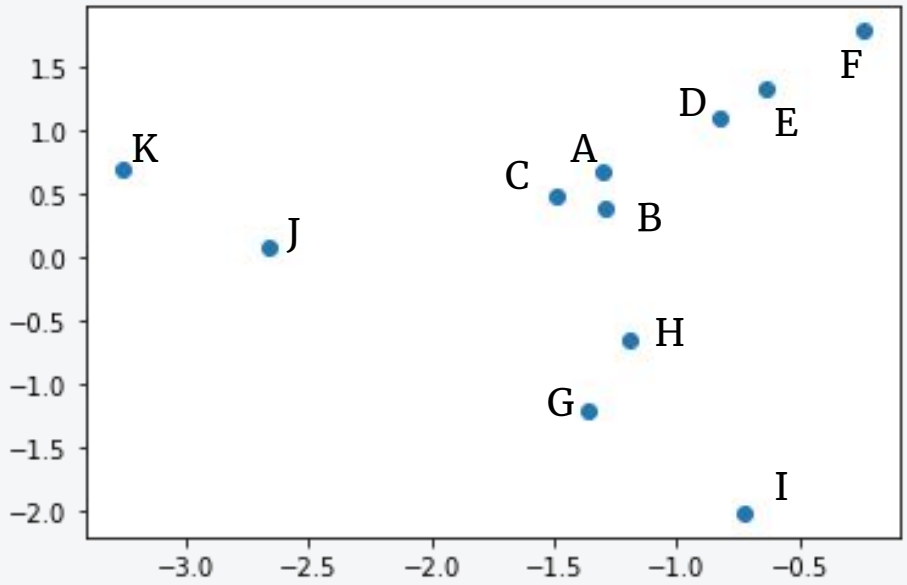
- Start with each point being each own cluster

- Compute similarity between all current clusters
- Use linkage function to combine the closest ones
- Repeat until desired number of clusters or all merged

Agglomerative hierarchical clustering

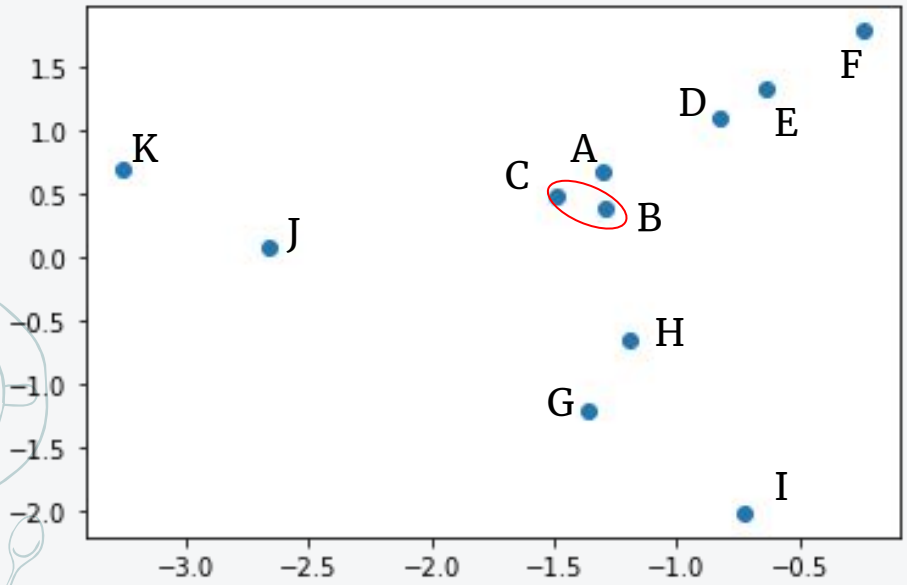


Agglomerative Hierarchical Clustering

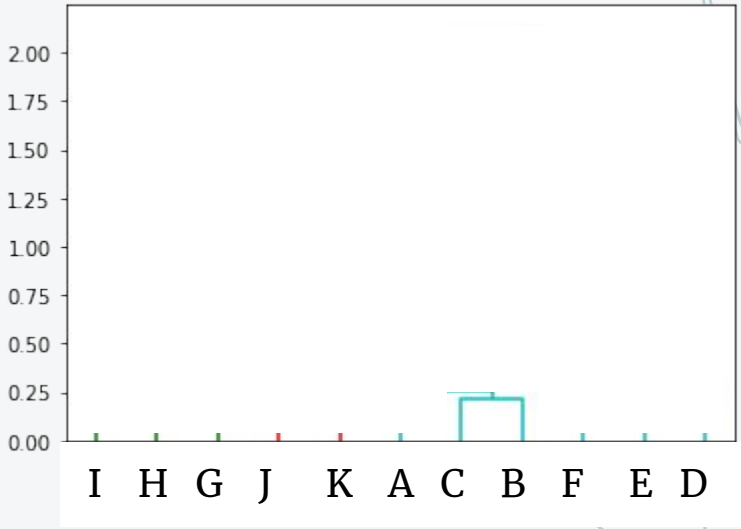


Distance: Euclidean
Linkage: Centroid

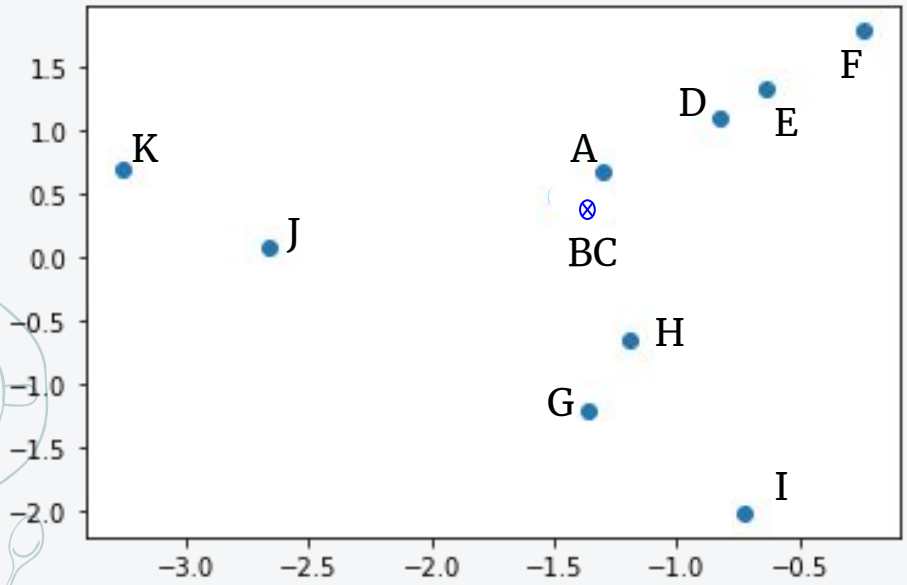
Agglomerative Hierarchical Clustering



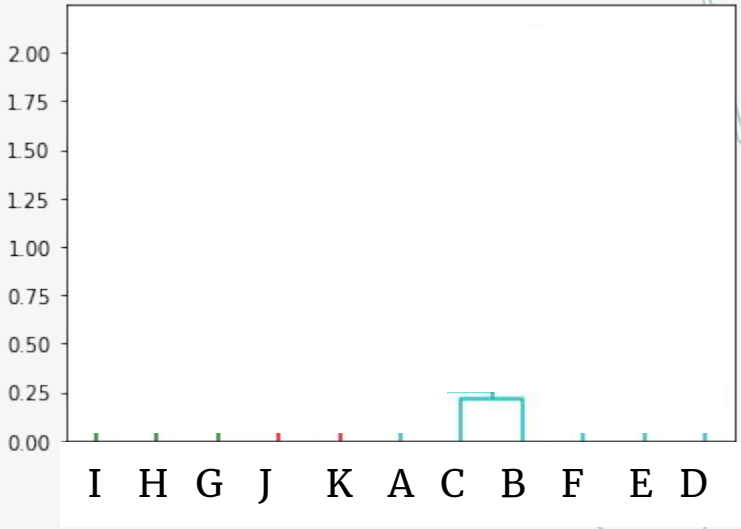
Dendrogram



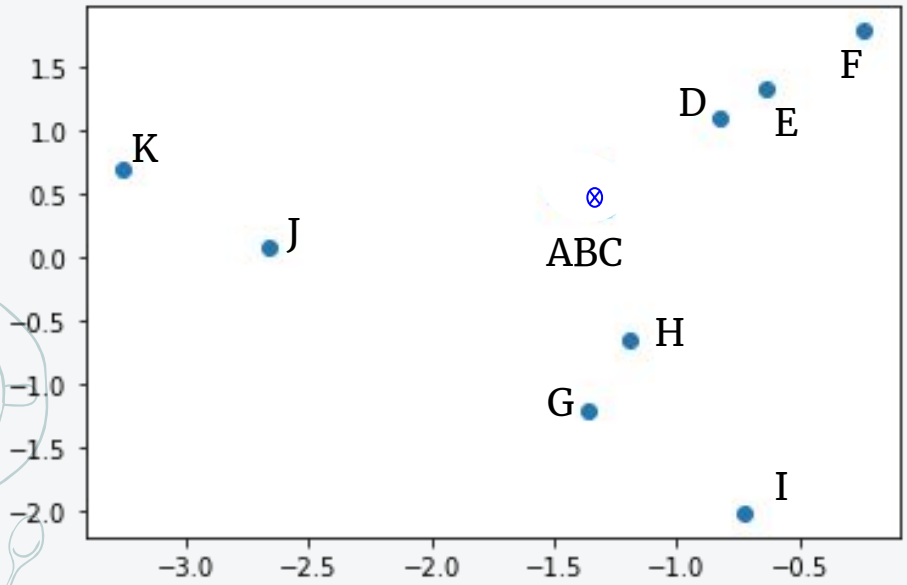
Agglomerative Hierarchical Clustering



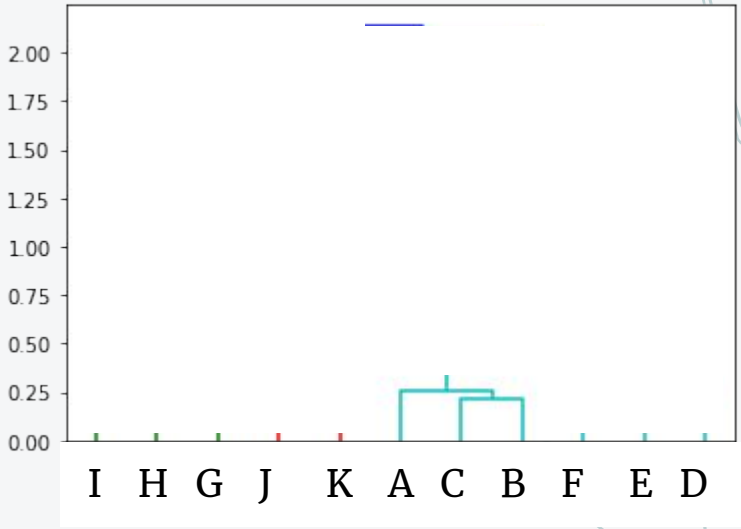
Dendrogram



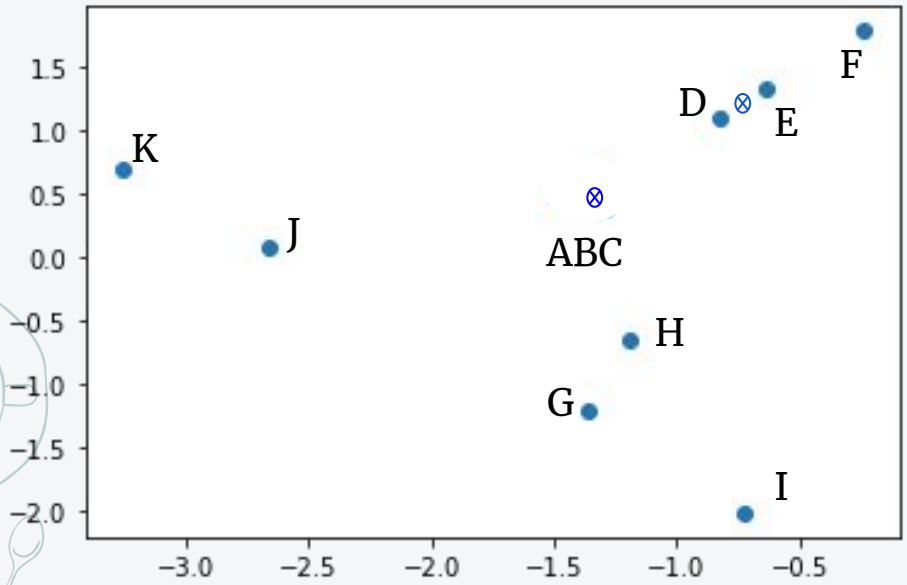
Agglomerative Hierarchical Clustering



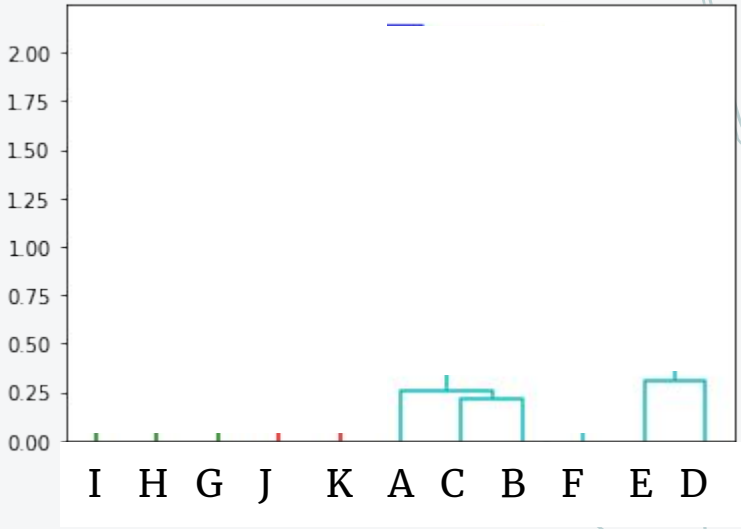
Dendrogram



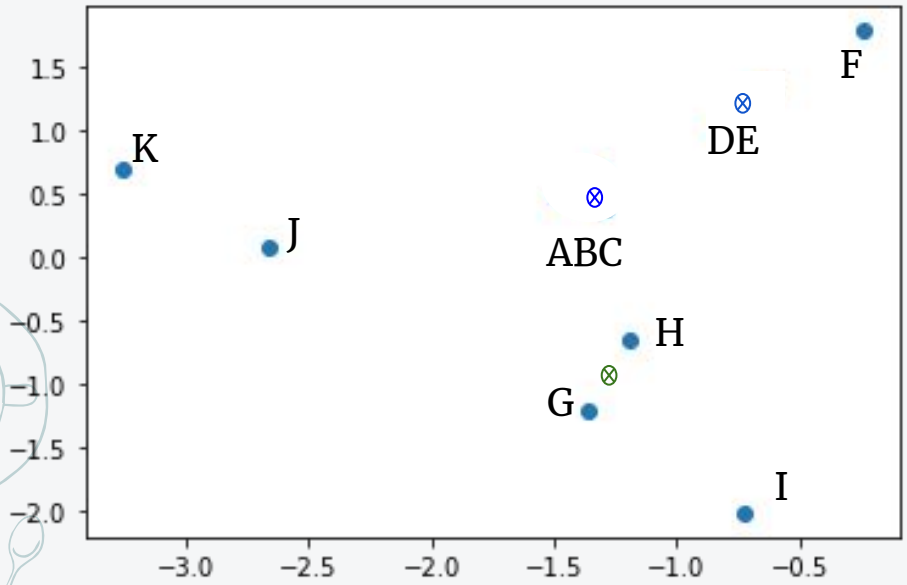
Agglomerative Hierarchical Clustering



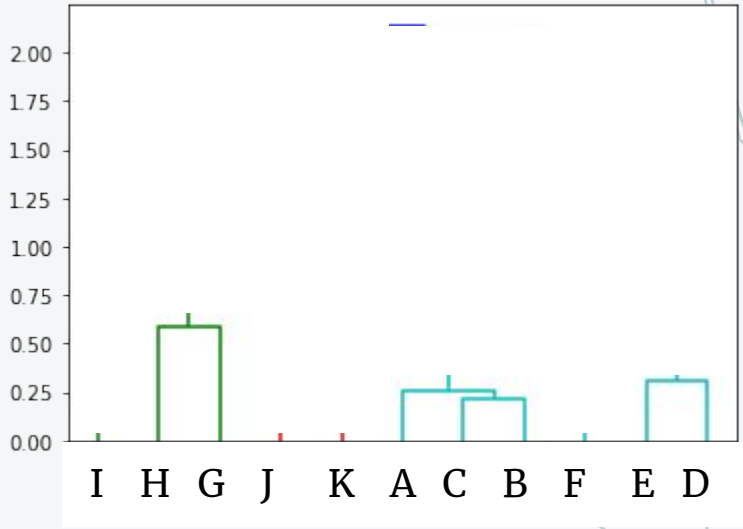
Dendrogram



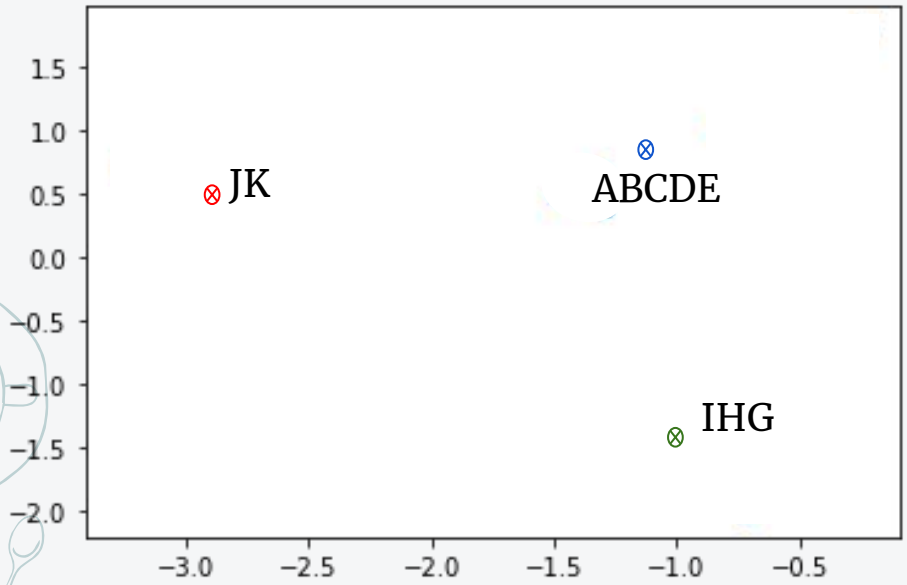
Agglomerative Hierarchical Clustering



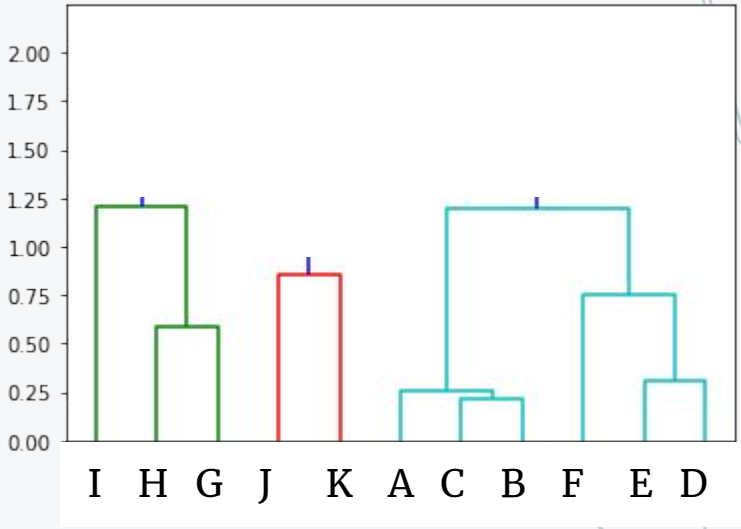
Dendrogram



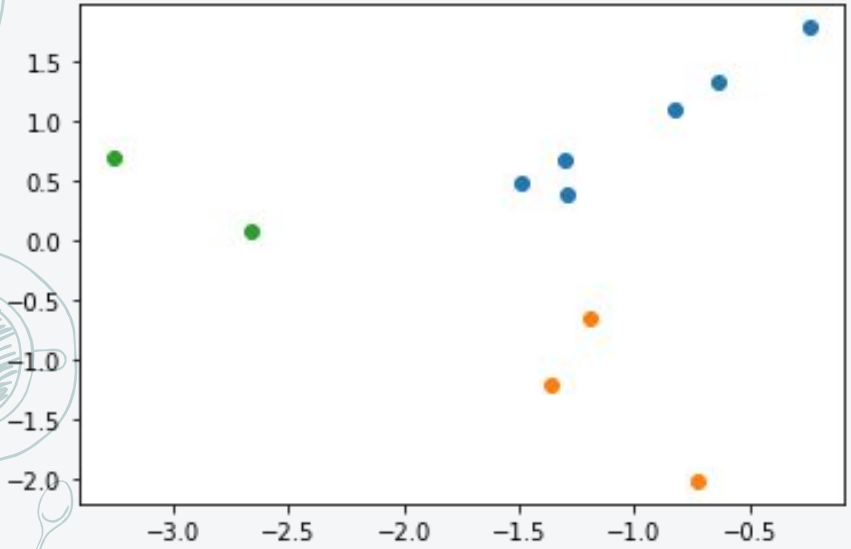
Agglomerative Hierarchical Clustering



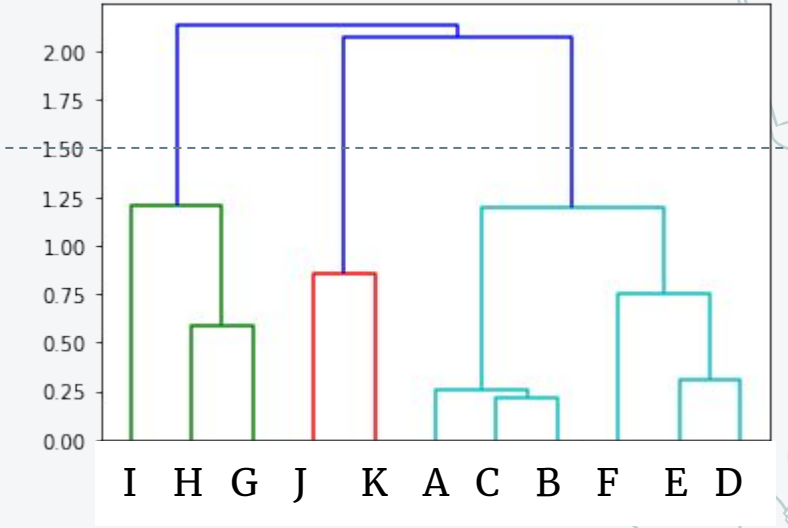
Dendrogram



Agglomerative Hierarchical Clustering

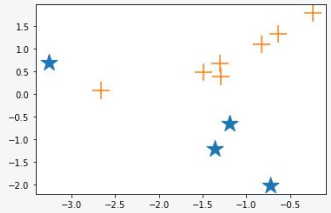


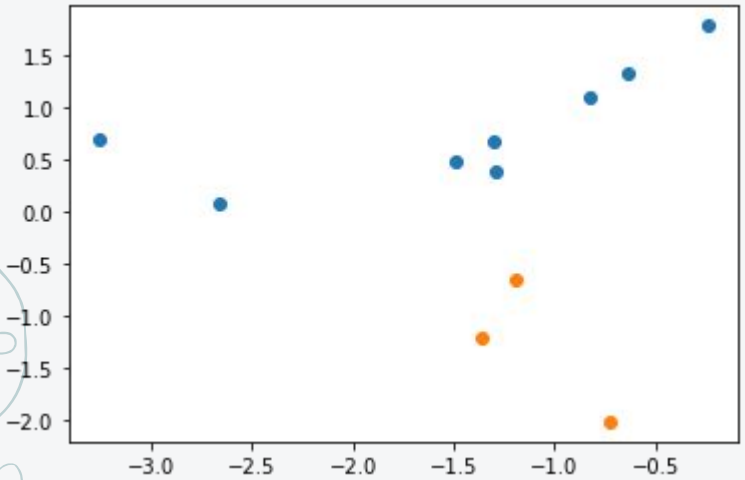
Dendrogram



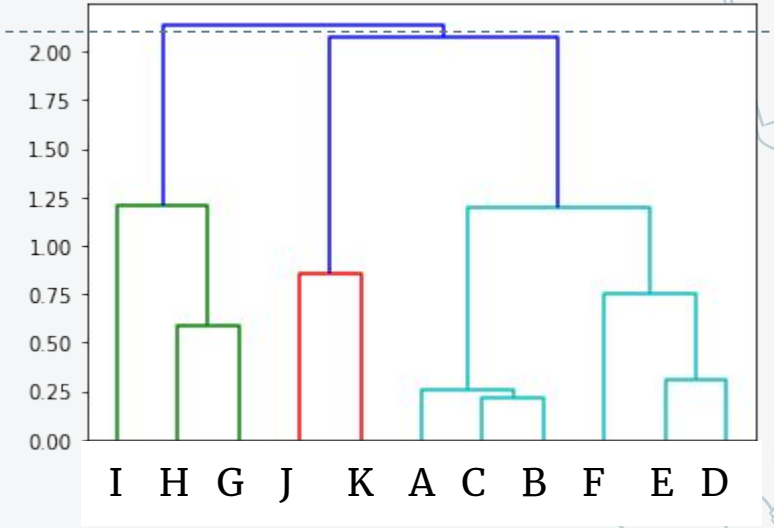
Agglomerative Hierarchical Clustering

K = 2





Dendrogram



What parameters does it take?

What parameters does it take?

Distance - computing how close points are

Linkage - deciding what to merge

Settings: distance

$\vec{x}=\{x_1,x_2,\ldots x_P\}$
$\vec{y}=\{y_1,y_2,\ldots,y_P\}$

- Euclidean (usually default, same as L2 norm) - shortest distance between 2 points $\sqrt{\sum_{i=1}^{P}(x_i-y_i)^2}$

https://towardsdatascience.com/machine-learning-algorithms-part-12-hierarchical-agglomerative-clustering-example-in-python-1e18e0075019

Settings: distance

$\vec{x}=\{x_1,x_2,\ldots,x_P\}$
$\vec{y}=\{y_1,y_2,\ldots,y_P\}$

- Euclidean (usually default, same as L2 norm) - shortest distance between 2 points $\sqrt{\sum_{i=1}^{P}(x_i-y_i)^2}$
- Manhattan distance (same as L1 norm) $\sum_{i=1}^{P}|x_i-y_i|$

https://towardsdatascience.com/machine-learning-algorithms-part-12-hierarchical-agglomerative-clustering-example-in-python-1e18e0075019

Settings: distance

$\vec{x}=\{x_1,x_2,\ldots,x_P\}$
$\vec{y}=\{y_1,y_2,\ldots,y_P\}$

- Euclidean (usually default, same as L2 norm) - shortest distance between 2 points $\sqrt{\sum_{i=1}^{P}(x_i-y_i)^2}$
- Manhattan distance (same as L1 norm) $\sum_{i=1}^{P}|x_i-y_i|$
- Cosine $\frac{\sum_{i=1}^{P}x_iy_i}{\sqrt{\sum_{i=1}^{P}x_i^2}\sqrt{\sum_{i=1}^{P}y_i^2}}$

note: euclidean distance applied to unit normalized vectors will provide the same ordering

https://towardsdatascience.com/machine-learning-algorithms-part-12-hierarchical-agglomerative-clustering-example-in-python-1e18e0075019

Settings: distance

$\vec{x}=\{x_1,x_2,\ldots,x_P\}$
$\vec{y}=\{y_1,y_2,\ldots,y_P\}$

- Euclidean (usually default, same as L2 norm) - shortest distance between 2 points $\sqrt{\sum_{i=1}^{P}(x_i-y_i)^2}$
- Manhattan distance (same as L1 norm) $\sum_{i=1}^{P}|x_i-y_i|$
- Cosine $\frac{\sum_{i=1}^{P}x_iy_i}{\sqrt{\sum_{i=1}^{P}x_i^2}\sqrt{\sum_{i=1}^{P}y_i^2}}$

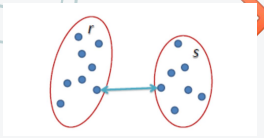
  note: euclidean distance applied to unit normalized vectors will provide the same ordering
- Mahalanobis distance: $d(\vec{x},\vec{y})=\sqrt{(\vec{x}-\vec{y})^T S^{-1}(\vec{x}-\vec{y})}$, where S is covariance

note: if S is an identity matrix, then Mahalanobis reduces to euclidean
If S is diagonal, then Mahalanobis is standardized euclidean

https://towardsdatascience.com/machine-learning-algorithms-part-12-hierarchical-agglomerative-clustering-example-in-python-1e18e0075019

Settings: Linkage

Linkage - takes distance information and groups pairs of objects into clusters based on their similarity

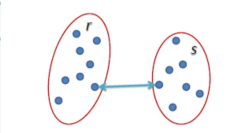


- single - min. of all pairwise distances between the elements in cluster 1 and the elements in cluster 2. Tends to produce long, “loose” clusters.

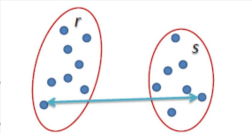
https://towardsdatascience.com/machine-learning-algorithms-part-12-hierarchical-agglomerative-clustering-example-in-python-1e18e0075019

Settings: Linkage

Linkage - takes distance information and groups pairs of objects into clusters based on their similarity



- single - min. of all pairwise distances between the elements in cluster 1 and the elements in cluster 2. Tends to produce long, “loose” clusters.

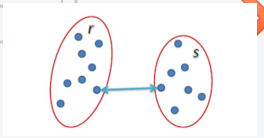


- complete - max. of all pairwise distances “. Tends to produce more compact clusters.

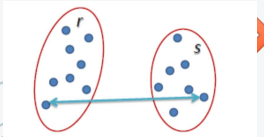
https://towardsdatascience.com/machine-learning-algorithms-part-12-hierarchical-agglomerative-clustering-example-in-python-1e18e0075019

Settings: Linkage

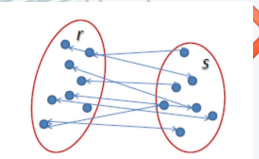
Linkage - takes distance information and groups pairs of objects into clusters based on their similarity



- single - min. of all pairwise distances between the elements in cluster 1 and the elements in cluster 2. Tends to produce long, “loose” clusters.



- complete - max. of all pairwise distances “. Tends to produce more compact clusters.

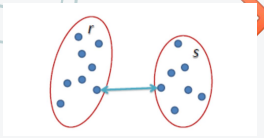


- average - avg. of all pairwise distances

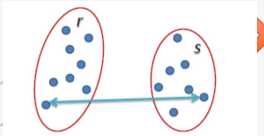
https://towardsdatascience.com/machine-learning-algorithms-part-12-hierarchical-agglomerative-clustering-example-in-python-1e18e0075019

Settings: Linkage

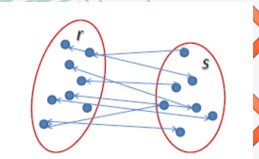
Linkage - takes distance information and groups pairs of objects into clusters based on their similarity



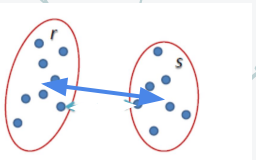
- single - min. of all pairwise distances between the elements in cluster 1 and the elements in cluster 2. Tends to produce long, “loose” clusters.



- complete - max. of all pairwise distances “. Tends to produce more compact clusters.



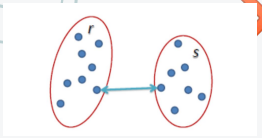
- average - avg. of all pairwise distances
- centroid - distance between cluster centroids



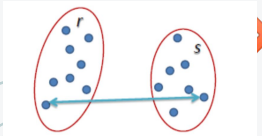
https://towardsdatascience.com/machine-learning-algorithms-part-12-hierarchical-agglomerative-clustering-example-in-python-1e18e0075019

Settings: Linkage

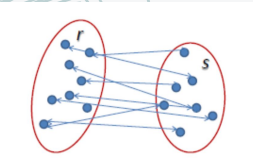
Linkage - takes distance information and groups pairs of objects into clusters based on their similarity



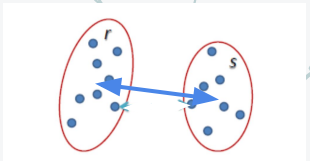
- single - min. of all pairwise distances between the elements in cluster 1 and the elements in cluster 2. Tends to produce long, “loose” clusters.

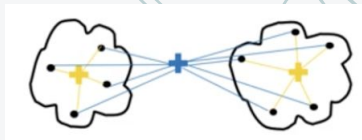


- complete - max. of all pairwise distances “. Tends to produce more compact clusters.



- average - avg. of all pairwise distances
- centroid - distance between cluster centroids
- Ward - minimizes within cluster variance

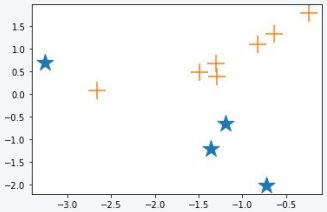




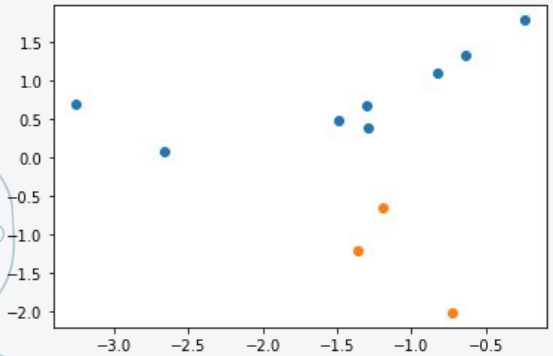
https://towardsdatascience.com/machine-learning-algorithms-part-12-hierarchical-agglomerative-clustering-example-in-python-1e18e0075019

Agglomerative Hierarchical Clustering

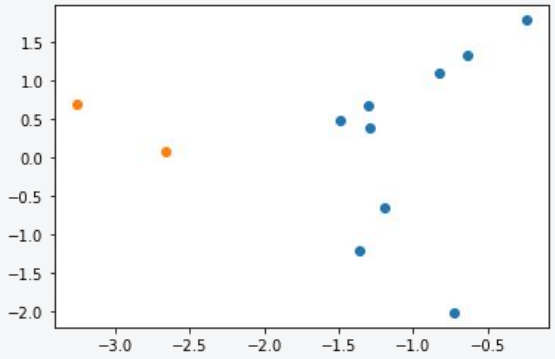
K = 2,
distance= Euclidean



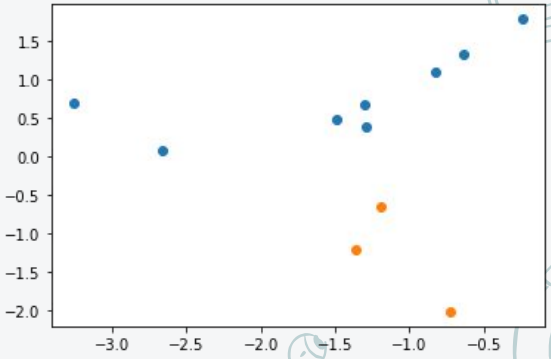
linkage = centroid

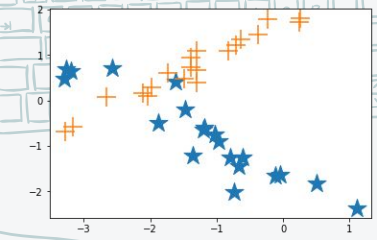


linkage = single



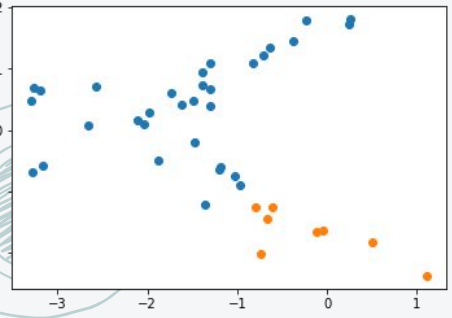
linkage = Ward



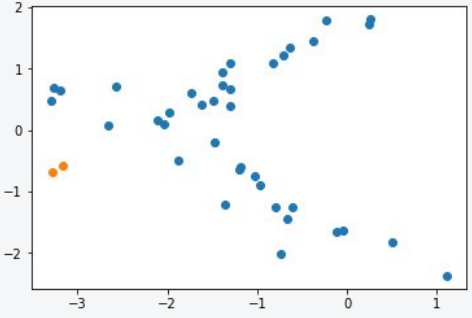


How are we doing on the original bigger dataset?

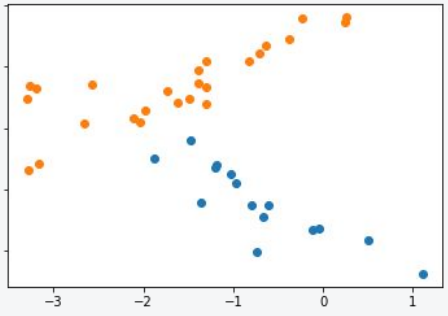
Ward, euclidean
Average, cosine



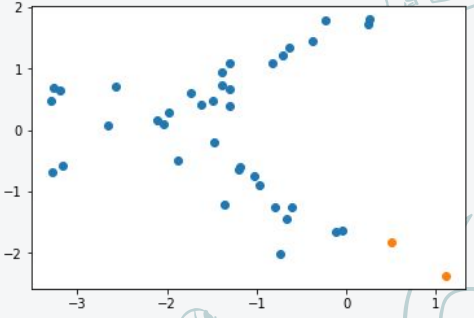
Single, euclidean
single , l1
single, l2
average, l1



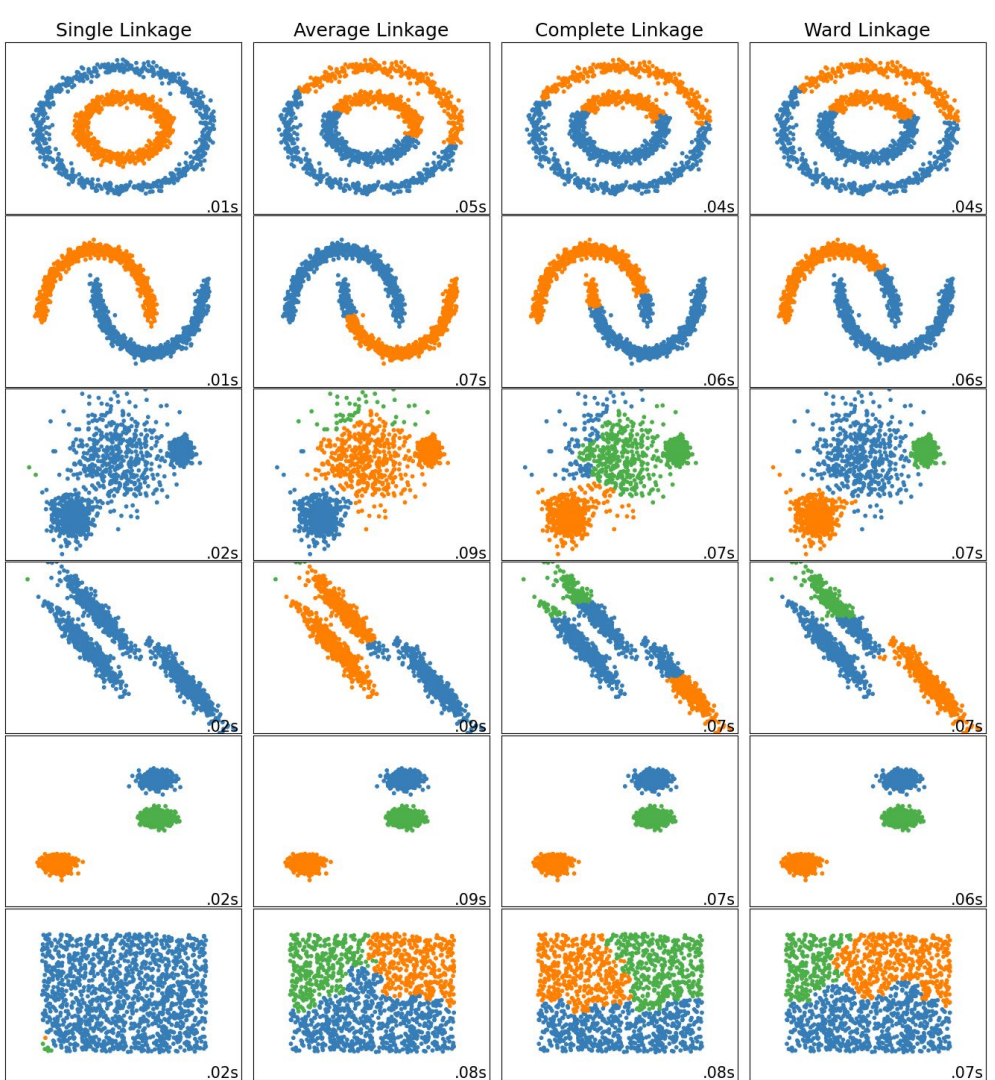
average, l2
average, euclidean



Single, cosine



Comparing Linkages across Datasets



Summary

- Each dataset needs its own distance and linkage
- Deciding on the number of clusters can be done after clustering is finished

K-means - Algorithm

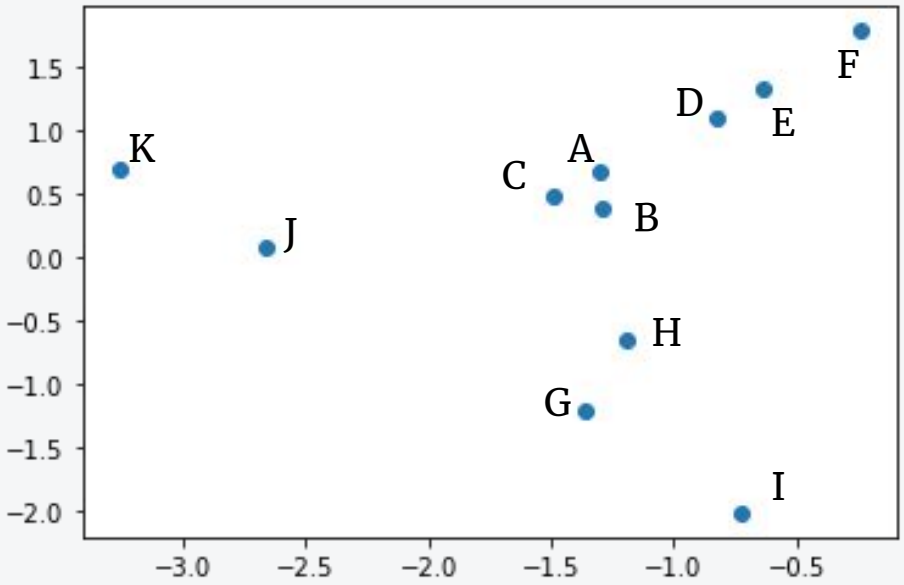
Decide on the number of clusters K

Pick K centroids at random

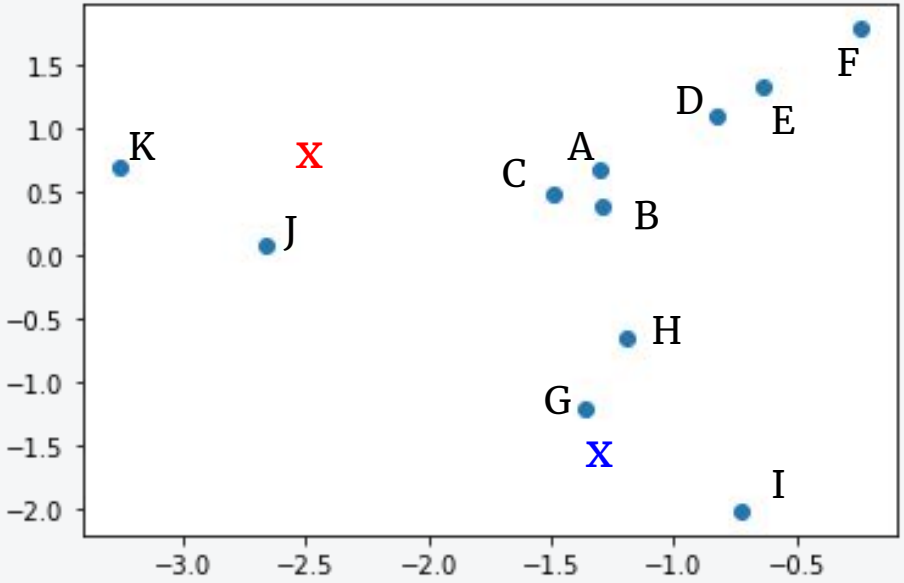
Repeat until convergence:

1) compute distance from each point to each centroid and assign points to the cluster with the closest centroid
2) Re-compute the centroid for the current cluster

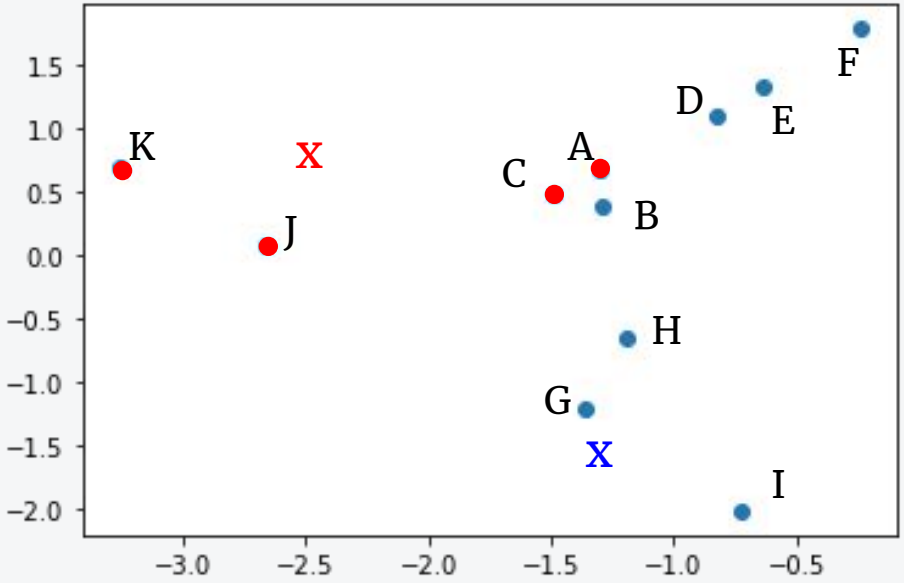
K-Means



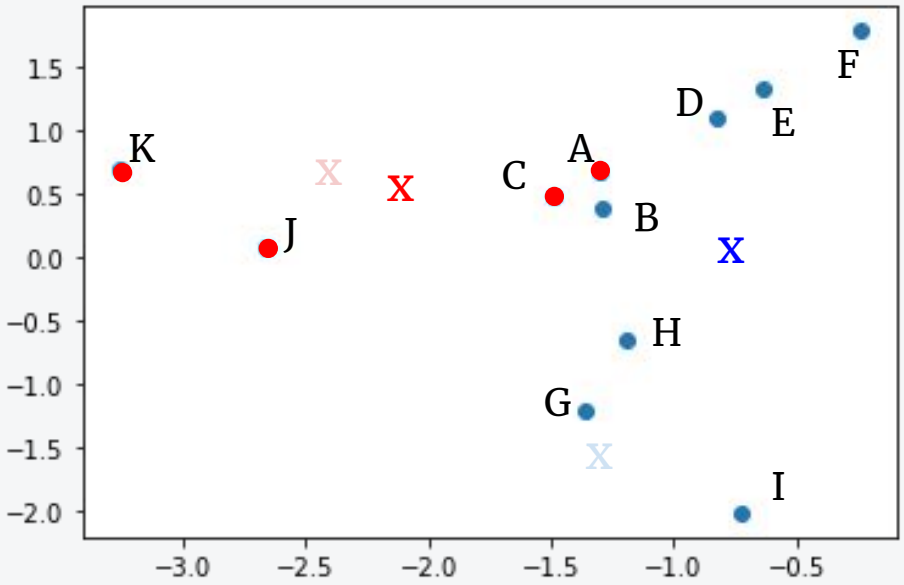
K-Means



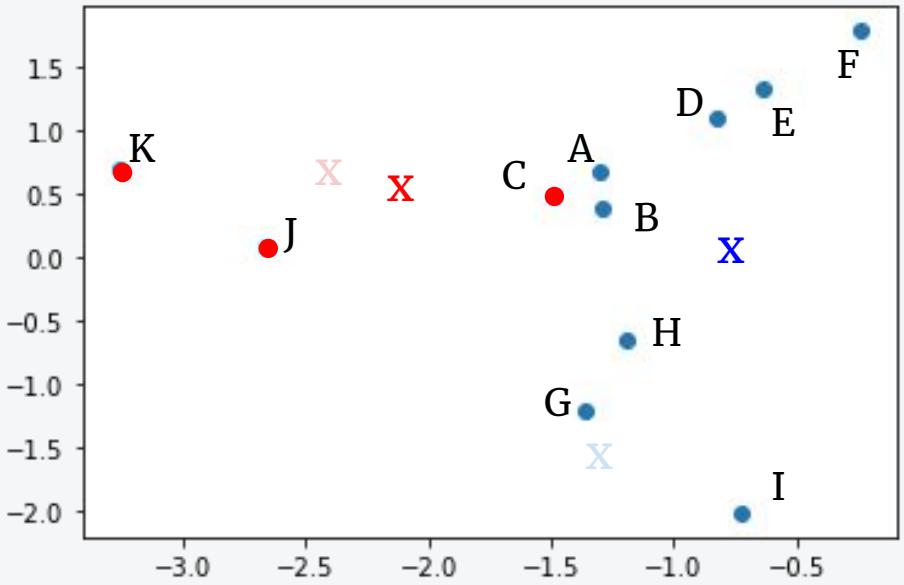
K-Means



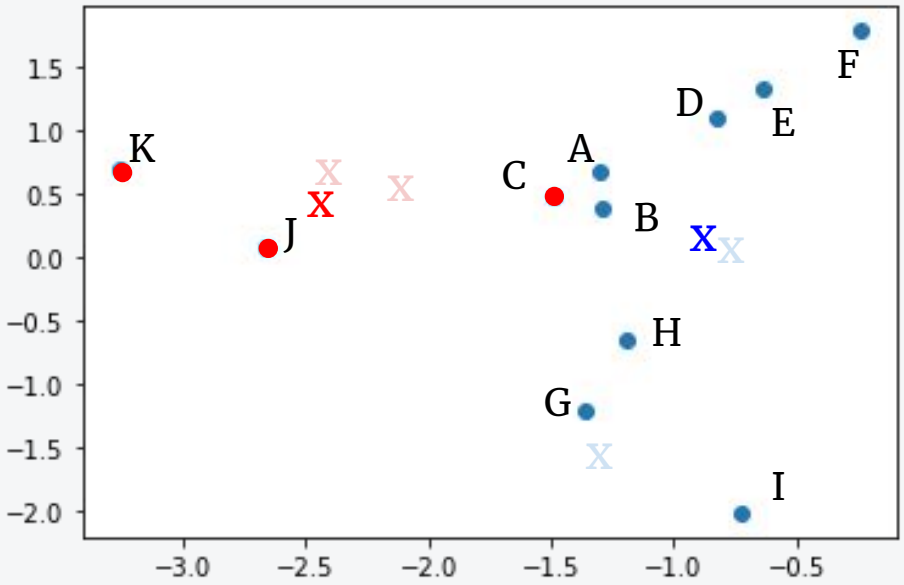
K-Means



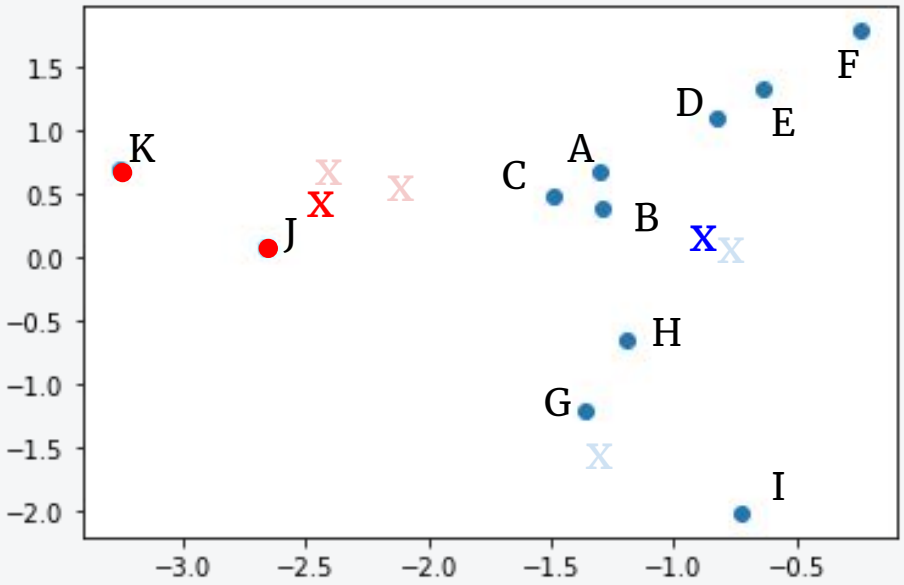
K-Means



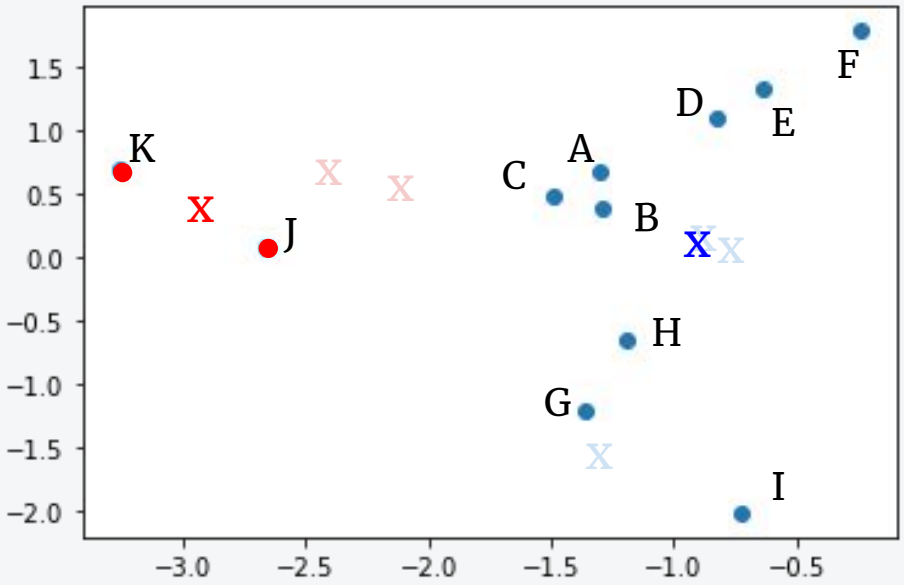
K-Means



K-Means

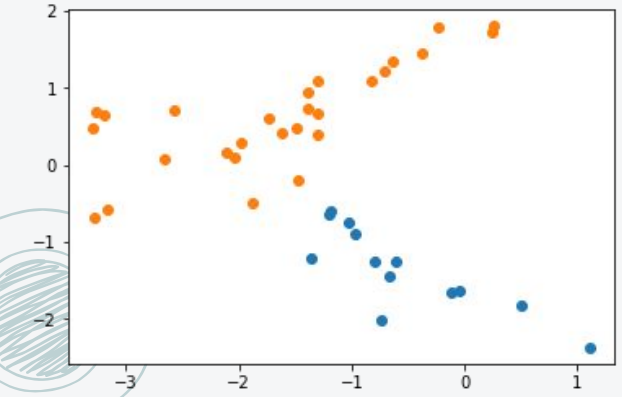


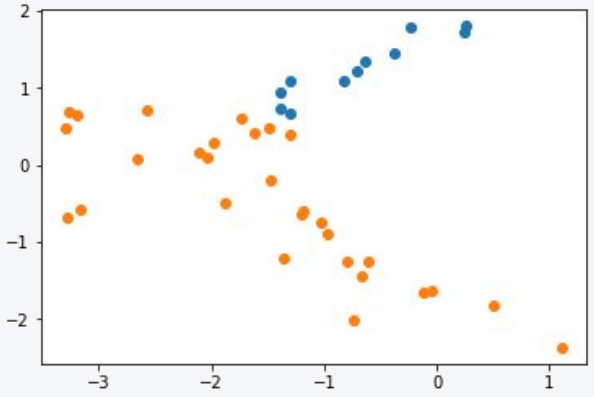
K-Means

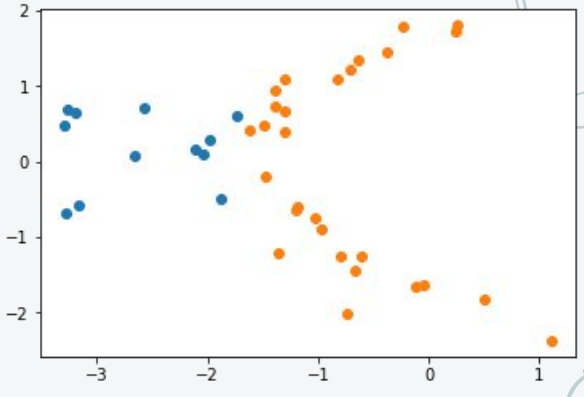


K-Means

Depends on initial conditions







How to make K-Means more robust?

How to make K-Means more robust?

- Run many times (scikit version default is 10)
- Have a better than random initialization
  - Pick data points as initializations
  - k-means++
  - Naive sharding

How to make K-Means more robust? k-means++

1. Choose one center uniformly at random among the data points.
2. For each data point x not chosen yet, compute D(x), the distance between x and the nearest center that has already been chosen.
3. Choose one new data point at random as a new center, using a weighted probability distribution where a point x is chosen with probability proportional to D(x).
4. Repeat Steps 2 and 3 until k centers have been chosen.
5. Now that the initial centers have been chosen, proceed using standard k-means clustering.

K-MEANS

Advantages:

Fast

Makes relatively few assumptions

Stable when robust initialization is used

Disadvantages:

Depends on the initialization

Can get stuck in local minimum

A bit about selecting the number of clusters

General approach

- Apply clustering for k = 1..K (e.g. K = 10)
- For each k, compute a statistic
- Plot statistic as a function of k

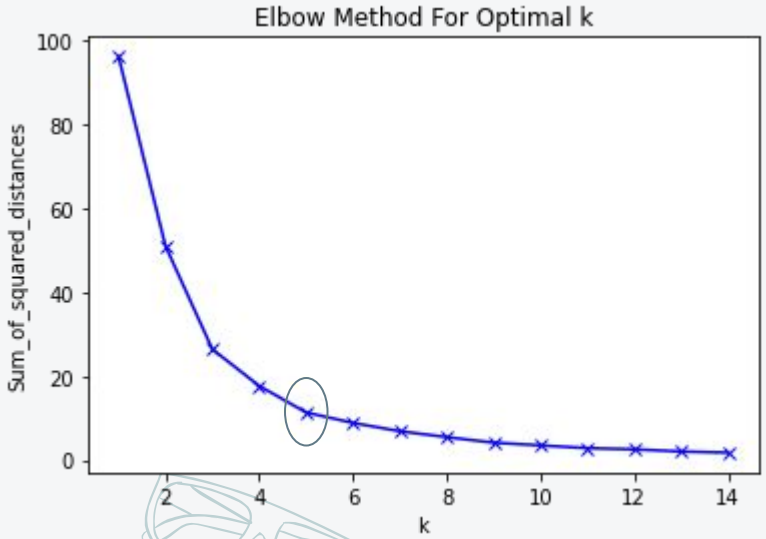
VARIOUS STATISTICS FOR SELECTING THE NUMBER OF CLUSTERS

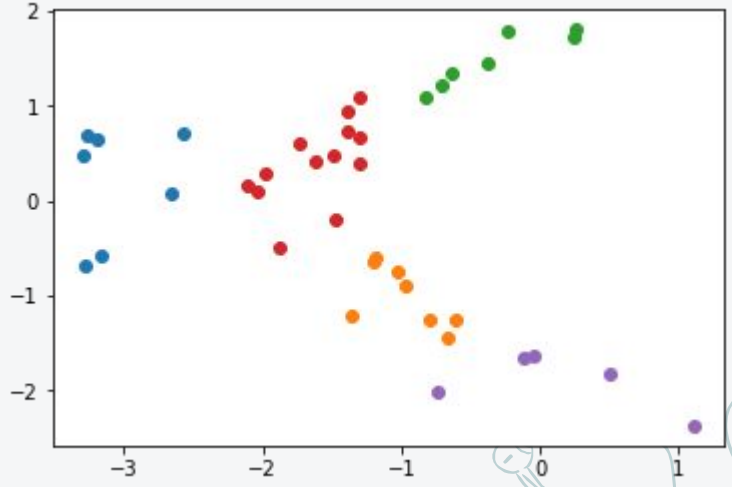
- Silhouette
- Gap
- Elbow
- Information Criterion (depend on the likelihood)
  - AIC
  - BIC
- ...

Elbow

Elbow method- heuristic

Elbow in the plot of the sum of squared distances to the nearest centre





Silhouette Score

For each data point i in cluster C_i, let

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i, j)$$ - average distance for point i to other points within its cluster

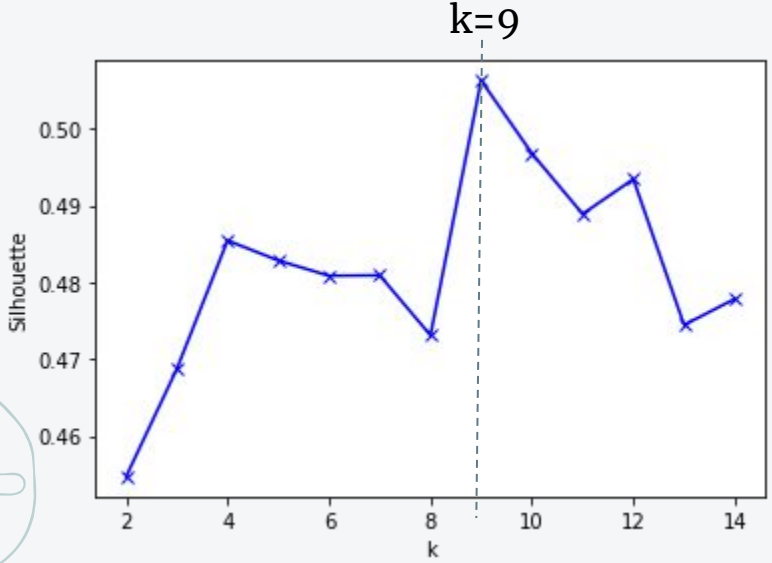
$$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i, j)$$ - min (across clusters) of the averages of intra-cluster distances

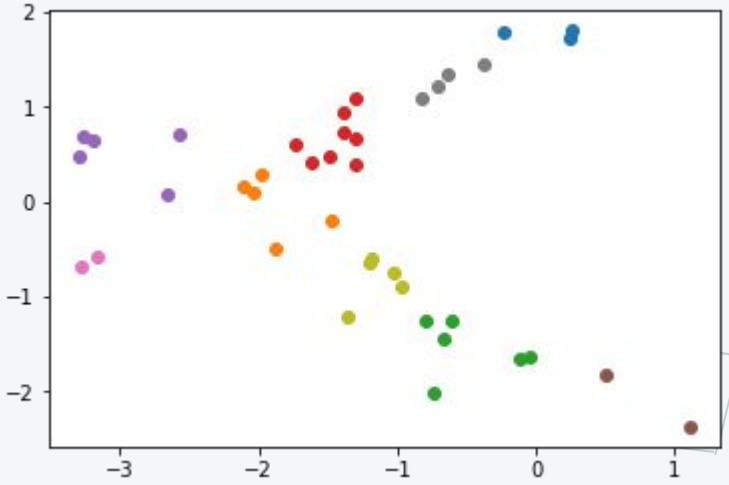
Silhouette: $$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$, Silhouette Coefficient: $$SC = \max_k \tilde{s}(k)$$

average over all points

Kauffman and Rousseeuw (1990)

Silhouette Score





Kauffman and Rousseeuw (1990)

GAP STATISTIC

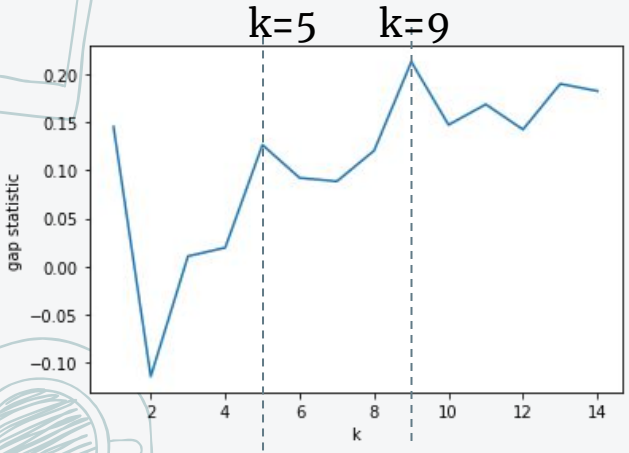
$D_r = \Sigma_{i,j \in Cr} d_{ij}$ - sum of pairwise distances in cluster i

$W_k = \Sigma_{r=1..k} 1/(2n_r) D_r$ - pooled within cluster sum of distances

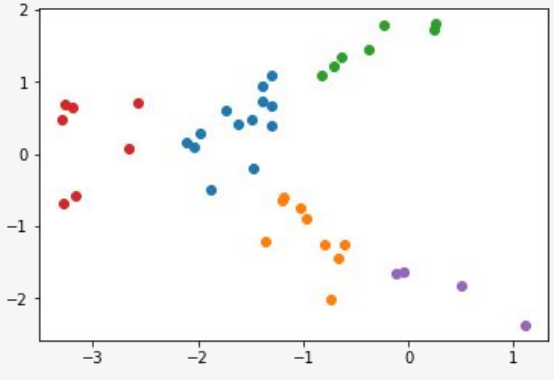
Main idea: compare $\log(W_k)$ to expectation under the null distribution $\text{Gap}_n(k) = E_n^*\{\log(W_k)\} - \log(W_k)$

Tibshirani et al (2000)

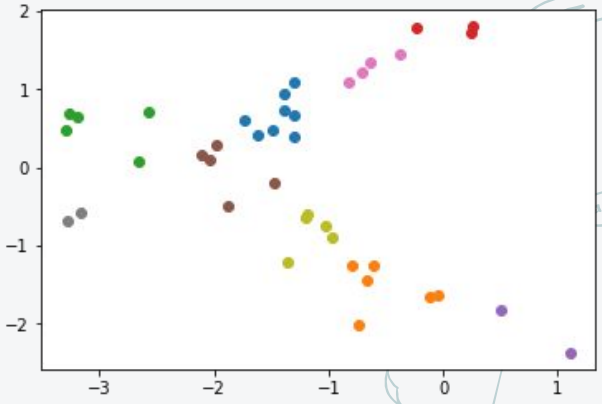
GAP STATISTIC



k=5

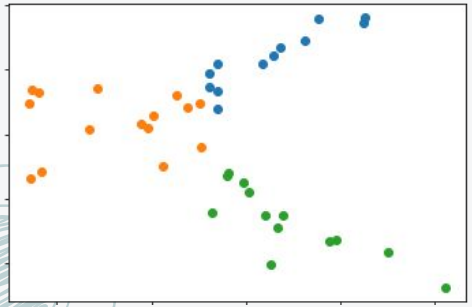


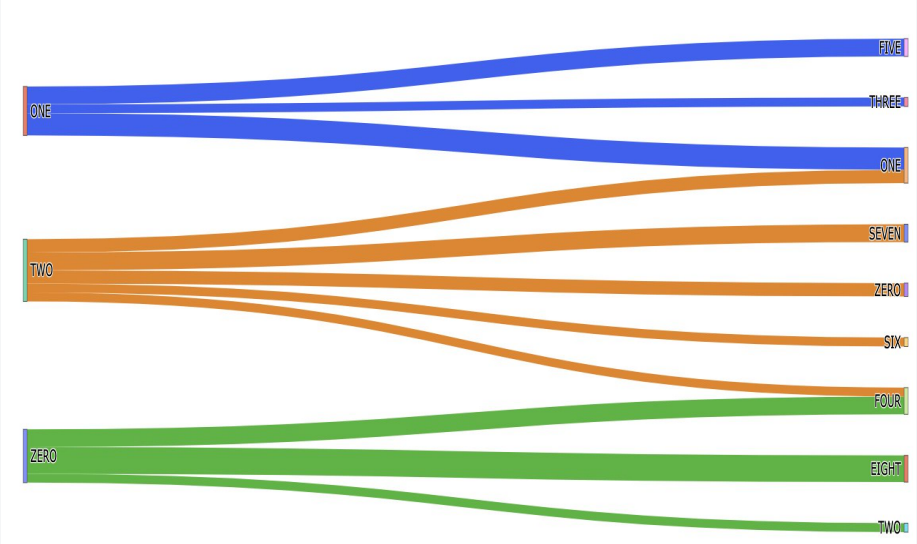
k=9



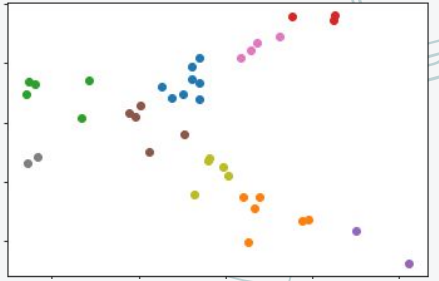
Alluvial (SANKY) plot

k=3





k=9



SUMMARY

- There are MANY clustering methods
- There is no one perfect method that works for every dataset
- There are many ways to select the number of clusters
- Examine multiple methods for each dataset
- Visualizations help to analyze clustering

Reading

5 common clustering methods:

https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68

10 clustering methods in python:

https://machinelearningmastery.com/clustering-algorithms-with-python/

Original k-means++ presentation:
http://theory.stanford.edu/~sergei/slides/BATS-Means.pdf

Original gap statistic paper:

https://statweb.stanford.edu/~gwalther/gap

Sanky plots:
https://towardsdatascience.com/sankey-diagram-basics-with-pythons-plotly-7a13d557401a