Lectures 12, 13:
Unsupervised learning, cluster analysis

Data Science 2
CS 109b, Stat 121b, AC 209b, E-109b

Mark Glickman Pavlos Protopapas

Reading: James et al., chapter 10.

Basics of unsupervised learning:

- Want to discover subgroups among variables or observations, but not in how they relate to a response variable.
- Usually no precise goal of analysis. Often can be used as a descriptive tool.
- Often easier to obtain data without a response variable.

We will focus on cluster analysis.

Example applications:

- Collect breast cancer patients; want to find subgroups according to similar gene expressions.
- Obtain browser usage and purchasing history of online shoppers; want to find subgroups that relate to their browsing and purchasing patterns.
- Want to group movies according to ratings assigned by movie viewers.

Difference in attitude between PCA and clustering:

Both are multivariate techniques, but with slightly different goals.

- PCA (principal components analysis) attempts to find a low-dimensional representation of the observations that explains a large fraction of the variation. Often used for variable reduction (as you have seen in CS109a), but can also be helpful for visualization in seeing groupings of data.
- Clustering instead seeks homogeneous subgroups among the observations.

Cluster analysis outline:

- Inter-observation distances

- Partition-based clustering
- Hierarchical clustering
- Diagnostics, and optimizing the number of clusters
- Plots, plots, plots and more plots

Clustering:

- Cluster analysis consist of techniques for finding subgroups or clusters in a multivariate data set.
- Want to partition the data into distinct groups so that observations within each group are similar to each other.
- Need to define concretely what it means for observations to be similar or different. This is usually domain-specific.

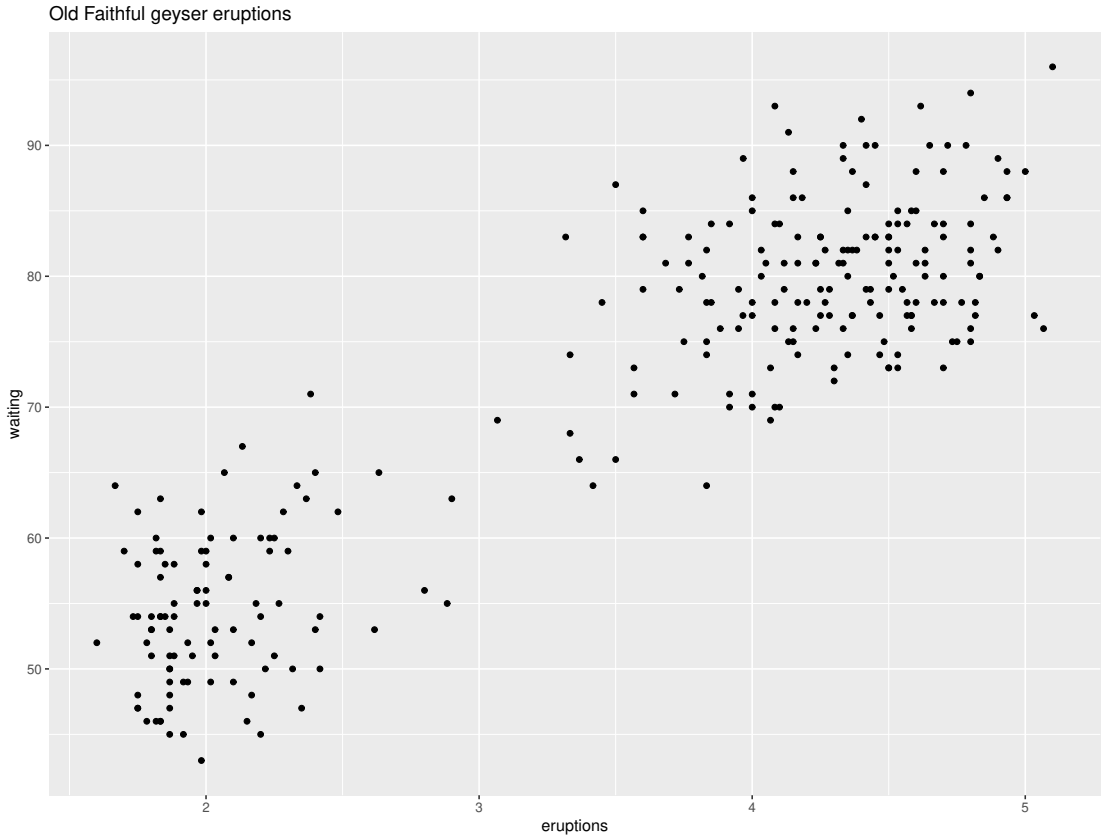
Notation:

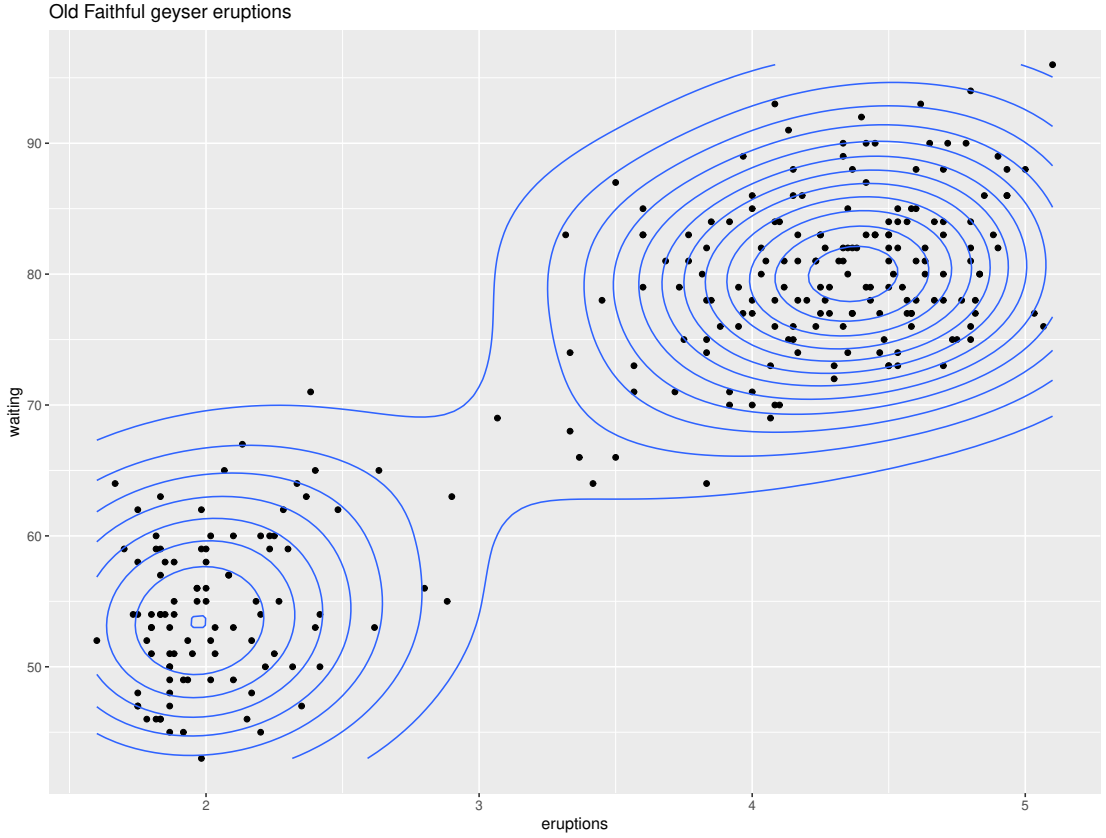
- Assume n observations are measured on p variables or features.
- Let X be an $n \times p$ matrix of data. Let X_{ij}, for $i = 1, \ldots, n$ and $j = 1, \ldots, p$, be the value of feature j on observation i.
- Interested in finding subsets of rows of X that are similar to each other, which will form a "cluster."

Simple example: Eruptions of Old Faithful

Data were collected on time between eruptions and the duration of eruptions for the Old Faithful geyser at Yellowstone National Park.

	eruptions	waiting
count	272.000000	272.000000
mean	3.487783	70.897059
std	1.141371	13.594974
min	1.600000	43.000000
25%	2.162750	58.000000
50%	4.000000	76.000000
75%	4.454250	82.000000
max	5.100000	96.000000


Old Faithful geyser eruptions
waiting
eruptions
90
80
70
60
50
2
3
4
5


Old Faithful geyser eruptions
waiting
eruptions
90
80
70
60
50
2
3
4
5

Two main types of approaches to clustering:

- Partitioning clustering - specify the number of clusters in advance, and then invoke an algorithm to partition the data
- Hierarchical clustering - iteratively merge (or divide) the data typically one observation at a time, and then decide on partitions afterward

Distances between observations: The building block of clustering

- The choice of distance measures is a critical step in clustering. The pairwise distance calculation can influence the shape of the clusters.
- Observations to be clustered can be non-standard (e.g., pictures, audio signals), so the process of computing distances first followed by clustering can be an effective approach.

Assume n observations are measured on p variables or features. Let X_{ij}, for $i = 1, \ldots, n$ and $j = 1, \ldots, p$, be the value of feature j on observation i.

Two common distance measures: For observations $\boldsymbol{x}_i$ and $\boldsymbol{x}_k$ (of length p)

$$d_{Euc}(\boldsymbol{x}_i, \boldsymbol{x}_k) = \sqrt{\sum_{j=1}^{p}(X_{ij} - X_{kj})^2} \quad \text{Euclidean distance } (L_2)$$

$$d_{Man}(\boldsymbol{x}_i, \boldsymbol{x}_k) = \sum_{j=1}^{p}|X_{ij} - X_{kj}| \quad \text{Manhattan distance } (L_1)$$

Other distance measures: Correlation-based measures

$$d_{Cor}(\boldsymbol{x}_i, \boldsymbol{x}_k) = 1 - \frac{\sum_{j=1}^{p}(X_{ij} - \bar{x}_i)(X_{kj} - \bar{x}_k)}{\sqrt{\sum_{j=1}^{p}(X_{ij} - \bar{x}_i)^2 \sum_{j=1}^{p}(X_{kj} - \bar{x}_k)^2}} \quad \text{Pearson correlation distance}$$

$$d_{Spear}(\boldsymbol{x}_i, \boldsymbol{x}_k) = 1 - \frac{\sum_{j=1}^{p}(W_{ij} - \bar{w}_i)(W_{kj} - \bar{w}_k)}{\sqrt{\sum_{j=1}^{p}(W_{ij} - \bar{w}_i)^2 \sum_{j=1}^{p}(W_{kj} - \bar{w}_k)^2}} \quad \text{Spearman correlation distance}$$

where W_{ij} are the ranks of X_{ij} for feature j, and $\bar{w}_i$ is the average of the ranks for observation i.

Correlation distances are often used for micro-array analyses.

Spearman correlation is used when outliers might be a concern in the data.

Distance and scaling:

Consider a data set that looks like the following:

```
     X1      X2     X3      X4
435201    0.04   0.21    0.35
839940    0.01   0.33    0.21
582048    0.23   0.19    0.25
390392    0.18   0.30    0.19
...        ...    ...     ...
298989    0.15   0.23    0.27
```

The distance between any two observations is basically determined by feature 1. This is an undesired consequence of variables having different scales.

Usually want to ensure each variable has equal contribution to the clustering algorithm, and therefore the distance computation.

Solution: Standardize the variables prior to computing distances.

Standardizing usually just involves transforming each X_{ij} by

$$\frac{X_{ij} - \text{center}(\boldsymbol{x}_j)}{\text{scale}(\boldsymbol{x}_j)}$$

where $\text{center}(\boldsymbol{x}_j)$ can be the sample mean or median of the j-th variable, and $\text{scale}(\boldsymbol{x}_j)$ can be the sample standard deviation or mean absolute deviation of the j-th variable.

Main example: Violent crime rates by State

This data set reports arrests per 100,000 residents for assault, murder, and rape in each of the 50 US states in 1973. Also given is the percent of the population living in urban areas.

Summary:

```
         Murder     Assault   UrbanPop       Rape
count  50.00000   50.000000  50.000000  50.000000
mean    7.78800  170.760000  65.540000  21.232000
std     4.35551   83.337661  14.474763   9.366385
min     0.80000   45.000000  32.000000   7.300000
25%     4.07500  109.000000  54.500000  15.075000
50%     7.25000  159.000000  66.000000  20.100000
75%    11.25000  249.000000  77.750000  26.175000
max    17.40000  337.000000  91.000000  46.000000
```

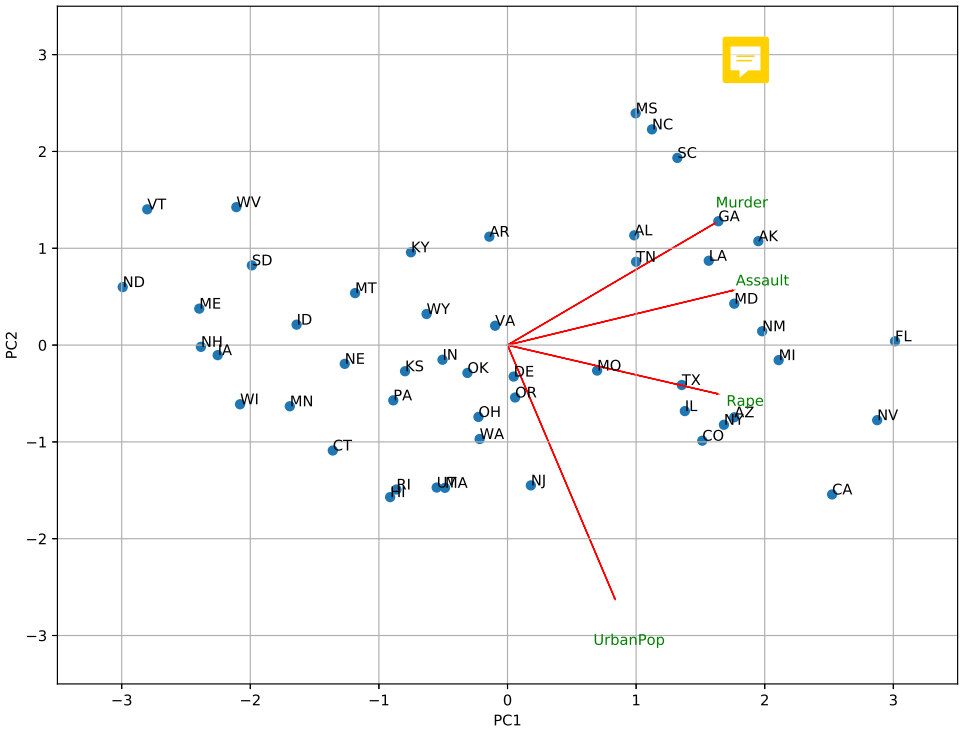


Figure contains a scatter plot of the first two principal components.

- Each state is plotted according to its first two principal components.
- The red arrows indicate the first two PC loading vectors (axes on the top and right). For example, the loading for `Rape` on the first PC is 0.543, and 0.167 on the second PC [the word `Rape` is centered at (0.543, 0.167)].
- This figure is known as a "biplot" as it displays both the PC scores and the PC loadings.

Distances and violent crimes:

First few randomly chosen observations

State	Murder	Assault	UrbanPop	Rape	StateAbbrv
Hawaii	5.3	46	83	20.2	HI
Indiana	7.2	113	65	21.0	IN
New Mexico	11.4	285	70	32.1	NM
Washington	4.0	145	73	26.2	WA
Maine	2.1	83	51	7.8	ME
Alabama	13.2	236	58	21.2	AL

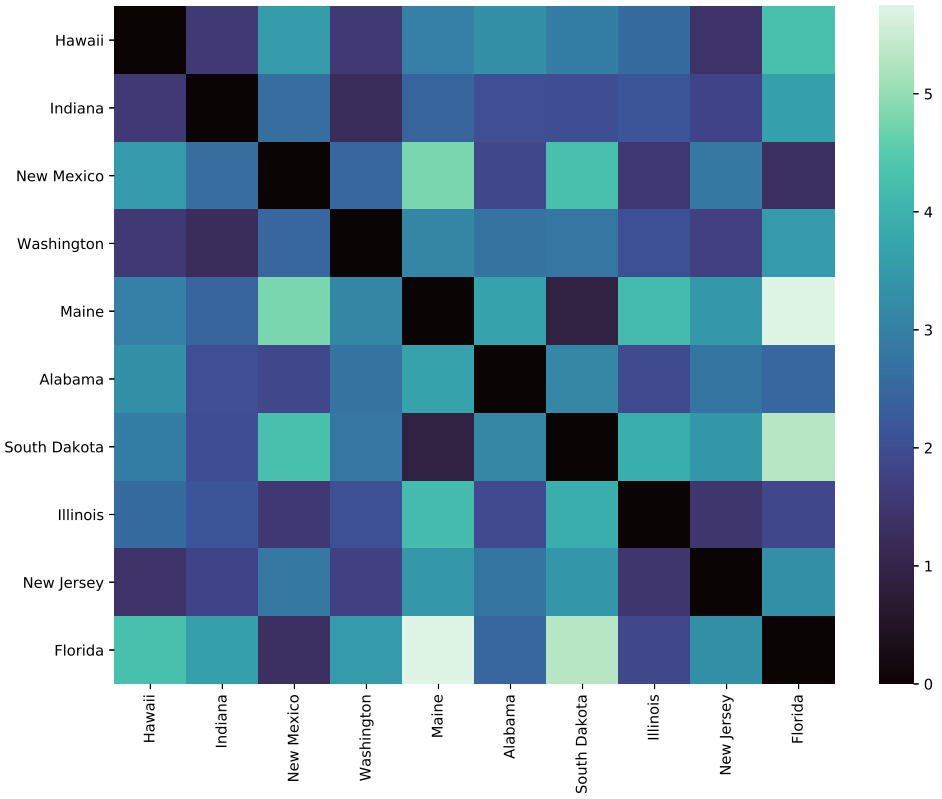
Now rescale the observations (subtract mean, divide by std dev):

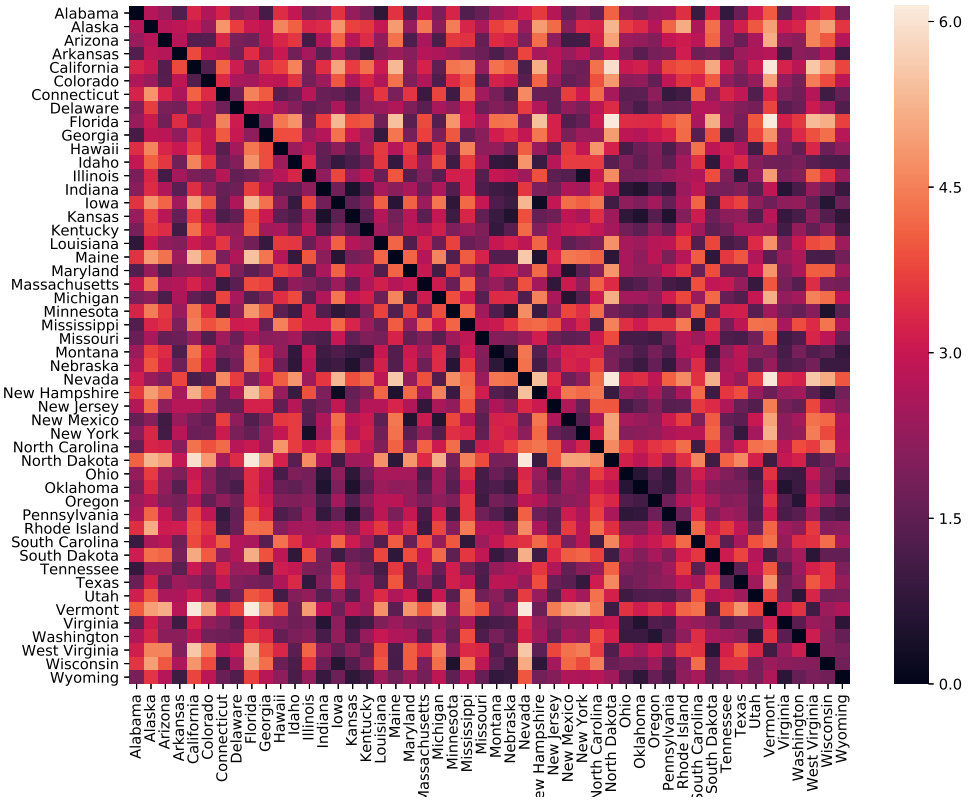
```
     0      1      2      3       State
 -0.65  -1.39   0.95  -0.19      Hawaii
 -0.20  -0.66  -0.34  -0.08     Indiana
  0.81   1.21   0.02   1.45  New Mexico
 -0.96  -0.31   0.24   0.64  Washington
 -1.41  -0.98  -1.33  -1.91       Maine
  1.24   0.68  -0.84  -0.06     Alabama
```

Euclidean distances on scaled observations:

```
        State
0      Hawaii
1     Indiana
2  New Mexico
3  Washington
4       Maine

array([[0. , 1.5, 3.5, 1.6, 3. ],
       [1.5, 0. , 2.6, 1.2, 2.4],
       [3.5, 2.6, 0. , 2.5, 4.8],
       [1.6, 1.2, 2.5, 0. , 3.1],
       [3. , 2.4, 4.8, 3.1, 0. ]])
```


Hawaii
Indiana
New Mexico
Washington
Maine
Alabama
South Dakota
Illinois
New Jersey
Florida
0
1
2
3
4
5



Clustering via partitioning methods:

Basic idea - specify the number of clusters into which the data will be partitioned, and then perform computation to group data so that

1. observations within clusters are similar (low distances/dissimilarities), and
2. observations in different clusters are dissimilar (high distances/dissimilarities)

Separately can address the optimal number of clusters.

K-means clustering:

Let $C_1, \ldots, C_K$ denote sets containing the indices of observations within the K clusters such that

1. $C_1 \cup C_2 \cup \cdots \cup C_K = \{1, 2, \ldots, n\}$, and
2. $C_k \cap C_\ell = \varnothing$ for $k \neq \ell$.

Let

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (X_{ij} - X_{i'j})^2$$

be the within-cluster variation.

An interesting mathematical fact that we will use shortly is that

$$W(C_k) = 2 \sum_{i \in C_k} \sum_{j=1}^{p} (X_{ij} - \bar{x}_{jk})^2$$

where $\bar{x}_{jk}$ is the sample mean of the X_{ij} for $i \in C_k$.

Goal of K-means clustering:

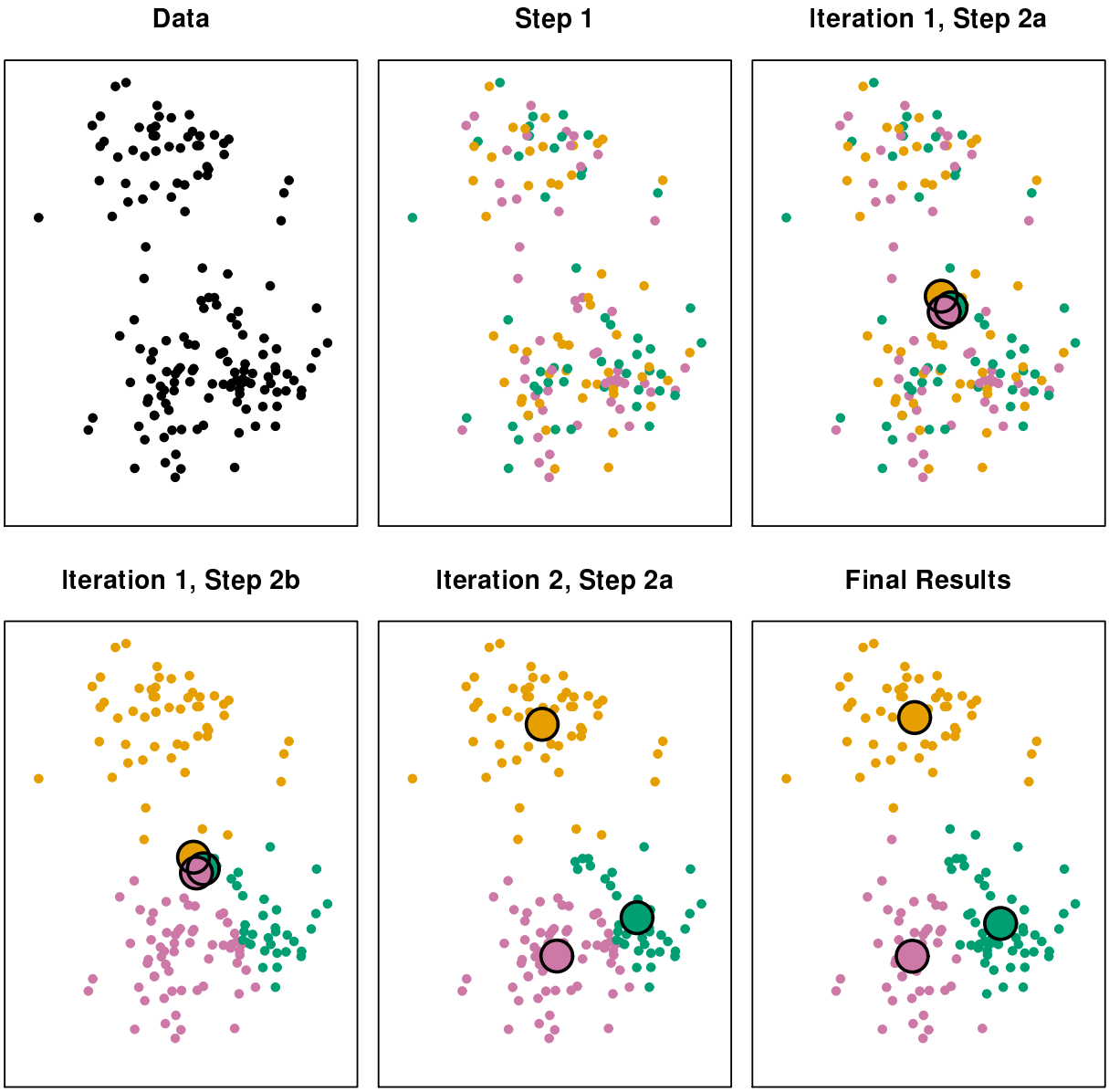
Determine $C_1, \ldots, C_K$ such that the expression

$$\sum_{k=1}^{K} W(C_k)$$

is minimized.

K-means clustering algorithm:

1. Randomly assign each observation to one of K clusters at random.
2. Repeat the following two steps until clusters do not change:
   (a) For each cluster k, compute the cluster centroid $\bar{x}_k$ (the variable-wise average of the observations in cluster k).
   (b) Given the k centroids, reassign all observations to clusters based on their closeness to the centroids.



The bad and the good:

- Requires analyst to select K in advance.
- The algorithm is locally optimal, not globally optimal. As a consequence this means you can get different clusterings depending on the starting cluster assignment.

Potential solutions:

- Can try various values of K and compare results. We will dig deeper into this approach shortly.

- Can try different initial cluster assignments in parallel, and then choose the solution with the best within-cluster sum of squared deviations.

Application to violent crimes data:

Choose 4 clusters – we will see later why 4 is reasonable.

```
from sklearn.cluster import KMeans
#random_state parameter sets seed for random number generation
arrests_km =
  KMeans(n_clusters=4,
         n_init=25,random_state=123).fit(USArrests_scaled[[0,1,2,3]])
```

```
pd.DataFrame(arrests_km.cluster_centers_,
    columns=['Murder','Assault','UrbanPop','Rape'],index = range(1,5))
```

```
       Murder     Assault    UrbanPop        Rape
1   -0.494407   -0.386484    0.581676   -0.264310
2    0.702127    1.049994    0.729974    1.289904
3   -0.971303   -1.117836   -0.939550   -0.976578
4    1.426224    0.883211   -0.822791    0.019467
```

```
pd.DataFrame(arrests_km.labels_+1,columns=['Cluster'],
    index=fullstatenames)
```

```
             Cluster
    Alabama        4
     Alaska        2
    Arizona        2
   Arkansas        4
 California        2
   Colorado        2
Connecticut        1
   Delaware        1
    Florida        2
        ...
```

Can compute the (unscaled) mean of each variable within cluster:

```
USArrests['Cluster'] = arrests_km.labels_+1
USArrests.groupby('Cluster').mean()
```

```
Cluster   Murder          Assault        UrbanPop           Rape
1       5.656250       138.875000       73.875000       18.781250
2      10.815385       257.384615       76.000000       33.192308
3       3.600000        78.538462       52.076923       12.176923
4      13.937500       243.625000       53.750000       21.412500
```

Silhouette plot: A diagnostic for K-means (and other) clustering

Once a clustering has been determined, let

- a_i = average dissimilarity between observation i and the other points in the cluster to which i belongs
- b_i = average dissimilarity between observation i and the other points in the next closest cluster to observation i

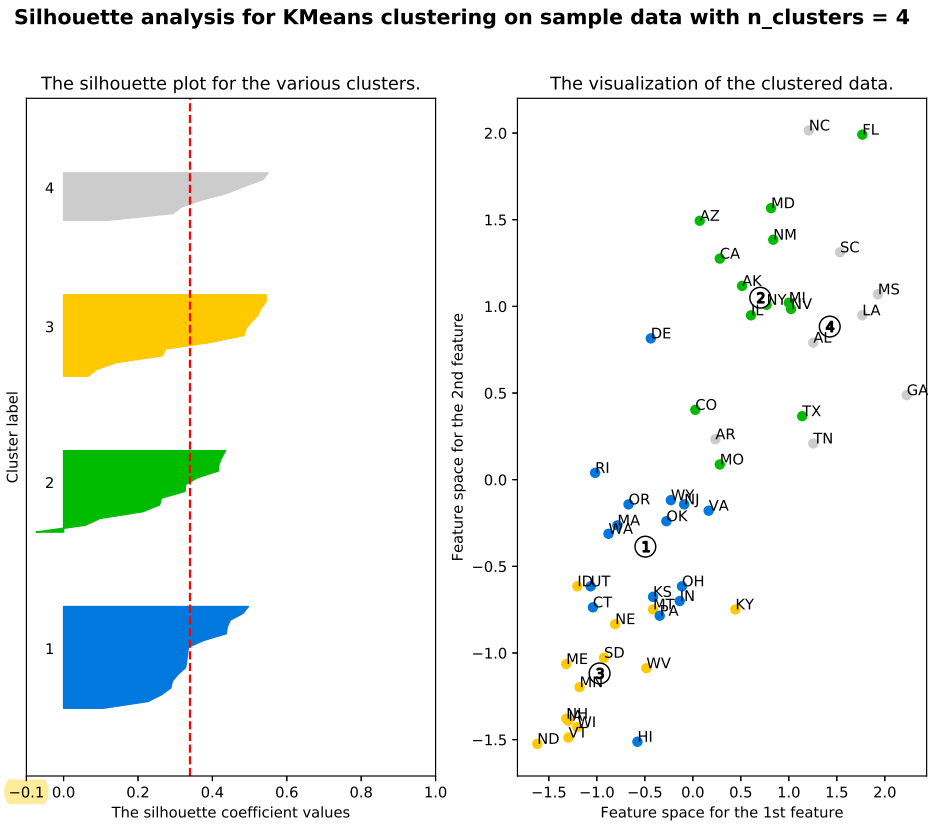
Let

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)}$$

be the silhouette for observation i.

Interpretation:

- Observations with $s_i \approx 1$ are well-clustered
- Observations with $s_i \approx 0$ lie between two clusters
- Observations with $s_i < 0$ are probably in the wrong cluster.



An observation may be in the wrong cluster!

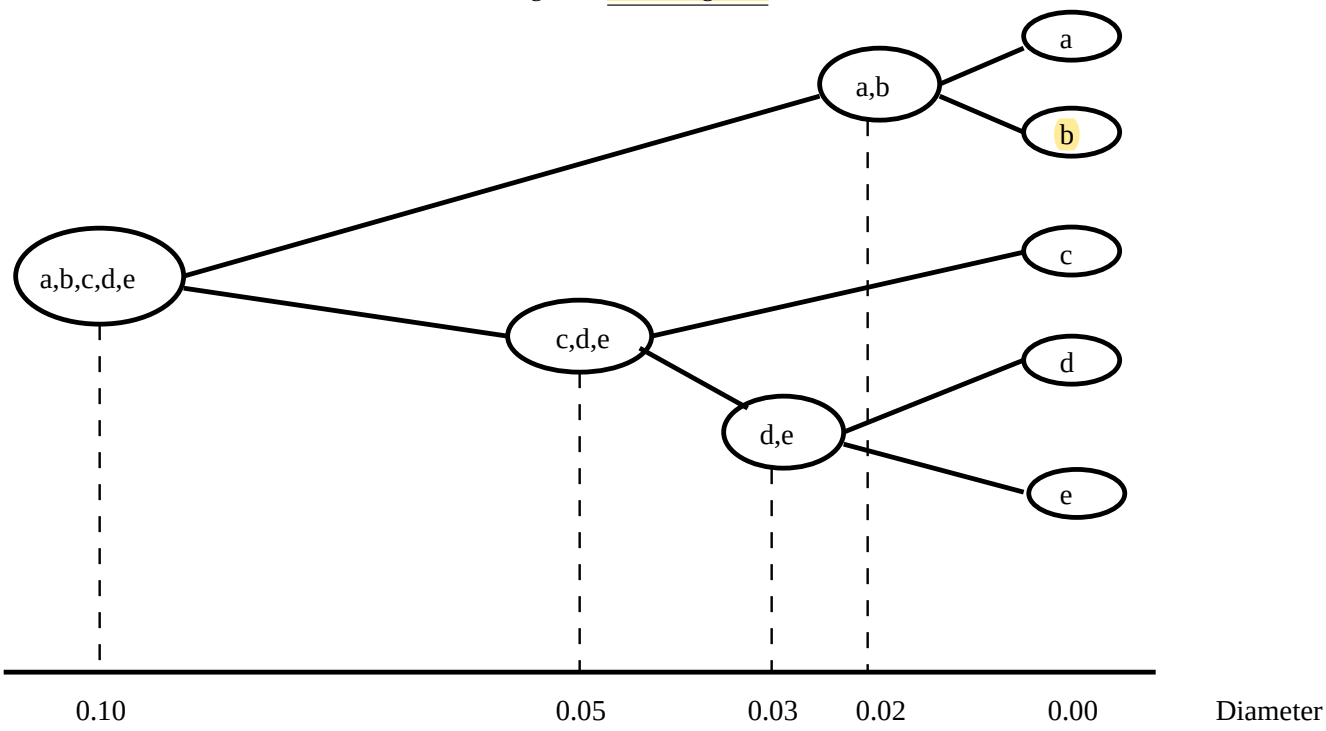
```
# Objects with negative silhouette
[fullstatenames[i] for i in range(len(sil)) if sil[i] < 0]
```

```
['Missouri']
```

Hierarchical clustering:

- K-means clustering requires pre-specifying the number of clusters K.
  (though we will come back to address this point)
- Hierarchical clustering does not require committing to a particular number of clusters, K.
- Two types of hierarchical clustering:
  - Agglomerative clustering (bottom-up approach).
  - Divisive clustering (top-down approach).

Can summarize hierarchical clustering in a dendrogram



Agglomerative clustering:

The basic algorithm:

- Each observation starts as its own cluster.
- At each step of the algorithm, two clusters that are most "similar" (to be described shortly) are combined into a new larger cluster.
- This process of combining clusters is repeated until all observations are members of one single large cluster.

Particularly well-suited at identifying small clusters.

Agglomerative clustering: How do we measure the dissimilarity between two clusters?

A few common approaches:

Complete (or maximum) linkage clustering: For two clusters, determine the maximum dissimilarity between any observation in the first cluster and any observation in the second cluster.

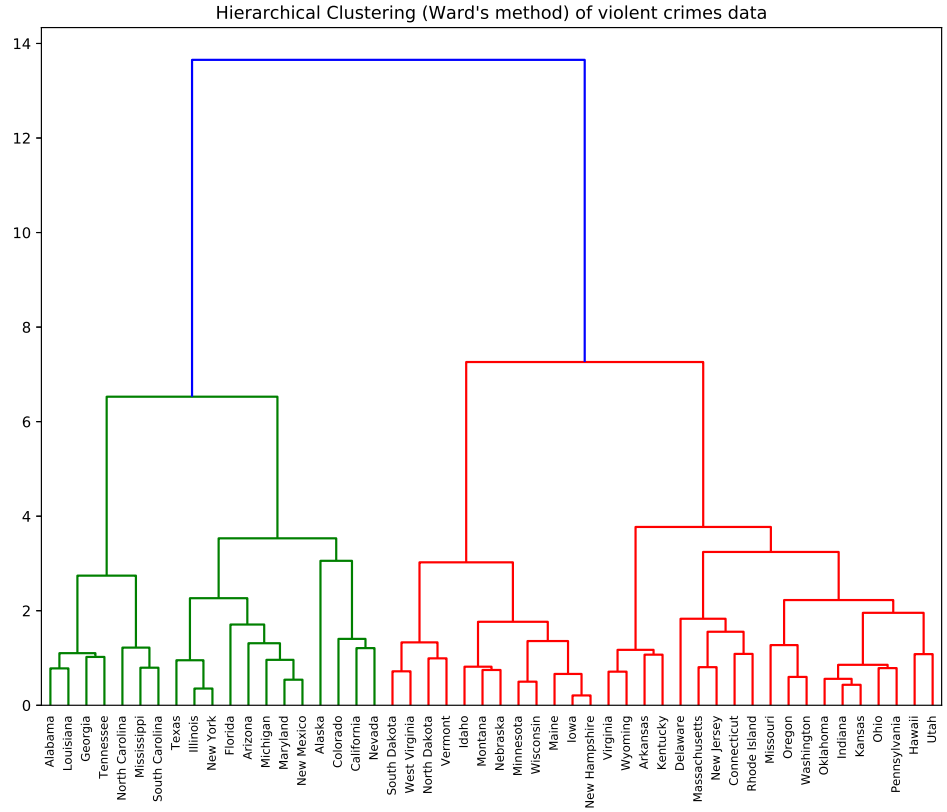
Single linkage clustering: For two cluster, determine the minimum dissimilarity between any observation in the first cluster and any observation in the second cluster.

Average linkage clustering: Compute all pairwise dissimilarities between observations in the first and second cluster, and calculate the average.

Another popular approach is Ward's method.

- Instead of joining two clusters based on their distances, use information about the variance of observations within clusters.
- Specifically, at each step, join two clusters whose merged cluster has the smallest within-cluster sum of squared distances.

Other aggregation approaches are possible.



Choosing the optimal number of clusters:

No single compelling way to choose clusters, but we will explore two types of methods.

- Direct methods that involve optimizing a particular criterion (elbow and silhouette methods)
- Testing methods that evaluate evidence against a null hypothesis (gap statistic)

Elbow method: A little squishy, but useful

As previously, let $W(C_k)$ be the within-cluster sum of squared distances between all pairs for cluster k for a particular clustering, and let

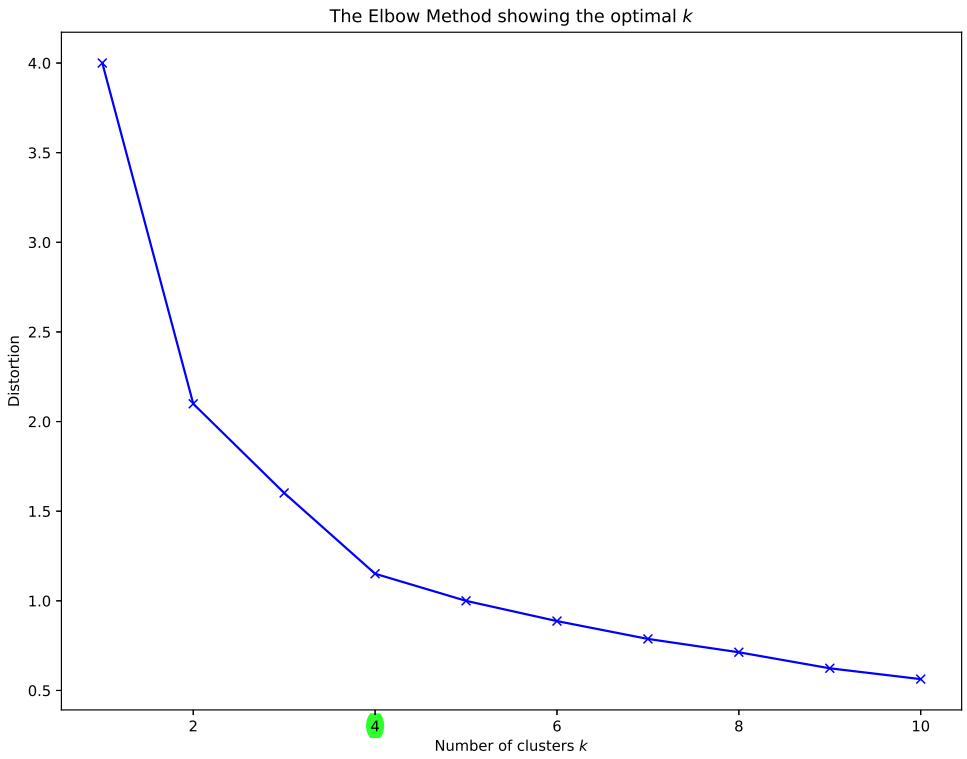
$$T_K = \sum_{k=1}^{K} W(C_k)$$

be the total within-cluster variation for the clustering with K clusters.

The method:

1. For the particular clustering method, let K vary over a range of values (say 1 to 10).
2. Compute T_K for each K.
3. Plot T_K against K, and look for a clear bend ("knee") in the graph.

The value of K where the bend occurs is considered the appropriate number of clusters.



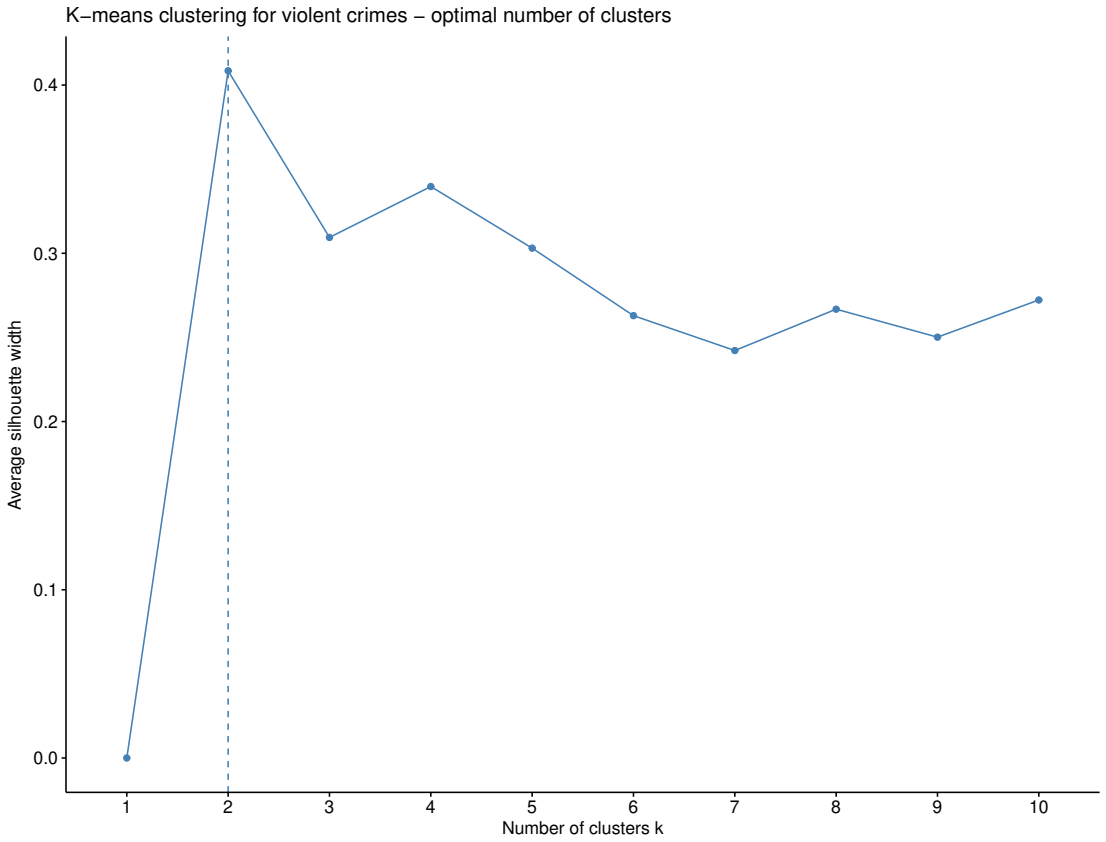
Average silhouette method: Similar construction to the elbow method

1. Compute clustering algorithm for different numbers of clusters K, varying K from (say) 1 to 10.
2. For each K, calculate the average silhouette

$$S_K = \frac{1}{n}\sum_{i=1}^{n} s_i$$

   across all observations.
3. Plot S_K against K.

The value of K where S_K is maximized is considered the appropriate number of clusters.



Comments:

- Elbow method suggests four clusters for K-means; average silhouette method suggests two clusters
- Computation for each method are different enough that inconsistent results are not uncommon
- Both approaches measure global clustering characteristics only, and are informal (and somewhat ad hoc) approaches

Gap statistic:

Due to Tibshirani et al., (JRSS-B, 2001).

Idea: For a particular choice of K clusters, compare the total within cluster variation to the expected within-cluster variation under the assumption that the data have no obvious clustering (i.e., randomly distributed).

The gap statistic in essence detects whether the data clustered into K groups is significantly better than if they were generated at random.

Algorithm for computing Gap statistic:

1. Cluster the data at varying number of total clusters K, say from 1 to 10. Let T_K be the total within-cluster sum of squared distances.

2. Generate B reference data sets of size n, with the simulated values of variable j uniformly generated over the range of the observed variable $\boldsymbol{x}_j$. Typically $B = 500$.

3. For each generated data set $b = 1, \ldots, B$, perform the clustering for each K (from 1 to 10, say). Compute the total within-cluster sum of squared distances $T_k^{(b)}$.

4. Compute the Gap statistic

$$\text{Gap}(K) = \left(\frac{1}{B} \sum_{b=1}^{B} \log(T_K^{(b)}) \right) - \log(T_K).$$

5. Let $\bar{w} = \frac{1}{B} \sum_{b=1}^{B} \log(T_K^{(b)})$. Compute the standard deviation

$$\text{sd}(K) = \sqrt{\frac{1}{B} \sum_{b=1}^{B} \left(\log(T_K^{(b)}) - \bar{w} \right)^2}.$$

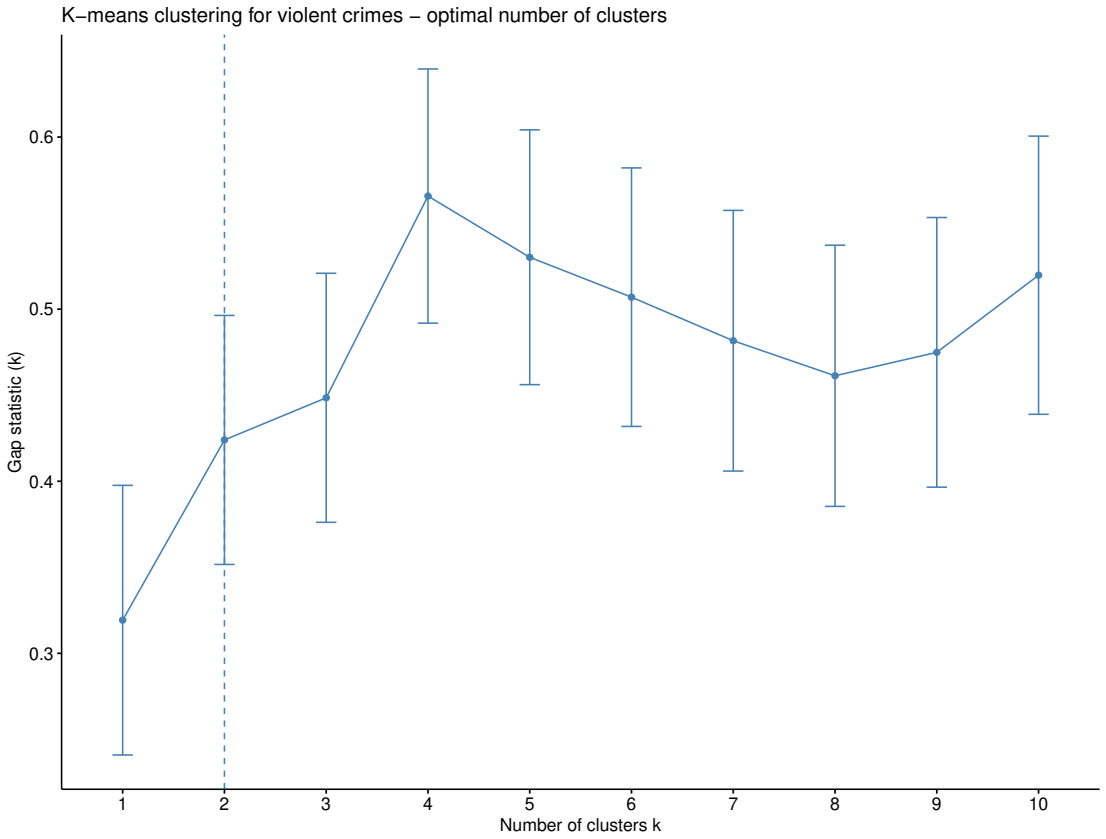
Define $s_K = \text{sd}(K)\sqrt{1 + 1/B}$.

6. Finally, choose the number of clusters as the smallest K such that

$$\text{Gap}(K) \geq \text{Gap}(K+1) - s_{K+1}.$$

Python results:

```
          logW   E.logW       gap     SE.sim
 [1,] 4.584967 4.904244 0.3192769 0.07833694
 [2,] 3.940245 4.364248 0.4240033 0.07234850
 [3,] 3.667698 4.116182 0.4484847 0.07235788
 [4,] 3.339378 3.905104 0.5657253 0.07385095
 [5,] 3.197534 3.727690 0.5301563 0.07402528
 [6,] 3.064162 3.571133 0.5069703 0.07510947
 [7,] 2.952411 3.434063 0.4816516 0.07574023
 [8,] 2.848435 3.309711 0.4612760 0.07588600
 [9,] 2.721434 3.196339 0.4749049 0.07834490
[10,] 2.571982 3.091706 0.5197243 0.08082849
```

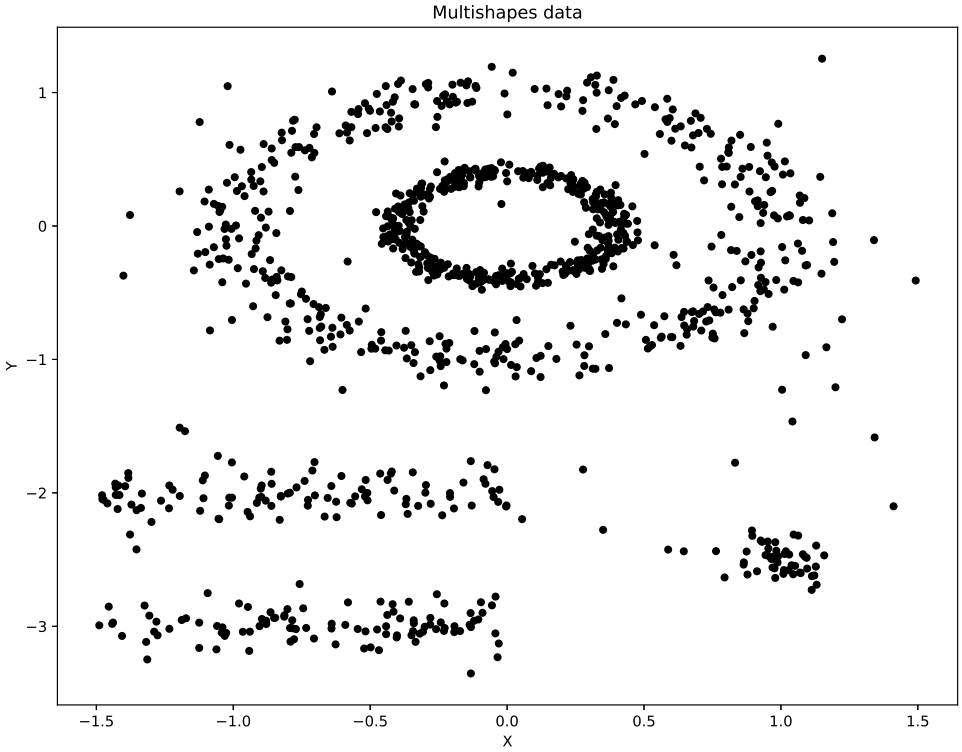


Comments:

- K-means clustering is optimized for $K = 2$ based on the Gap statistic.
- This is a more principled approach to choosing cluster sizes, though clearly different conclusions can be reached based on different clustering approaches.

One final clustering algorithm:

Imagine trying to cluster the following set of observations.



What happens if we try K-means?

DBSCAN: Density-based clustering algorithm (Ester et al., 1996, KDD-96)

Unlike previous clustering algorithms, DBSCAN

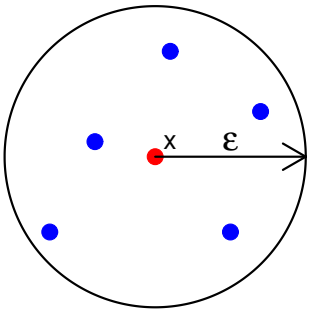
- can find any shape of clusters
- identifies observations that do not belong to clusters as outliers
- does not require specifying the number of clusters (like hierarchical clustering)
- can be used for predicting cluster membership for new data

DBSCAN algorithm: Identify dense regions of observations

Need to specify two parameters of the algorithm

1. ϵ: the radius of a neighborhood around an observation
2. `MinPts`: the minimum number of points within an ϵ radius of an observation to be considered a "core" point

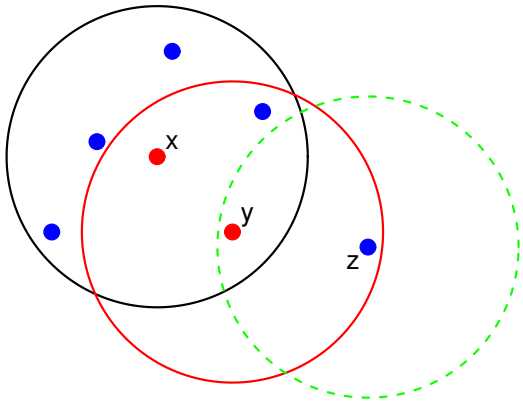
Example of core observation with `MinPts`$= 6$.



Three types of points:

- Core points - observations with `MinPts` total observations within an ϵ radius
- Border points - observations that are not core points, but are within ϵ of a core point
- Noise points - everything else

In the following, `x` is a core point, `y` is a border point, and `z` is a noise point.



Two terms:

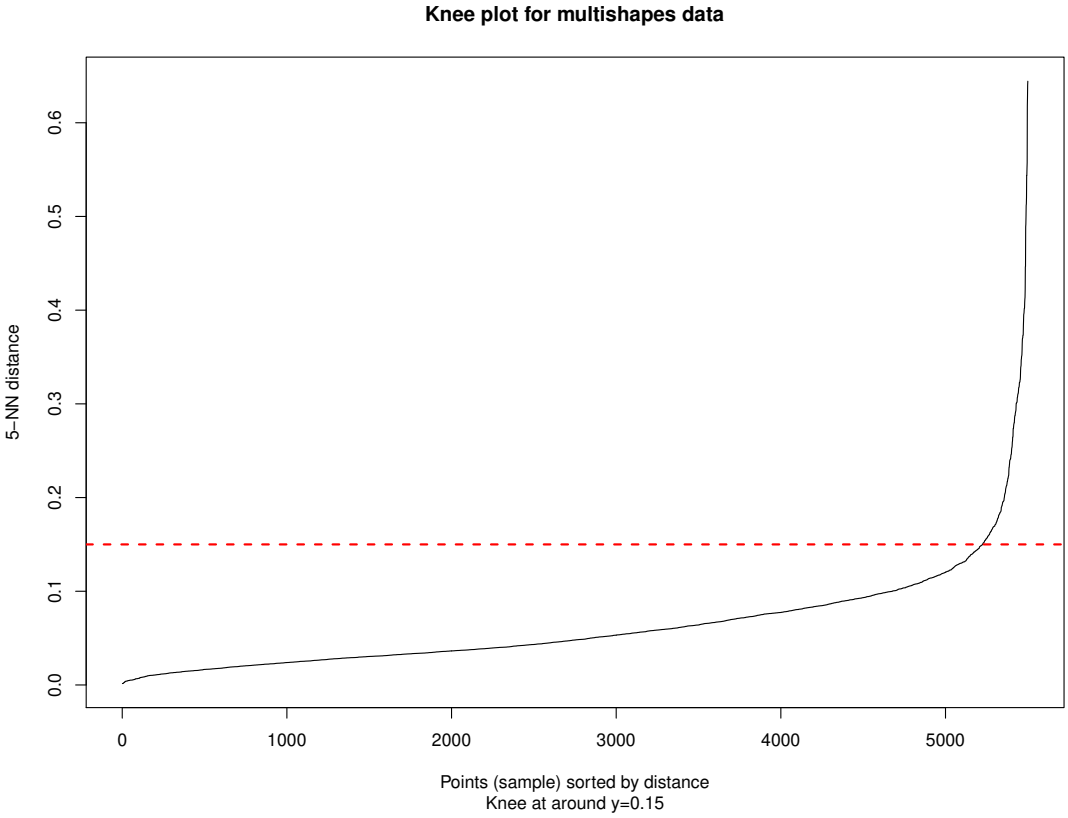
- Density-reachable: Point A is density-reachable from point B if there is a set of core points leading from B to A.
- Density-connected: Two points A and B are density-connected if there is a core point C such that both A and B are density-reachable from C.

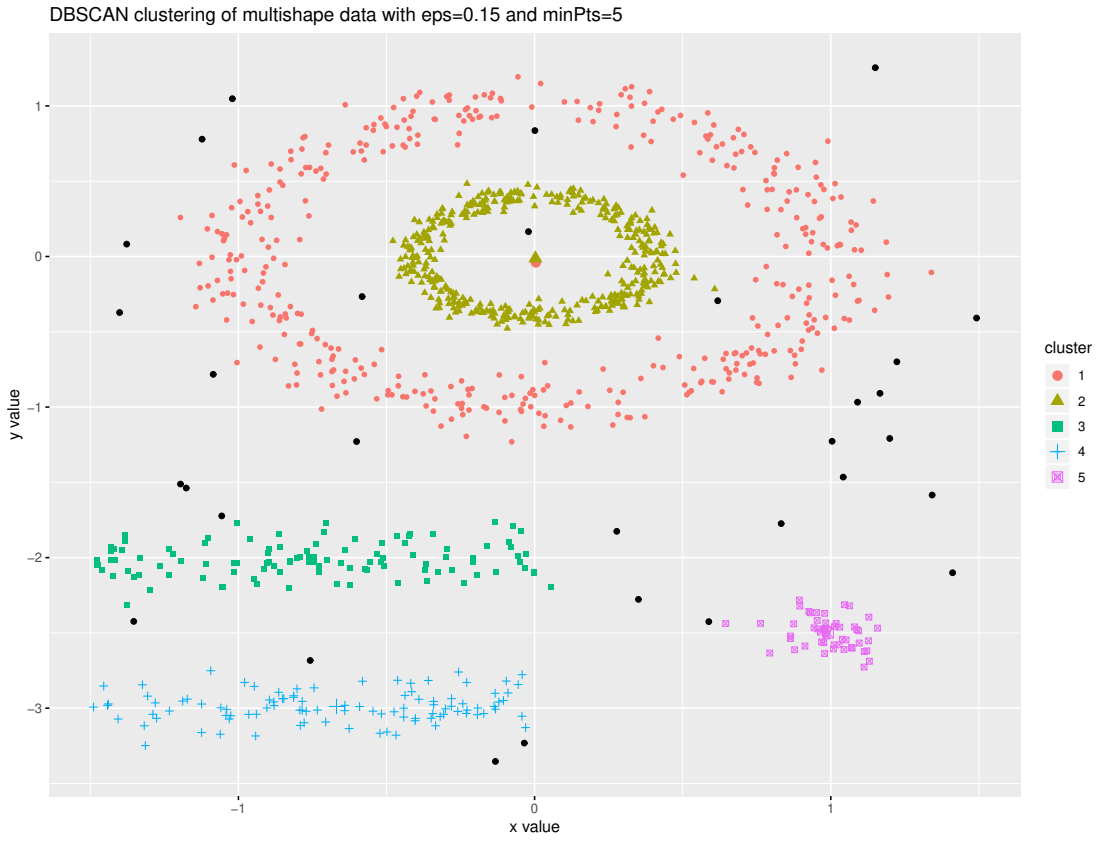
A density-based cluster is defined as a group of density-connected points.

Choosing ϵ:

- Compute the k-th nearest neighbor distance for each point.
- Plot the distances in sorted order.
- Look for a bend (the "knee") in the plot, and use the distance at the knee as the choice of ϵ.

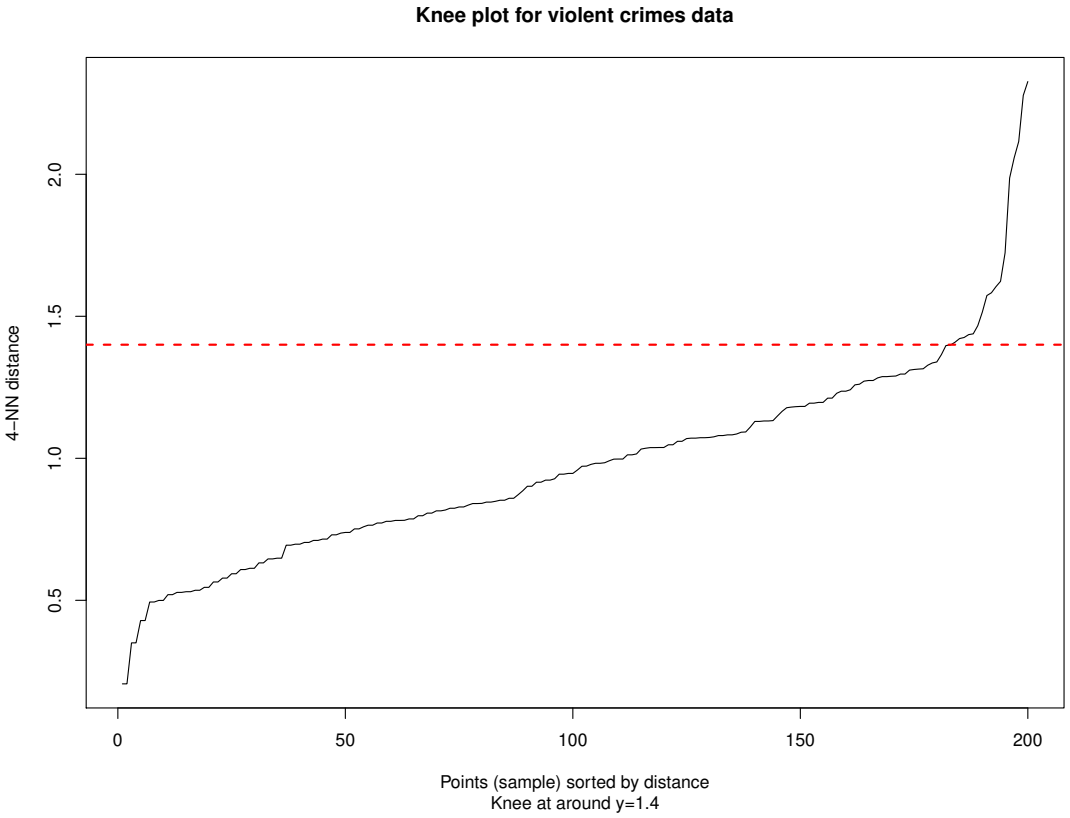
Refer to Ester et al. for the algorithm details, as well as the optimizing the choice of ϵ.

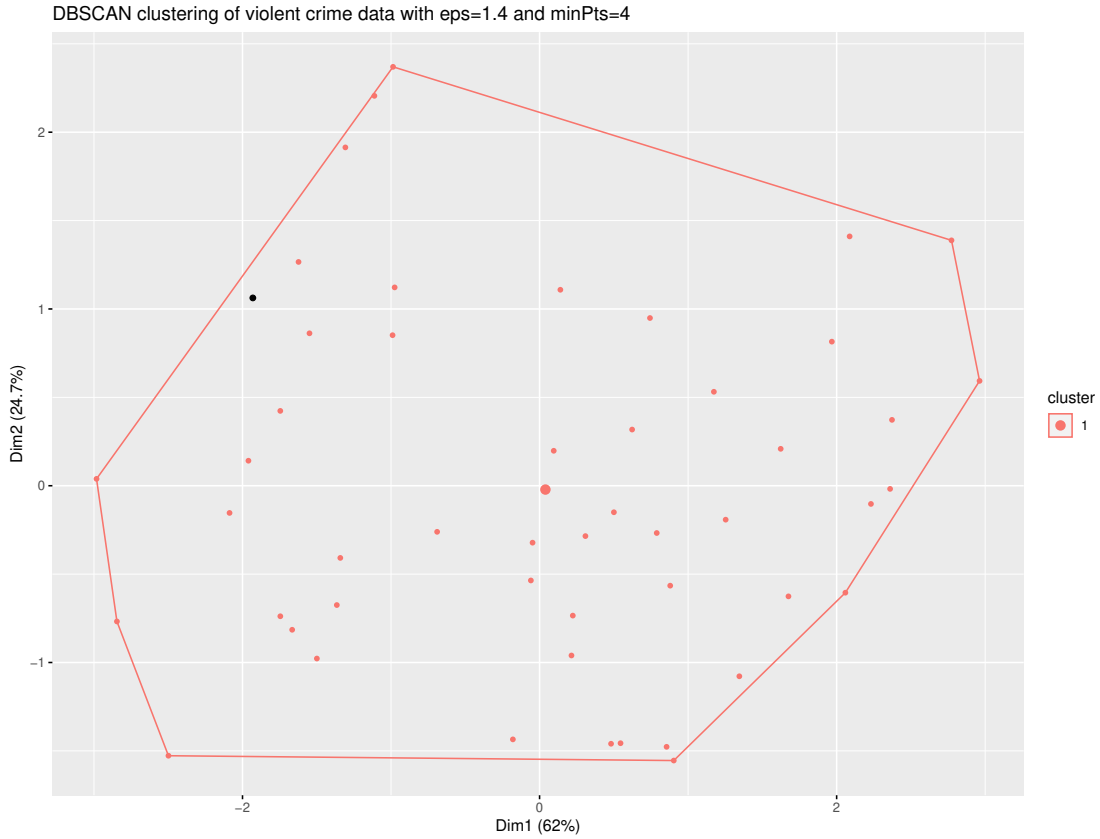

Knee plot for multishapes data
5−NN distance
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0
1000
2000
3000
4000
5000
Points (sample) sorted by distance
Knee at around y=0.15


DBSCAN clustering of multishape data with eps=0.15 and minPts=5
y value
1
0
−1
−2
−3
−1
0
1
x value
cluster
1
2
3
4
5

Application to violent crime data:

- Chose $\epsilon = 1.4$ on standardized data based on knee plot, and `MinPts`$= 4$.
- Resulted in 1 density-based cluster, and 1 outlier.





Now for something really different: Chernoff faces

An unusual graphical method to cluster data:

- Each observation is a face
- Each variable is a facial feature
- Invented mostly as a joke!

Car data: Data on model year 1993 cars

Variables include:

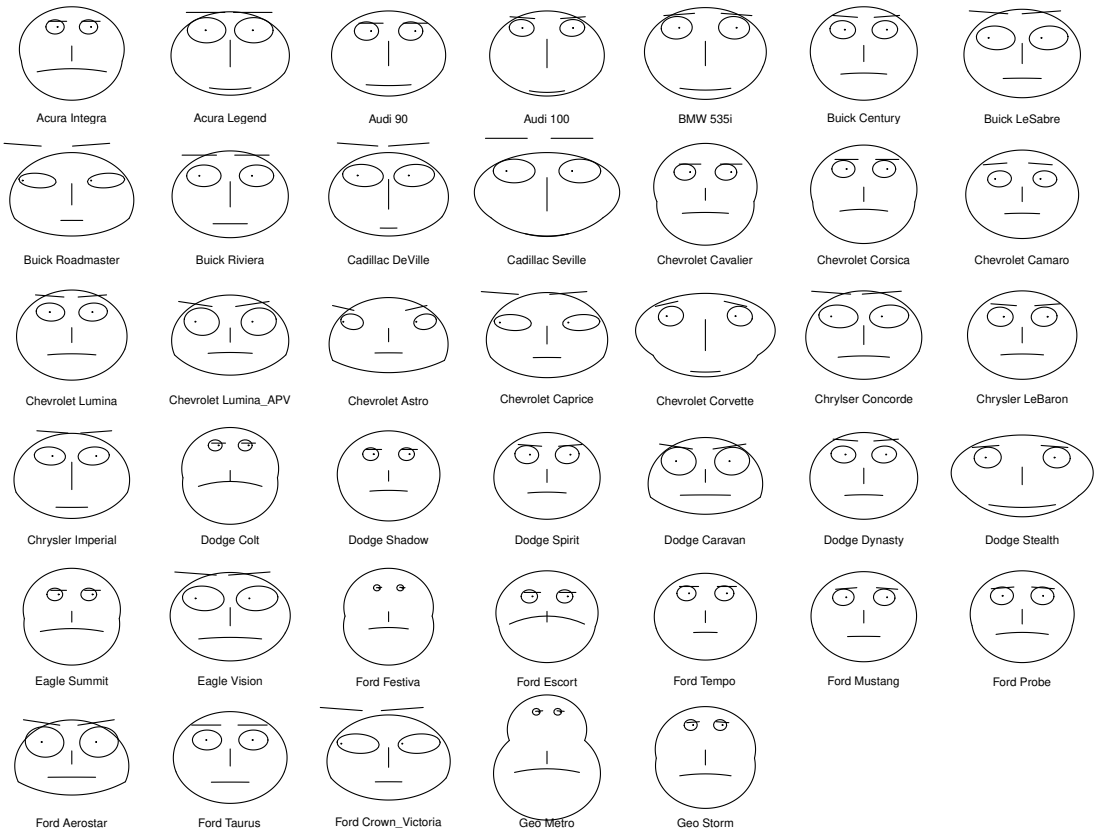
Weight, MPG (city), MPG (highway), horsepower, engine size, minimum price, price, maximum price, RPM, revs per mile, fuel tank capacity, passenger capacity, length, wheel base.

Example Chernoff faces: Car data

- Width of face – vehicle weight
- Height of face split – City MPG
- Length of face – Highway MPG

- Width of top half of face – Horsepower
- Width of bottom half of face – Engine size

Other facial features: Length of nose, curvature of mouth, size of eyes, angle of eyebrows, etc.




Acura Integra
Acura Legend
Audi 90
Audi 100
BMW 535i
Buick Century
Buick LeSabre
Buick Roadmaster
Buick Riviera
Cadillac DeVille
Cadillac Seville
Chevrolet Cavalier
Chevrolet Corsica
Chevrolet Camaro
Chevrolet Lumina
Chevrolet Lumina_APV
Chevrolet Astro
Chevrolet Caprice
Chevrolet Corvette
Chrylser Concorde
Chrysler LeBaron
Chrysler Imperial
Dodge Colt
Dodge Shadow
Dodge Spirit
Dodge Caravan
Dodge Dynasty
Dodge Stealth
Eagle Summit
Eagle Vision
Ford Festiva
Ford Escort
Ford Tempo
Ford Mustang
Ford Probe
Ford Aerostar
Ford Taurus
Ford Crown_Victoria
Geo Metro
Geo Storm


Acura Integra
Acura Legend
Audi 90
Audi 100
BMW 535i
Buick Century
Buick LeSabre
Buick Roadmaster
Buick Riviera
Cadillac DeVille
Cadillac Seville
Chevrolet Cavalier
Chevrolet Corsica
Chevrolet Camaro
Chevrolet Lumina
Chevrolet Lumina_APV
Chevrolet Astro
Chevrolet Caprice
Chevrolet Corvette
Chrylser Concorde
Chrysler LeBaron
Chrysler Imperial
Dodge Colt
Dodge Shadow
Dodge Spirit
Dodge Caravan
Dodge Dynasty
Dodge Stealth
Eagle Summit
Eagle Vision
Ford Festiva
Ford Escort
Ford Tempo
Ford Mustang
Ford Probe
Ford Aerostar
Ford Taurus
Ford Crown_Victoria
Geo Metro
Geo Storm