



Education is the most powerful weapon which you can use to change the world. – Nelson Mandela

# Clustering

---

(SEGMENTATION)

# Clustering

---

- Unsupervised: no target variable for training
- Partition the data into groups (clusters) so that:
  - Observations within a cluster are similar in some sense
  - Observations in different clusters are different in some sense
- There is no one correct answer, though there are good and bad cluster solutions
- No method works best all the time

# Examples of Clustering (segmentation)

---

- Customer segmentation: groups of customers with similar shopping or buying patterns
- Dimension reduction:
  - Cluster individuals together and use cluster variable as proxy for demographic or behavioral variables
- Gather stores with similar characteristics for sales forecasting
- Find topics (clusters of words) in text data
- Find communities in social networks

# Hard versus Fuzzy

---

Hard: objects can belong to only one cluster

- k-means
- DBSCAN
- Hierarchical

Fuzzy: objects can belong to more than one cluster (usually with some probability)

- Fuzzy C-means (FCM)
- Gaussian Mixture Models / Expectation-Maximization (EM)

# Hierarchical versus Flat

---

Hierarchical: clusters form a tree so you can visually see which clusters are most similar to each other.

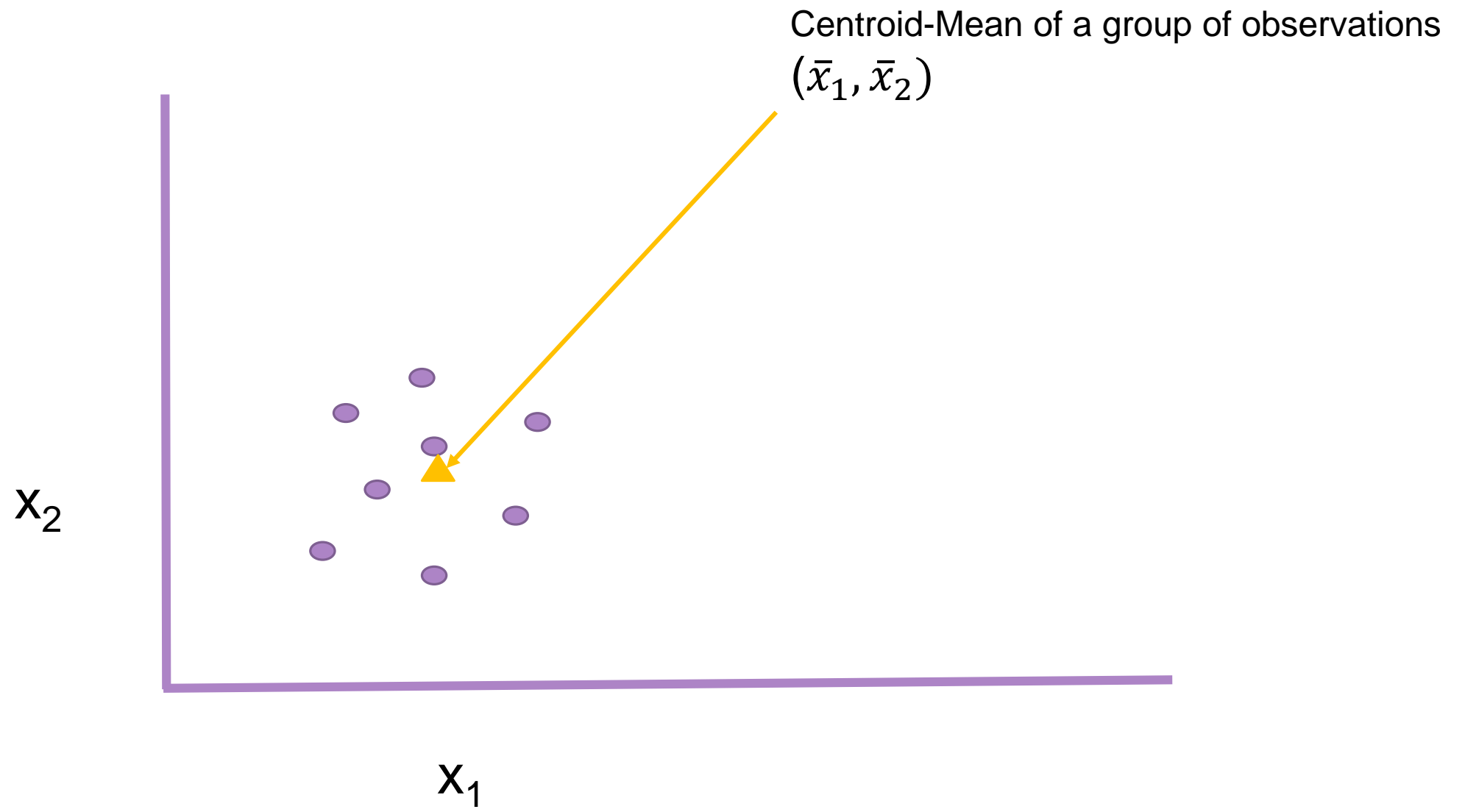
- *Agglomerative*: points start out as individual clusters, and they are combined until everything is in one cluster. (ex: Linkages - coming next class)
- *Divisive*: All points start in same cluster and at each step a cluster is divided into two clusters. (ex: DIANA)

Flat: Clusters are created according to some other process, usually iteratively updating cluster assignments (usually specify the number of clusters apriori)

# K-Means

---

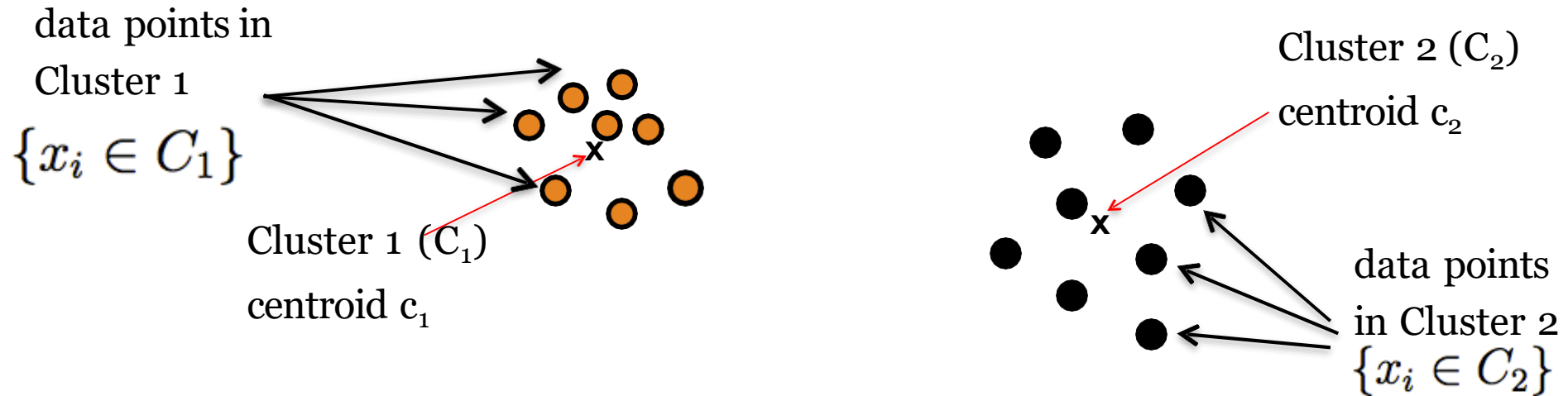
MOST POPULAR CLUSTERING METHOD





# k-Means Clustering

- The most popular clustering algorithm



- Tries to minimize the sum of squared distances from each point to its cluster centroid. (Global objective function)

$$\sum_{C_k} \sum_{x_i \in C_k} \|x_i - c_k\|^2$$

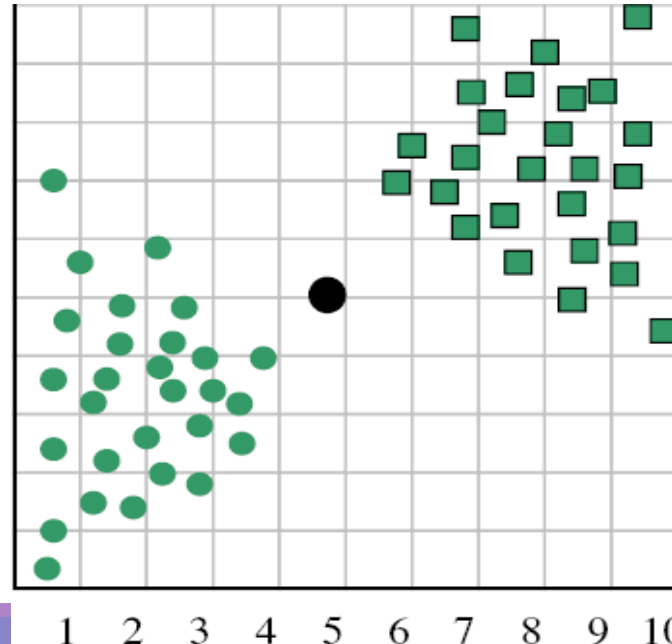
# K-means algorithm

---

- Start with k “seed points”
  - Randomly initialized (most software)
- Assign each data point to the closest seed point.
- The seed point then represents a cluster of data
- Reset seed points to be the centroids of the cluster
- Repeat steps 2-4 updating the cluster centroids until they do not change.

# Determining Number of Clusters (SSE)

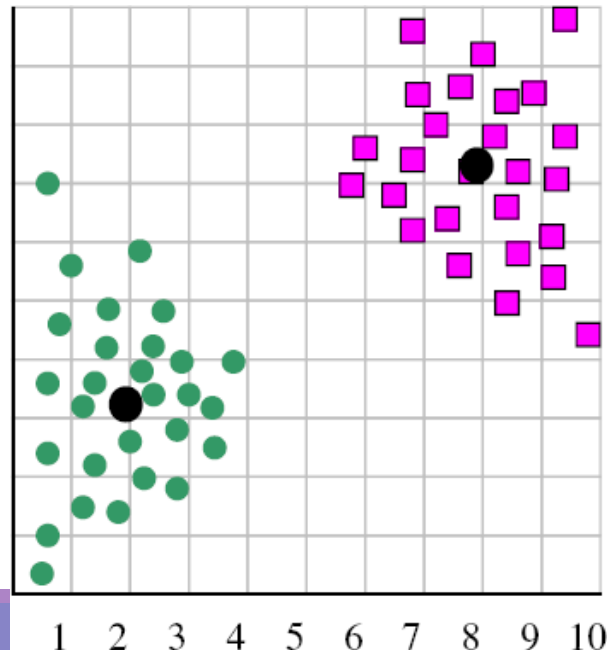
- Try the algorithm with  $k=1,2,3,\dots$
- Examine the objective function values
- Look for a place where the marginal benefit to objective function for adding a cluster becomes small



$k=1$  objective function  
(SSE) is 902

# Determining Number of Clusters (SSE)

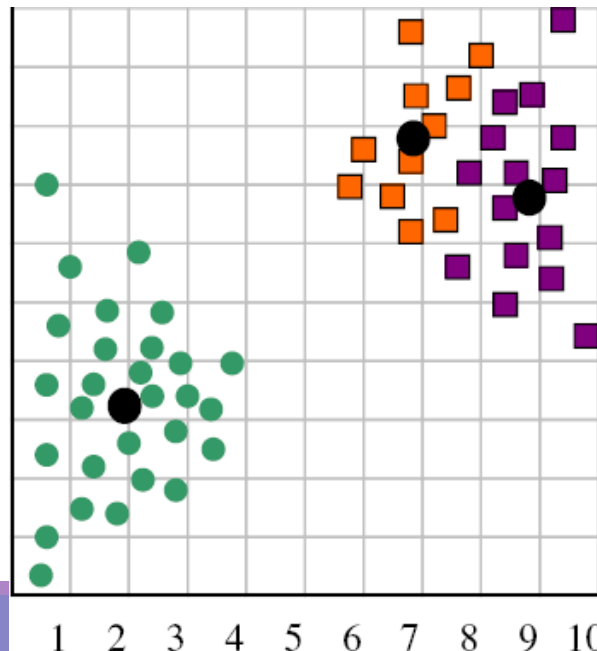
- Try the algorithm with  $k=1,2,3,\dots$
- Examine the objective function values
- Look for a place where the marginal benefit to objective function for adding a cluster becomes small



$k=2$  objective  
function (SSE) is 213

# Determining Number of Clusters (SSE)

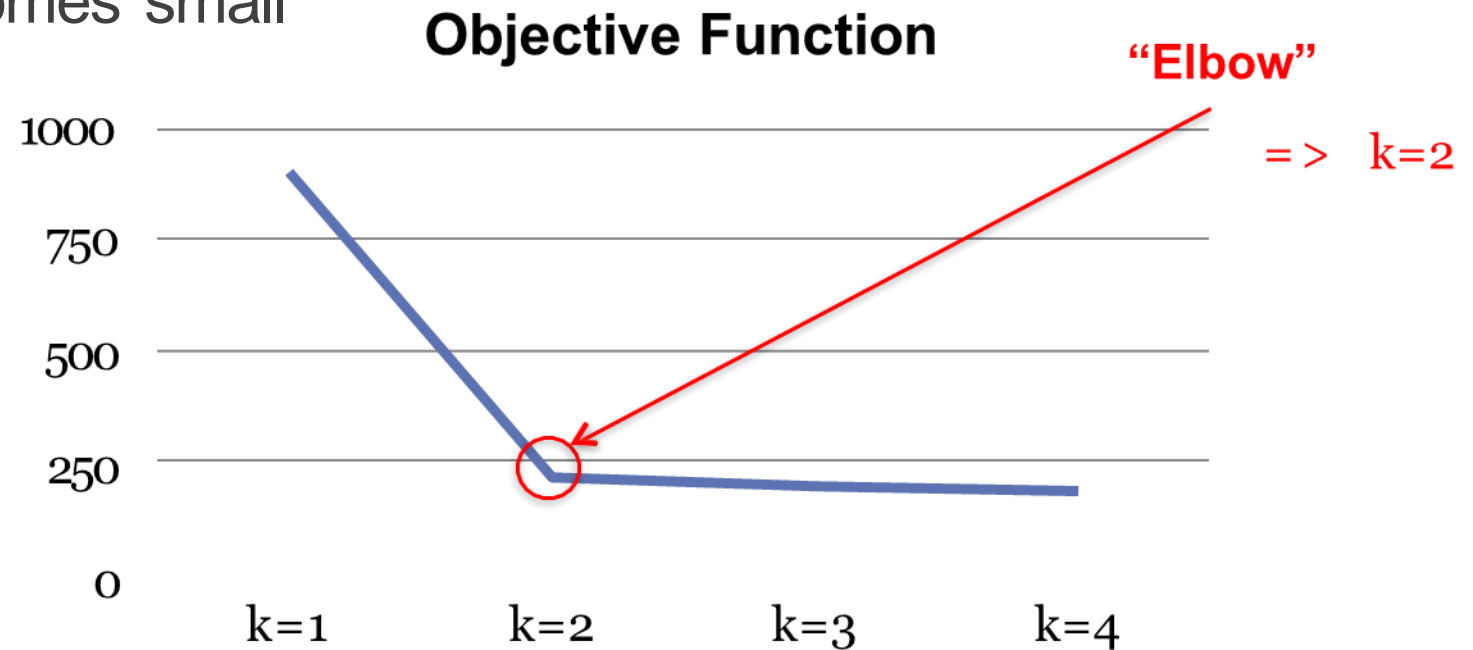
- Try the algorithm with  $k=1,2,3,\dots$
- Examine the objective function values
- Look for a place where the marginal benefit to objective function for adding a cluster becomes small



$k=3$  objective  
function (SSE) is 193

# Determining Number of Clusters (SSE)

- Try the algorithm with  $k=1,2,3,\dots$
- Examine the objective function values
- Look for a place where the marginal benefit to objective function for adding a cluster becomes small



# K-means summary

---

## Disadvantages

- Dependent on initialization (initial seeds)
- Can be sensitive to outliers
  - If problem, should consider k-medoids
- Have to input the number of clusters
- Difficulty detecting non-spheroidal (globular) clusters

## Advantages

- Modest time/storage requirements.
- Shown you can terminate method after small number of iterations with good results.
- Good for wide variety of data types

# Example

---

ARREST DATA



# USArrests data set

---

Data set is already in R

Has 50 observations (one observation per state)

Provides information on arrests per 100,000 in each state in assault, murder and rape

Also contains information on percent of state's population living in an urban area

Data is from 1973



# Preprocessing

---

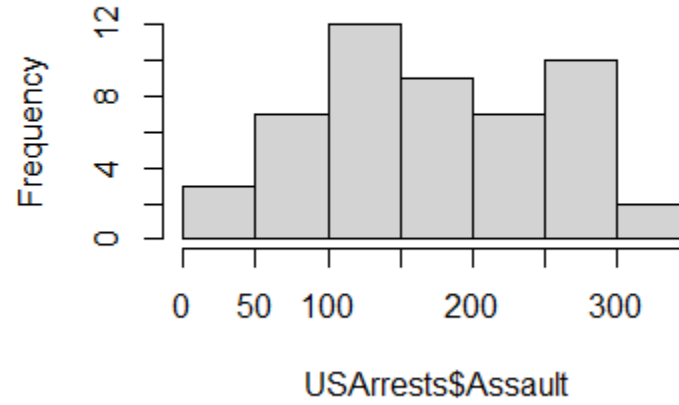
You will need to do some data exploration BEFORE trying to cluster the data

- **Missing values**
  - you will either need to impute missing values or remove that variable or observation
- **Categorical variables**
  - Do you need to bin categorical variables?
  - Need to create dummy variables for ALL categorical variables
- **Continuous variables**
  - If outliers or heavy skewness, potentially consider transformation
  - At a minimum, center the continuous variables (better to standardized, for example z-scores...this would happen AFTER any transformations are made)
- You can try clustering on original data (after scaled and dummy coded, etc) or you can try it on PCA of the variables (especially useful if data is BIG)

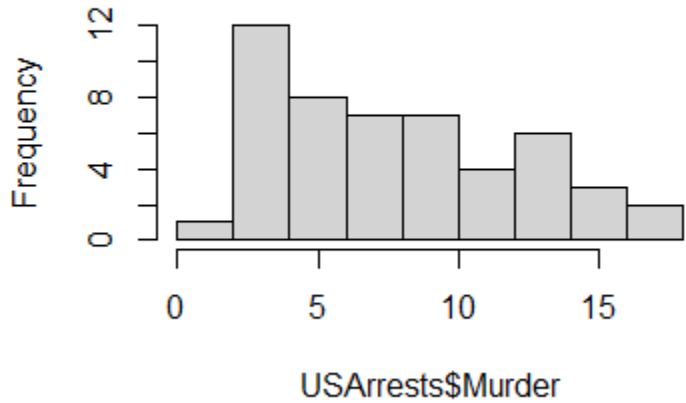
```
summary(USArrests)
hist(USArrests$Murder)
hist(USArrests$Assault)
hist(USArrests$Rape)
hist(USArrests$UrbanPop)
arrest.scal=scale(USArrests)
```

No missing values in data set!!

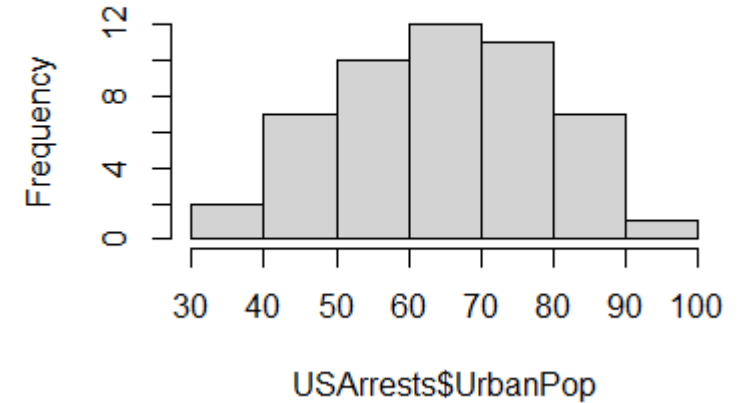
**Histogram of USArrests\$Assault**



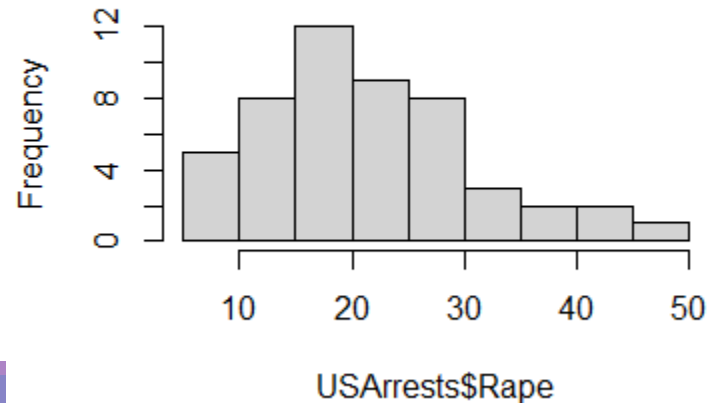
**Histogram of USArrests\$Murder**



**Histogram of USArrests\$UrbanPop**



**Histogram of USArrests\$Rape**



```
clus2=kmeans(arrest.scal,centers=2,nstart = 25)
clus2
```

K-means clustering with 2 clusters of sizes 30, 20

Cluster means:

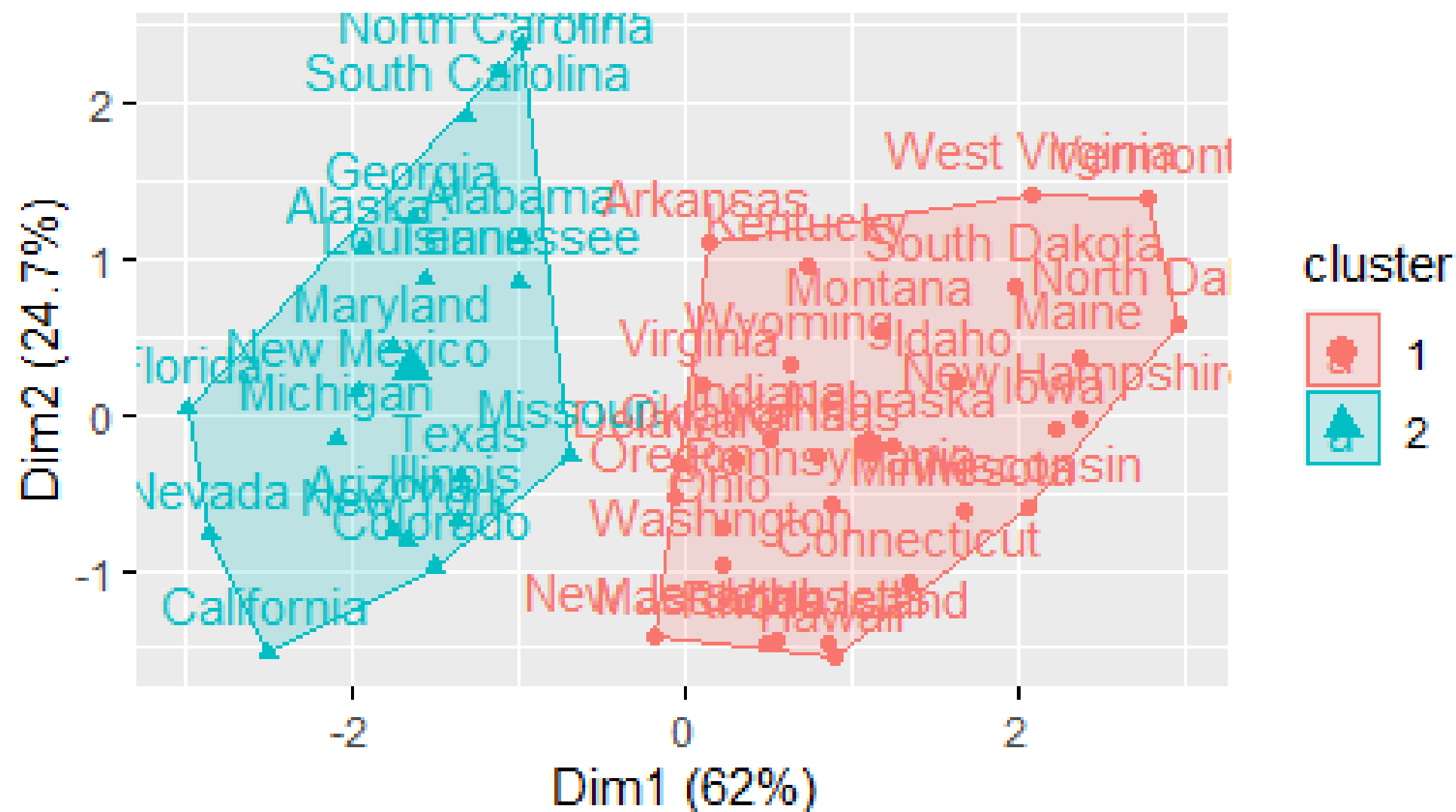
	Murder	Assault	UrbanPop	Rape
1	-0.669956	-0.6758849	-0.1317235	-0.5646433
2	1.004934	1.0138274	0.1975853	0.8469650

Clustering vector:

Alabama	Alaska	Arizona	Arkansas
2	2	2	1
California	Colorado	Connecticut	Delaware
2	2	1	1
Florida	Georgia	Hawaii	Idaho
2	2	1	1

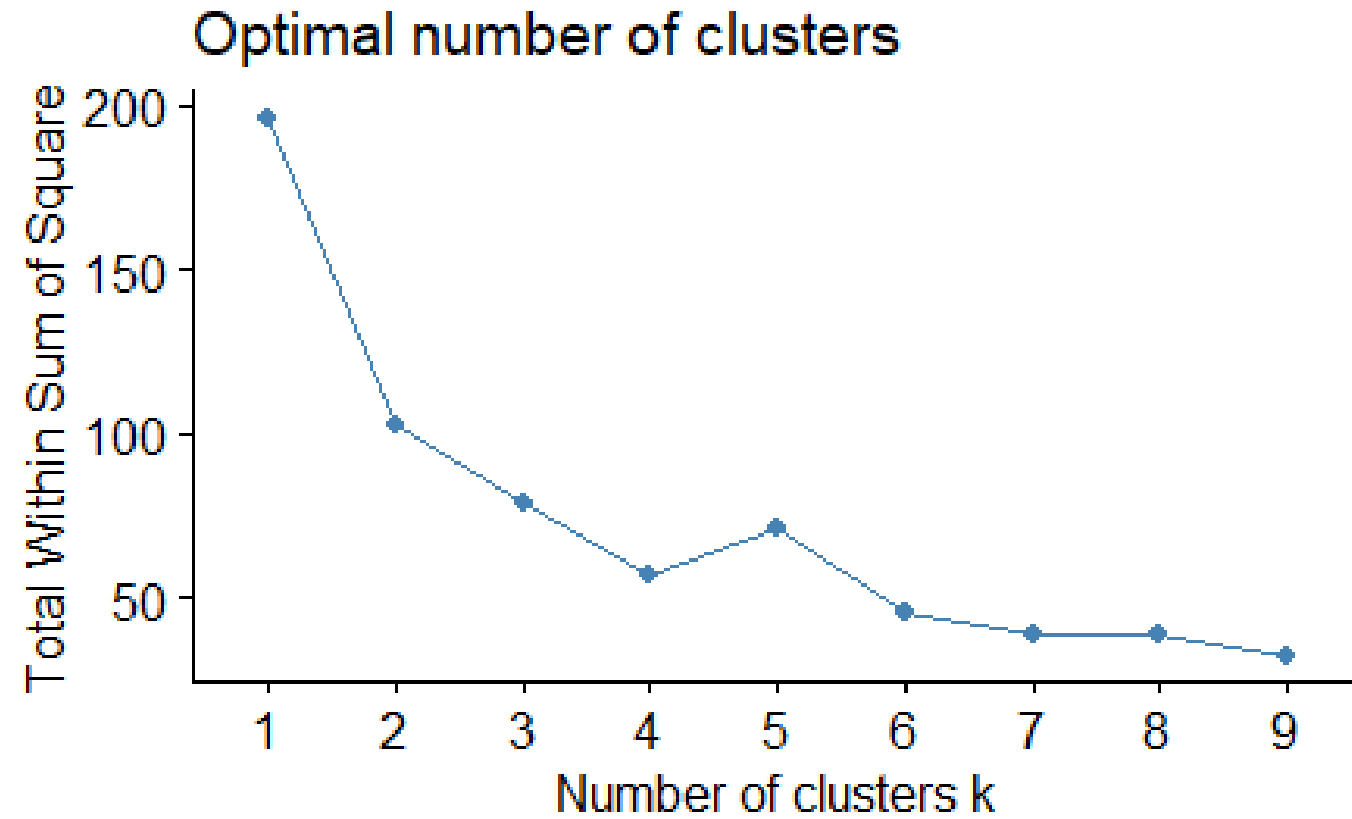
```
fviz_cluster(clus2, data = arrest.scal)
```

## Cluster plot



```
set.seed(12964)
```

```
fviz_nbclust(arrest.scal, kmeans, method = "wss", k.max = 9,)
```



# Silhouette method

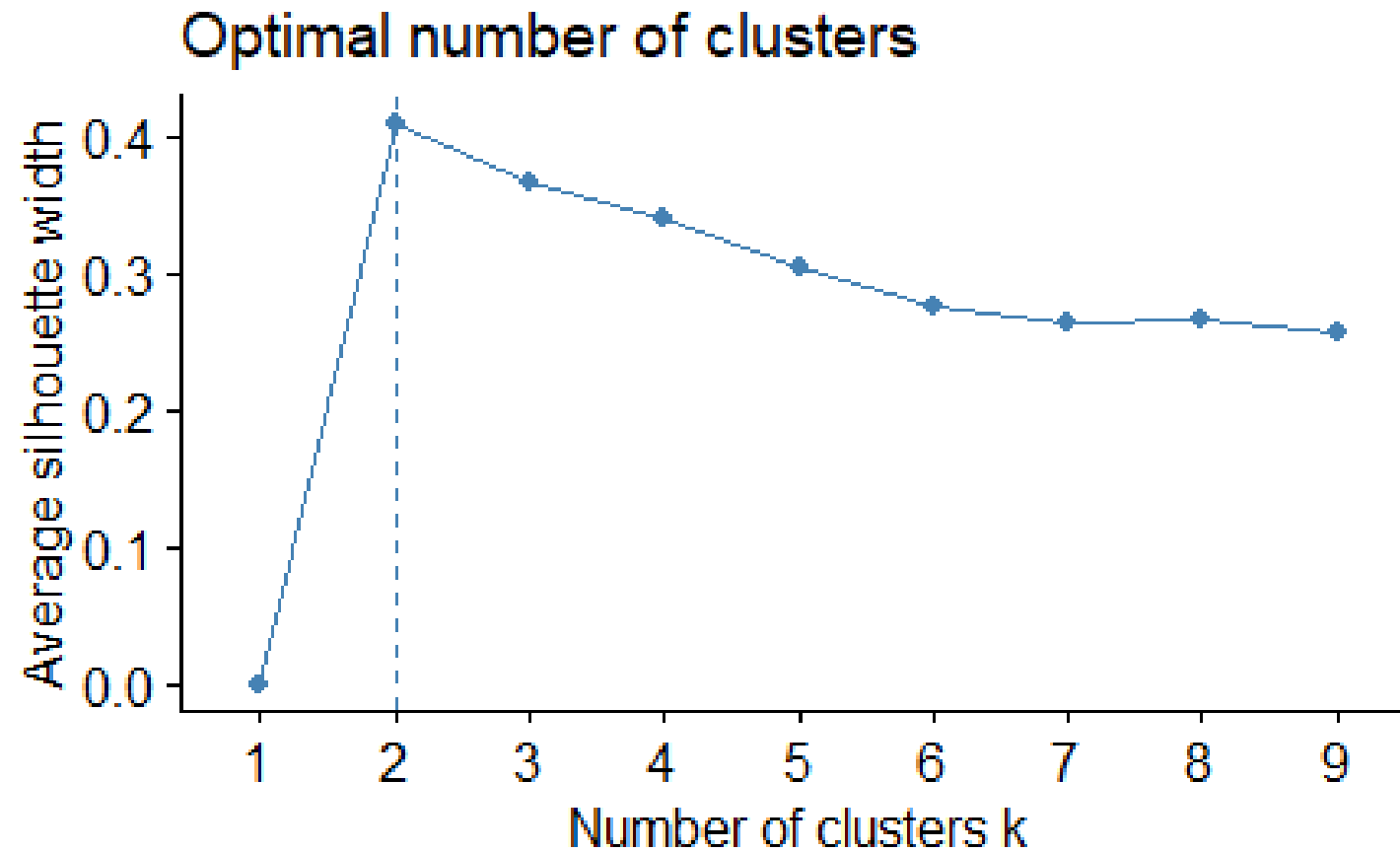
---

Another method that can be used to define the number of clusters is the Silhouette method.

This method estimates how well each observation falls within its cluster (looks at the distance the point is to all other points in that cluster and compare it to the distance from that point to points in other clusters).

Want to find the number of clusters that maximize this ratio (get close to 1)

```
fviz_nbclust(arrest.scal, kmeans, method = "silhouette", k.max = 9)
```





# Gap statistic

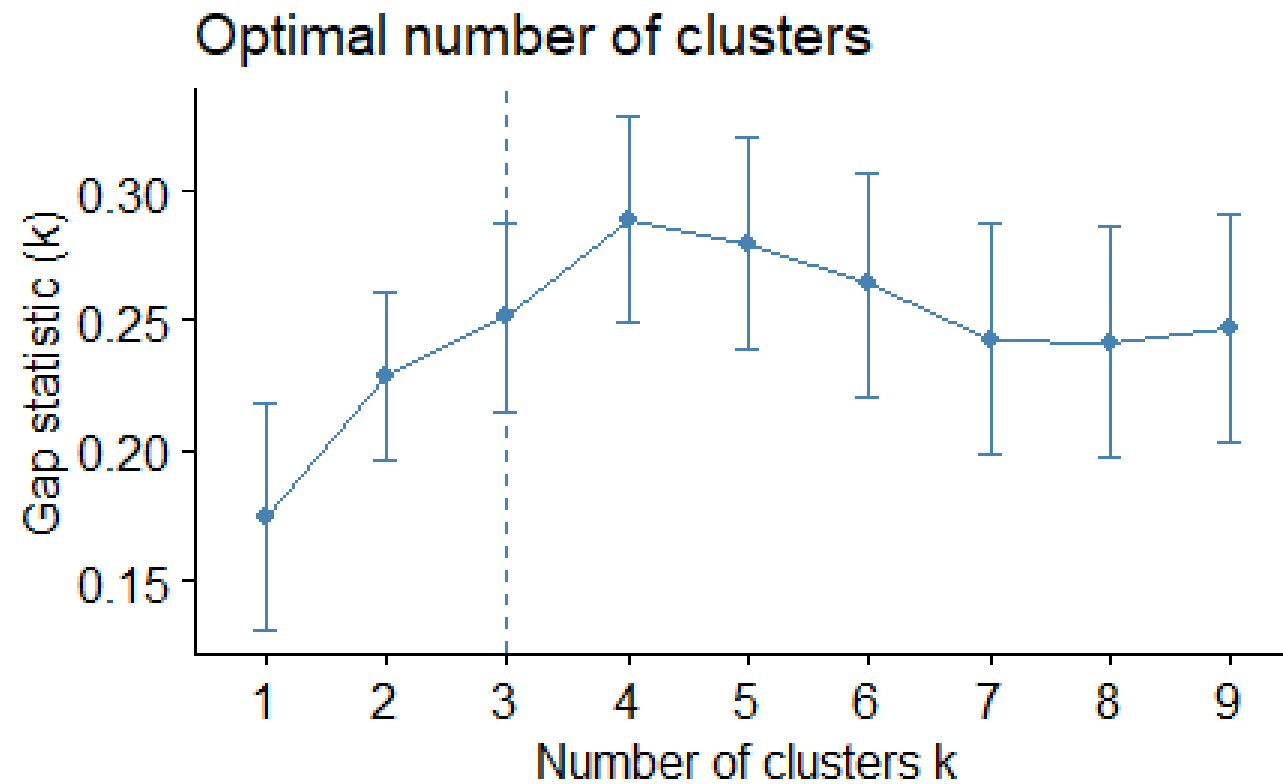
---

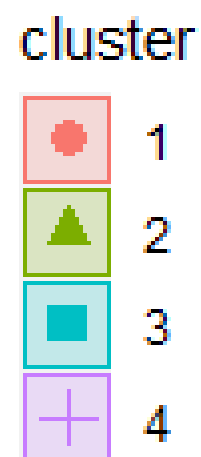
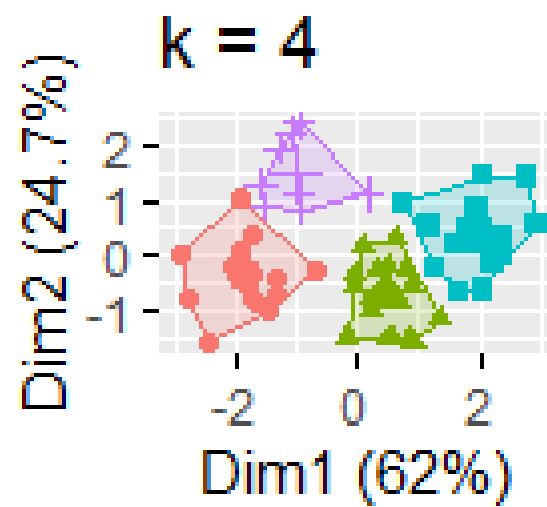
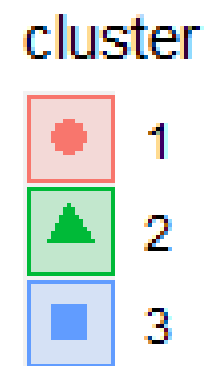
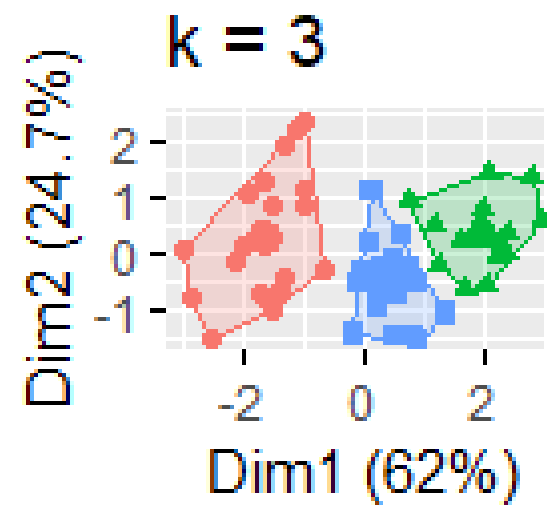
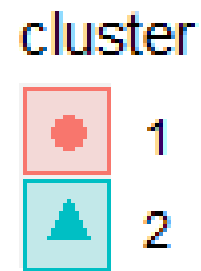
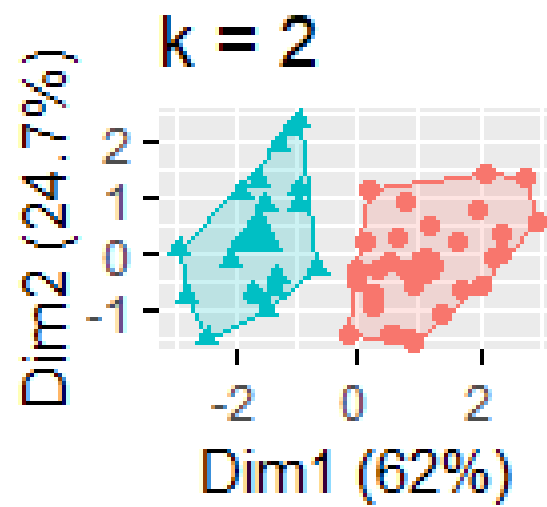
The gap statistic compares the total intracluster variation for different values of  $k$  with their expected values under null reference distribution of the data (i.e. a distribution with no obvious clustering).

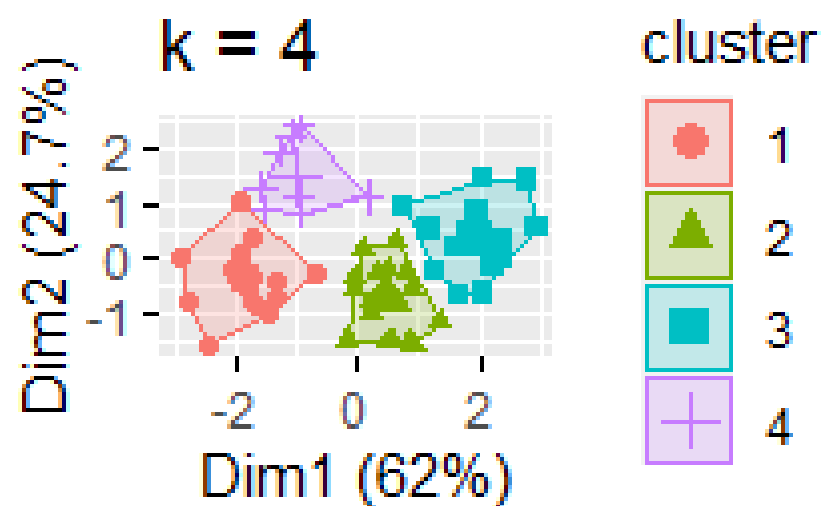
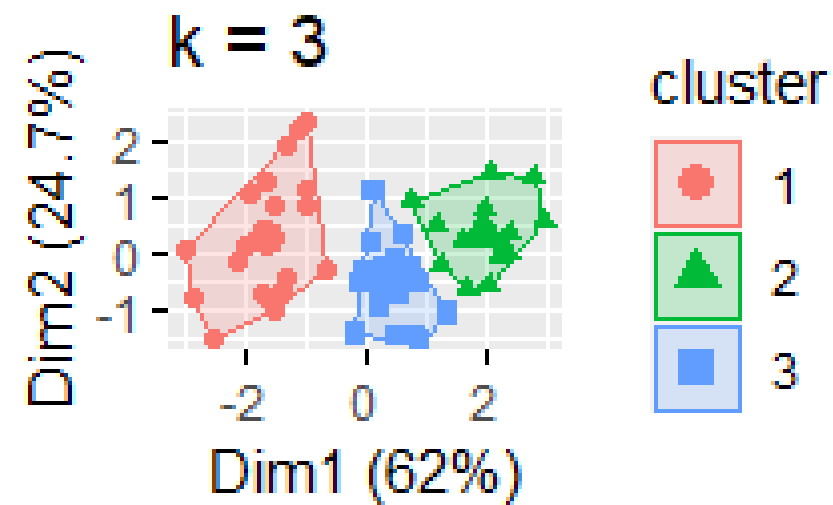
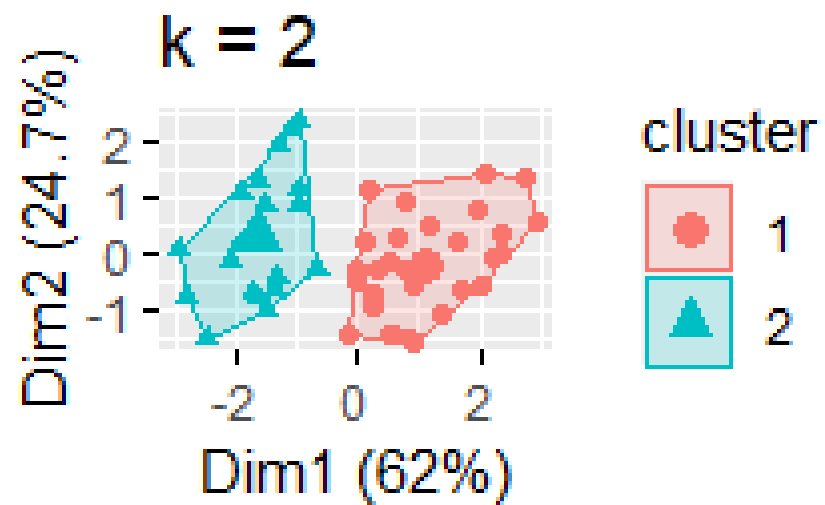
The reference data set is generated by sampling points uniformly from the minimum to the maximum value for each variable. We then choose  $B$  bootstrap samples from this reference distribution to compare to the data (compare using intracluster variation).

The “best” number of clusters is the value for  $k$  in which the gap statistics first exceeds a set critical level.

```
set.seed(123)
gap_stat = clusGap(arrest.scal, FUN = kmeans, nstart = 25, K.max = 9, B = 50)
fviz_gap_stat(gap_stat)
```







```
profile.kmeans=cbind(USArrests,k2$cluster)
```

```
all.k=profile.kmeans %>% group_by(k2$cluster) %>%
```

```
summarise(mean.assault=mean(Assault),mean.murder=m  
ean(Murder),mean.rape=mean(Rape),mean.pop=mean(U  
rbanPop))
```

**NOW, go back to original data to summarize the clusters!!**

cluster`	assault	murder	rape	pop
1	114.	4.87	15.9	63.6
2	255.	12.2	29.2	68.4

SAFE HAVENS



```
profile.kmeans=cbind(USArrests,k2$cluster)
```

```
all.k=profile.kmeans %>% group_by(k2$cluster) %>%
```

```
summarise(mean.assault=mean(Assault),mean.murder=m  
ean(Murder),mean.rape=mean(Rape),mean.pop=mean(U  
rbanPop))
```

**NOW, go back to original data to summarize the clusters!!**

cluster`	assault	murder	rape	pop
1	114.	4.87	15.9	63.6
2	255.	12.2	29.2	68.4

RISKY BUSINESS



INTERESTING LIBRARY...outputs 24 measures for clustering

```
NbClust(arrest.scal,method="kmeans",min.nc=2,max.nc = 4)
```

```
*****
```

```
*
```

```
* Among all indices:
```

```
* 13 proposed 2 as the best number of clusters
```

```
* 9 proposed 3 as the best number of clusters
```

```
* 2 proposed 4 as the best number of clusters
```

```
***** Conclusion *****
```

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

<http://www.jstatsoft.org/v61/i06/>

# Hierarchical Clustering

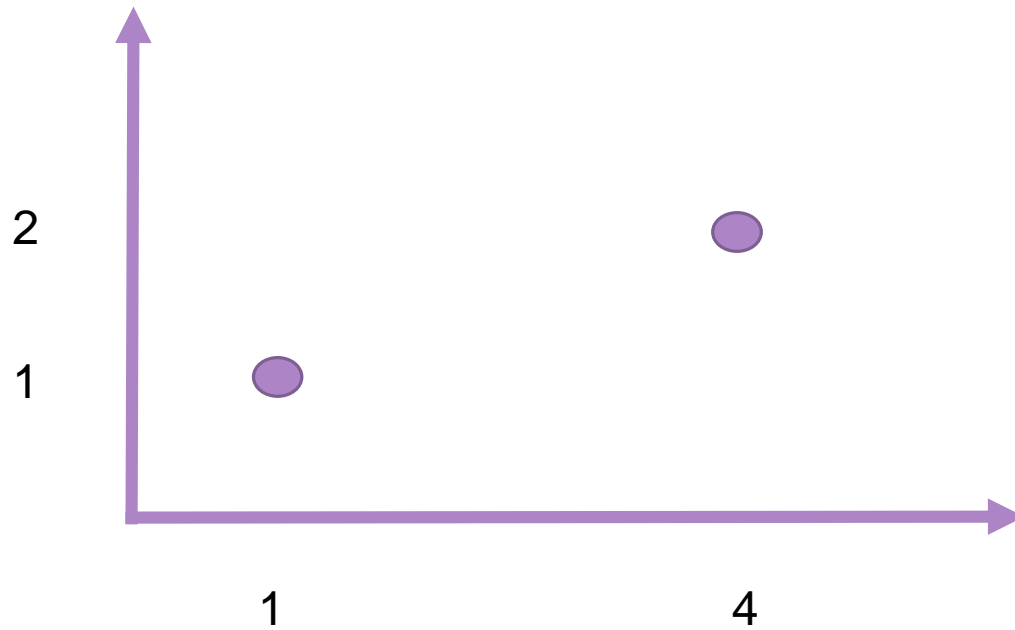
---

AGGLOMERATIVE



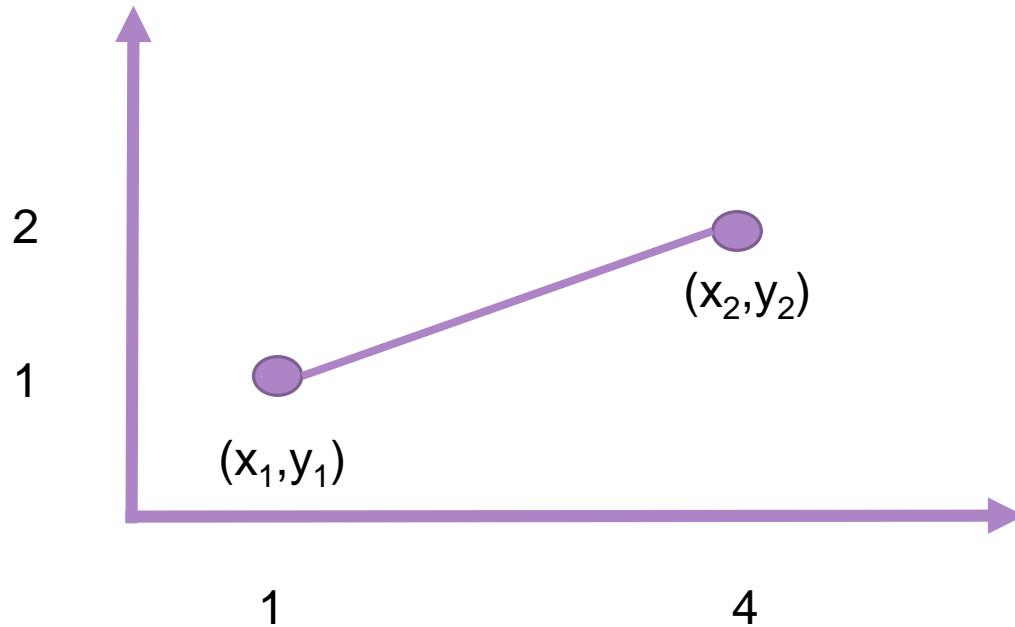
# Distance measures

---



# Distance measures

---

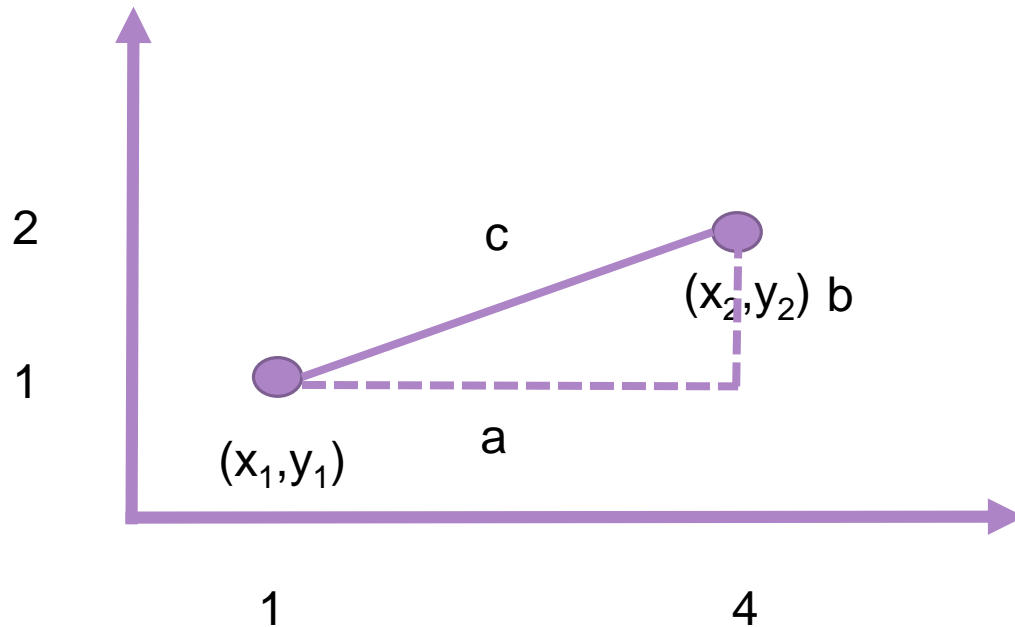


Euclidean distance between two points is:

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

# Distance measures

---



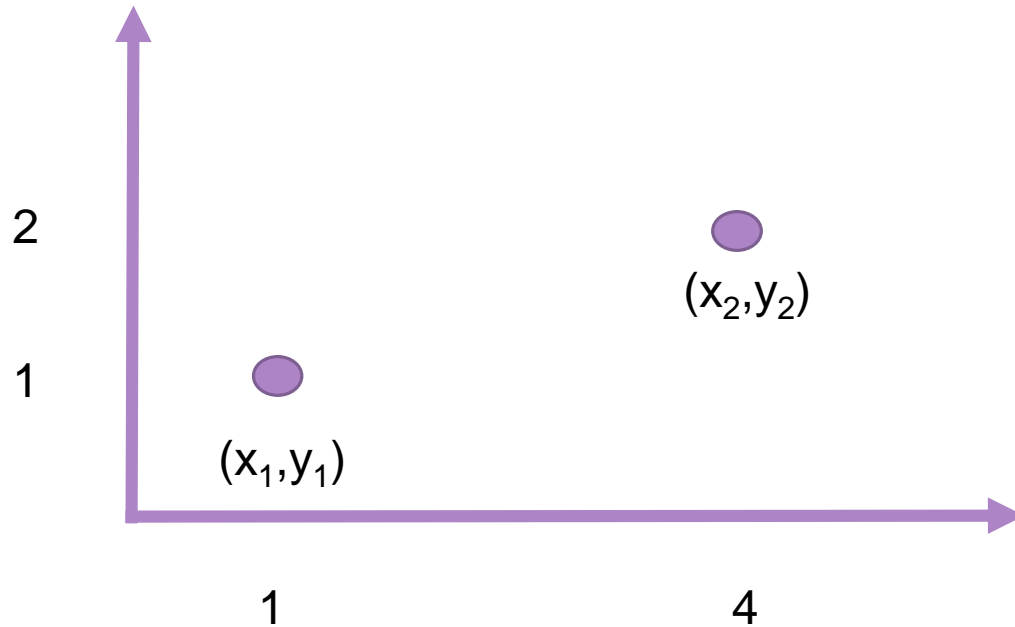
Euclidean distance between two points is:

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Pythagorean theorem:  $c = \sqrt{a^2 + b^2}$

# Distance measures

---

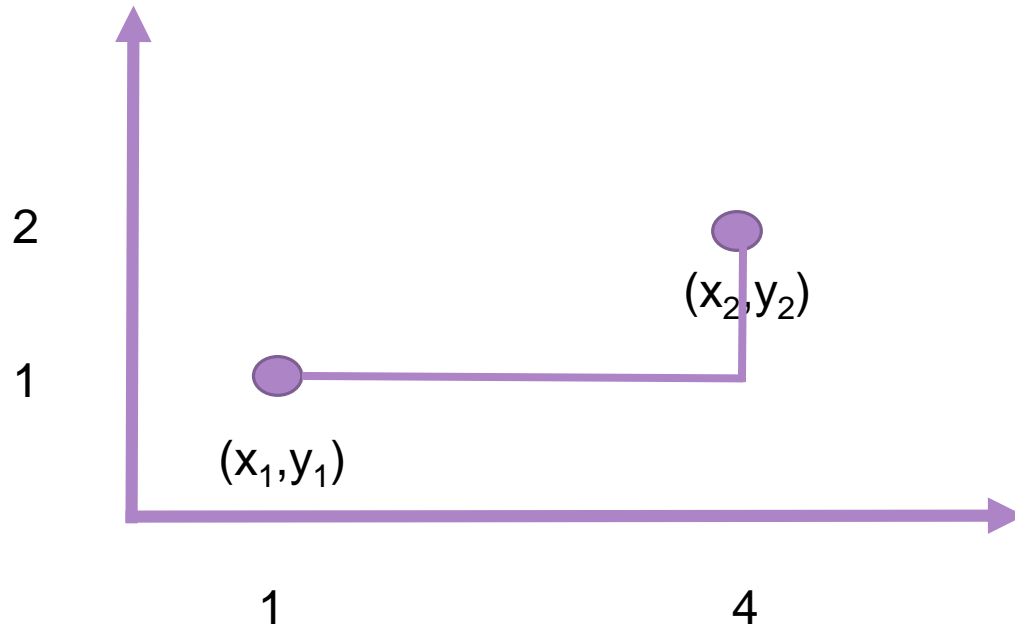


Manhattan distance between two points is:

$$|x_1 - x_2| + |y_1 - y_2|$$

# Distance measures

---



Manhattan distance between two points is:

$$|x_1 - x_2| + |y_1 - y_2|$$

# Agglomerative Hierarchical Algorithm

---

Each point starts as its own cluster (start with  $n$  clusters)

1. Calculate the distance between each point (using the specified distance measure)
2. Choose the two points that are the closest and form a cluster (now there are  $n-1$  clusters)
3. Calculate distance between all single points and all clustered points
4. Find smallest distance and combine to form a cluster

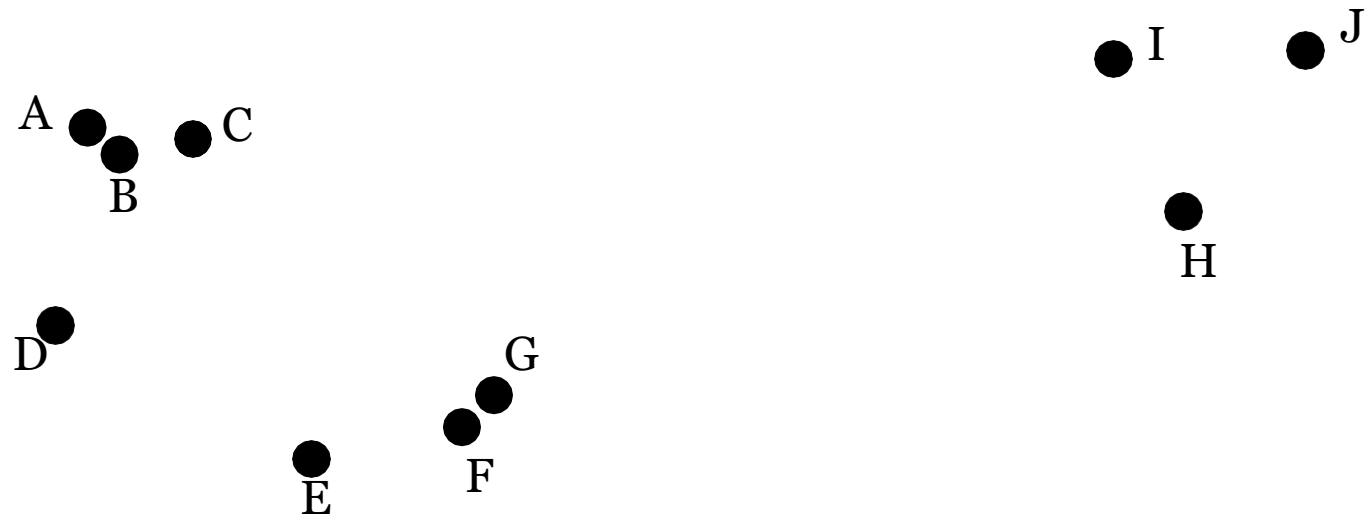
Repeat steps 3 and 4 until all observations are in one cluster

# Hierarchical Clustering

---

(Agglomerative)

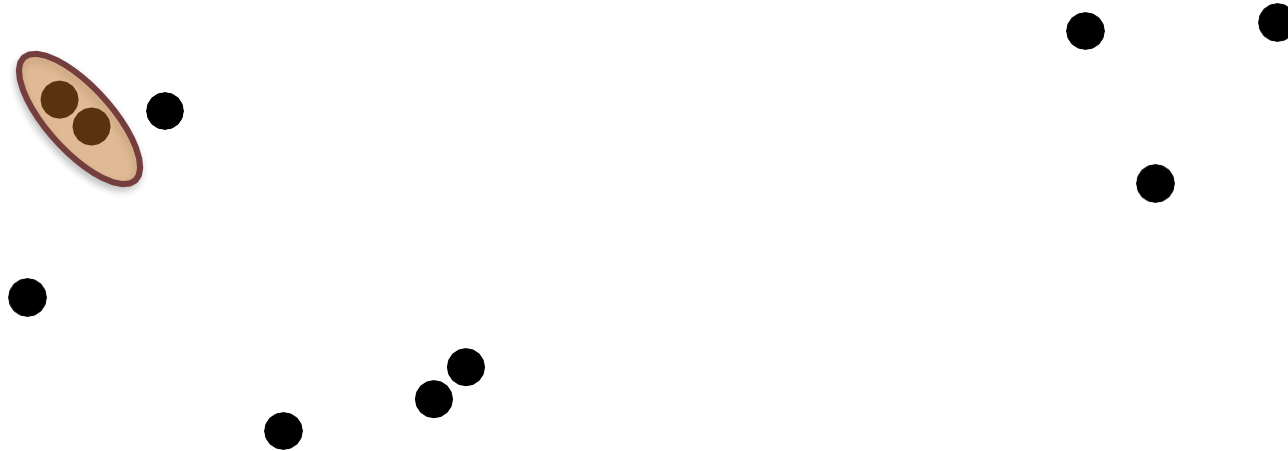
Some Data



# Hierarchical Clustering

(Agglomerative)

First Step





# Hierarchical Clustering

(Agglomerative)

Second Step



# Hierarchical Clustering

(Agglomerative)

Third Step



# Hierarchical Clustering

(Agglomerative)

Fourth Step



# Hierarchical Clustering

(Agglomerative)

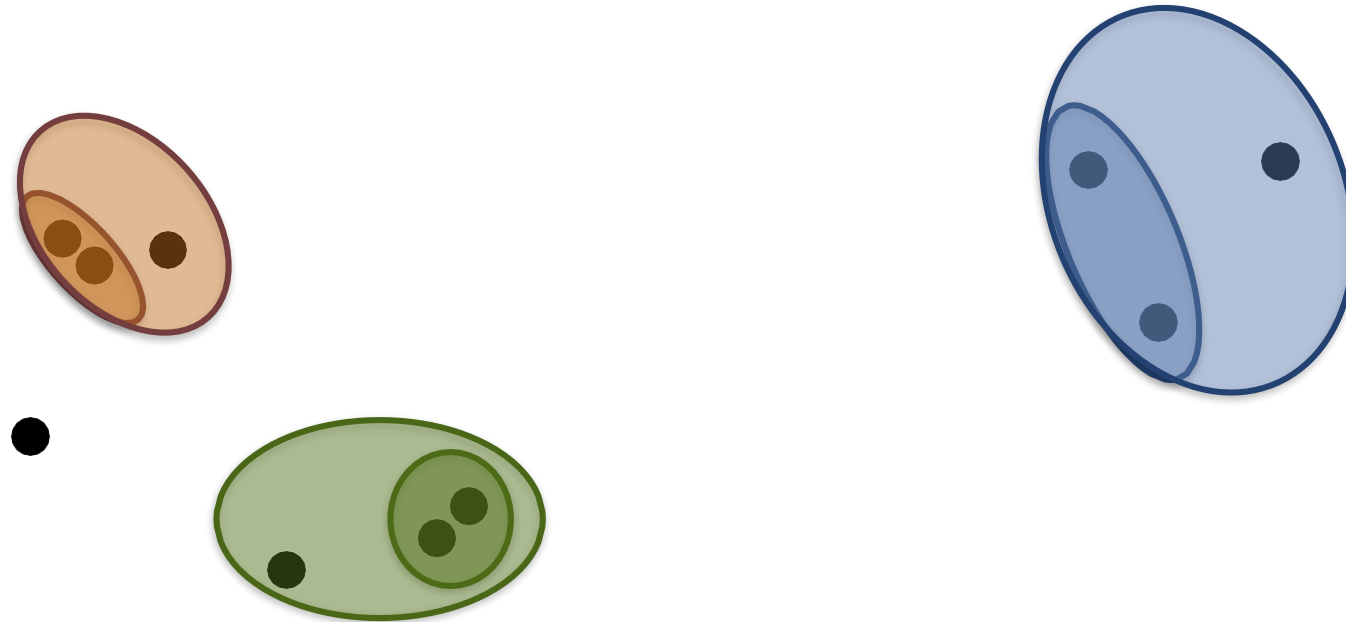
Fifth Step



# Hierarchical Clustering

(Agglomerative)

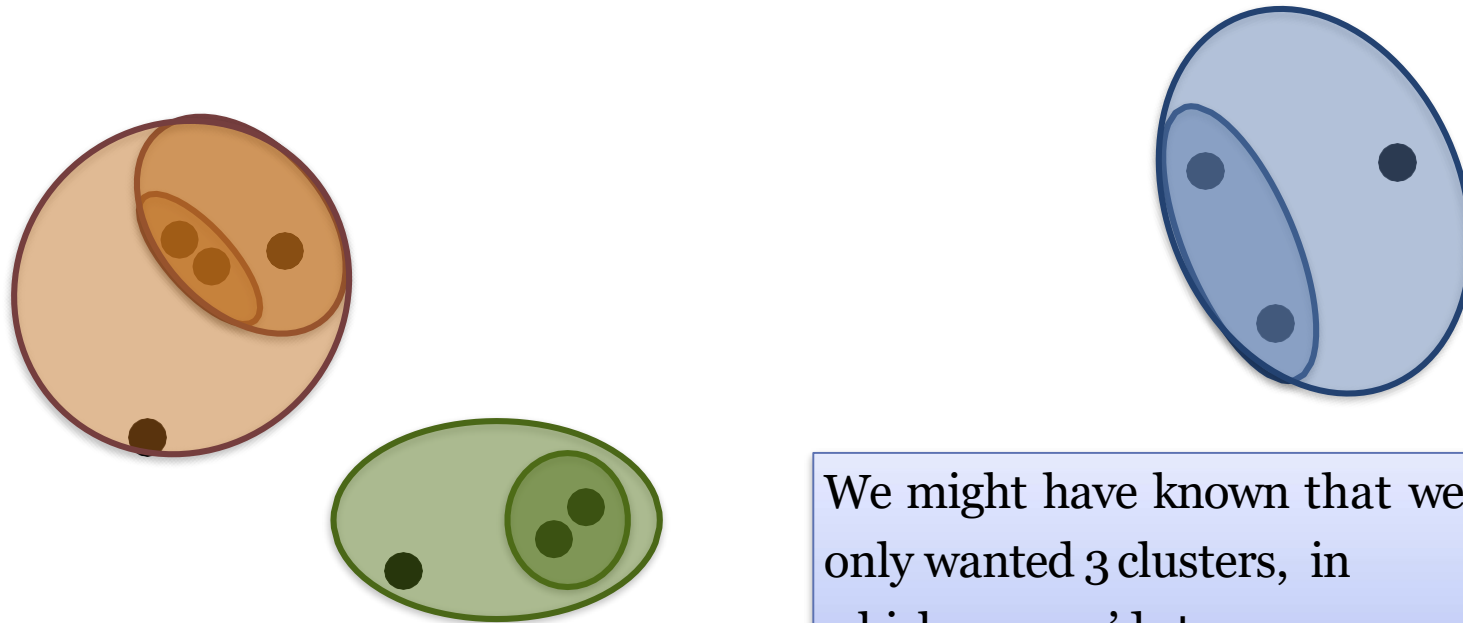
Sixth Step



# Hierarchical Clustering

(Agglomerative)

Seventh Step

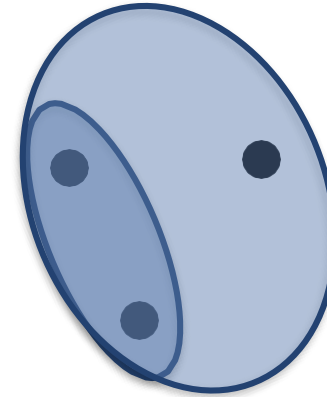
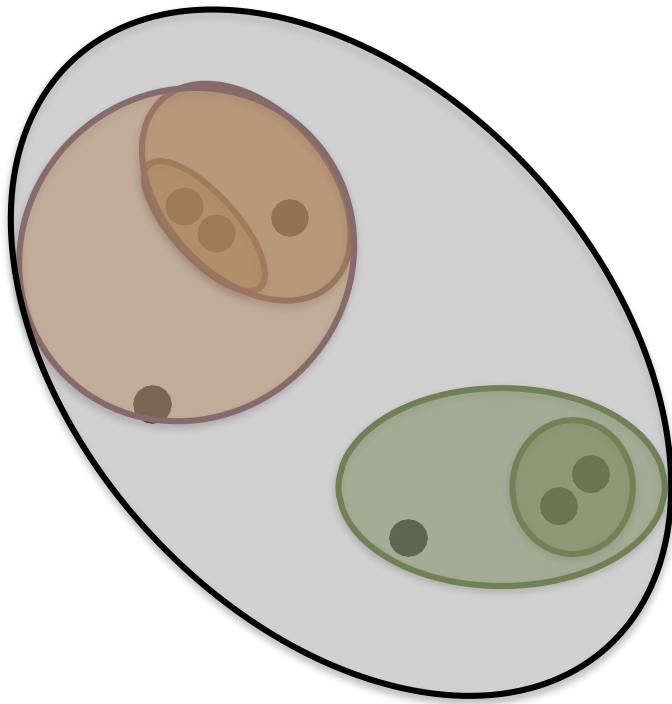


We might have known that we only wanted 3 clusters, in which case we'd stop once we had 3.

# Hierarchical Clustering

(Agglomerative)

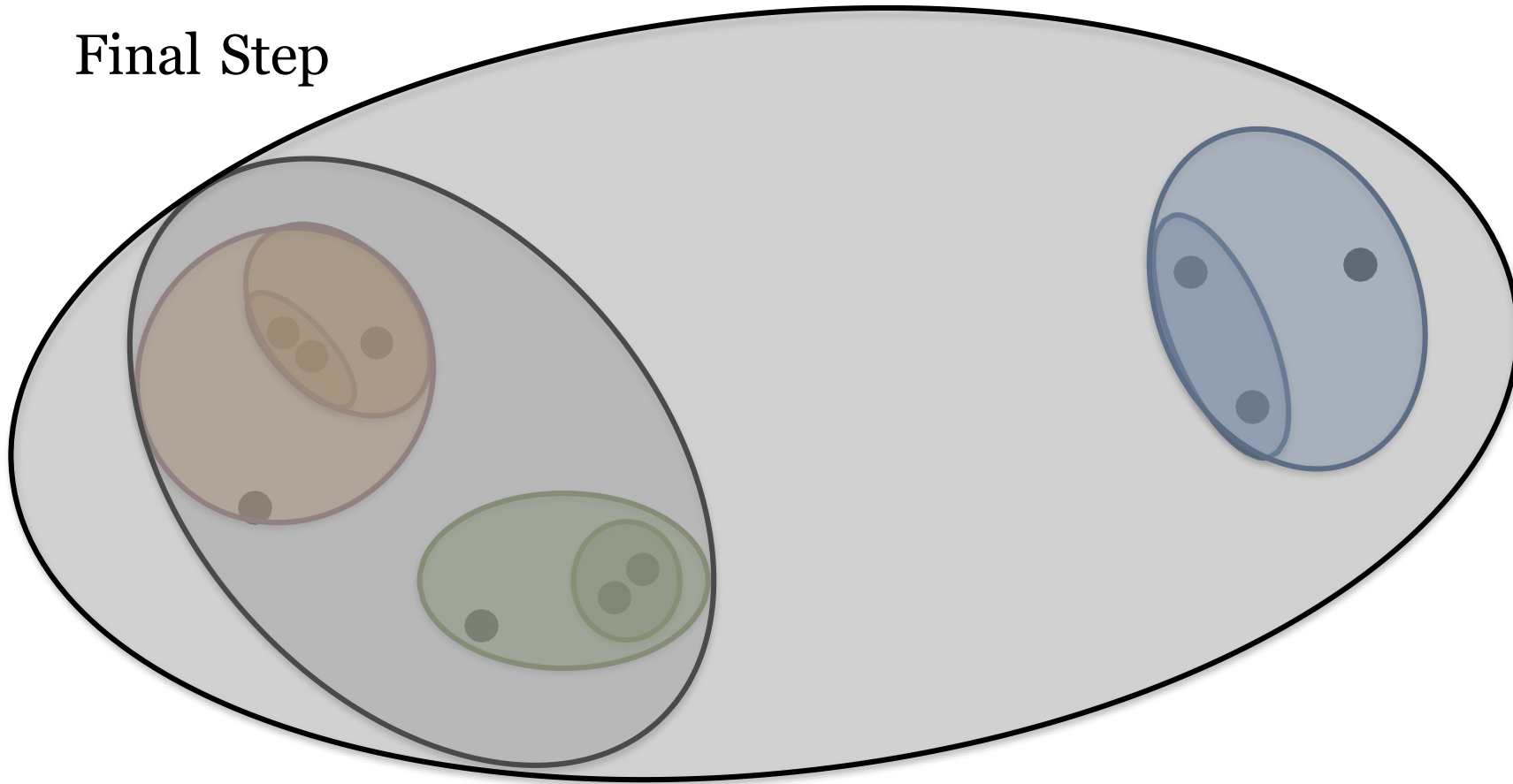
Eighth Step



# Hierarchical Clustering

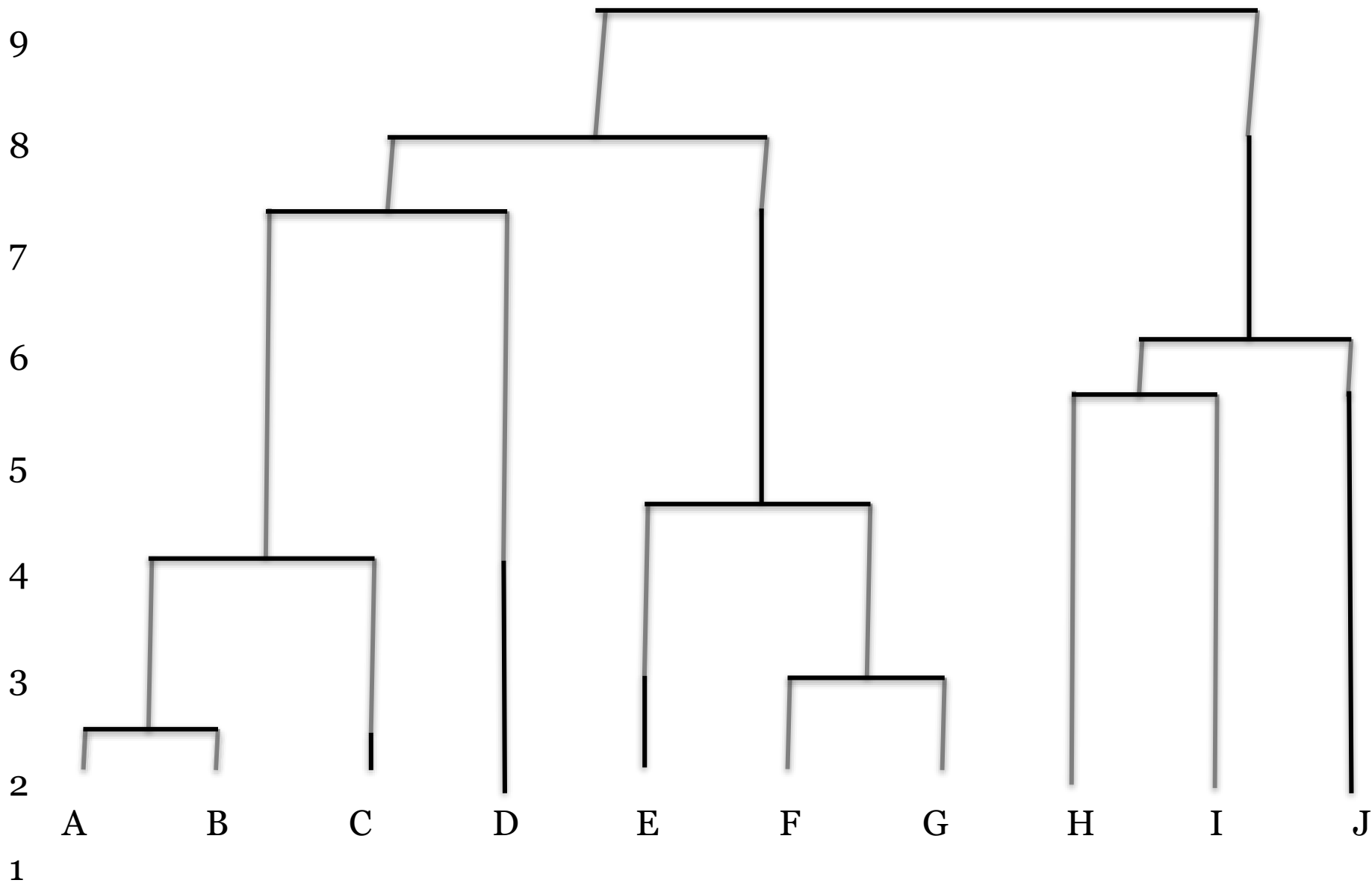
(Agglomerative)

Final Step





# Resulting Dendrogram (not to scale)

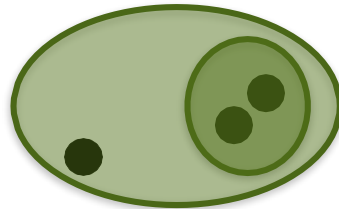


# Linkages

---

Which clusters/points are closest to each other?

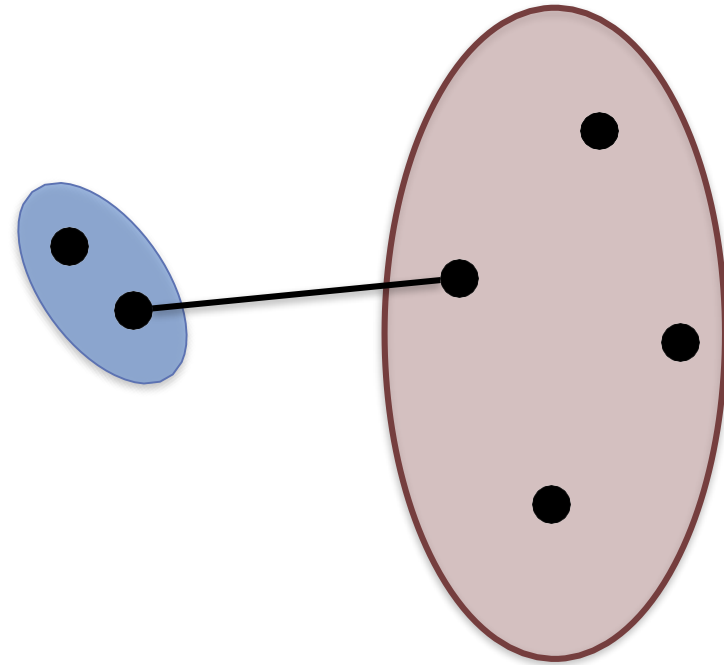
How do I measure the distance between a point/cluster and a cluster?



# Linkages

---

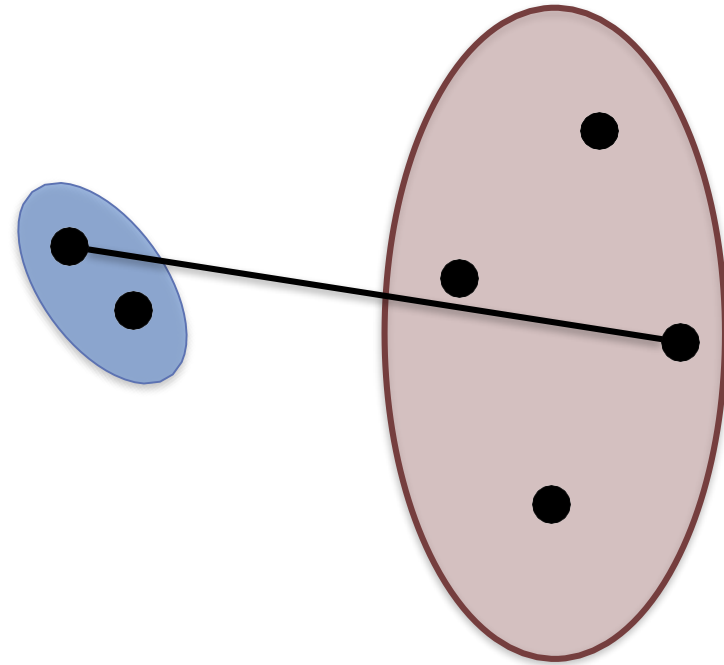
**Single Linkage:** Distance between the closest points in the clusters.



# Linkages

---

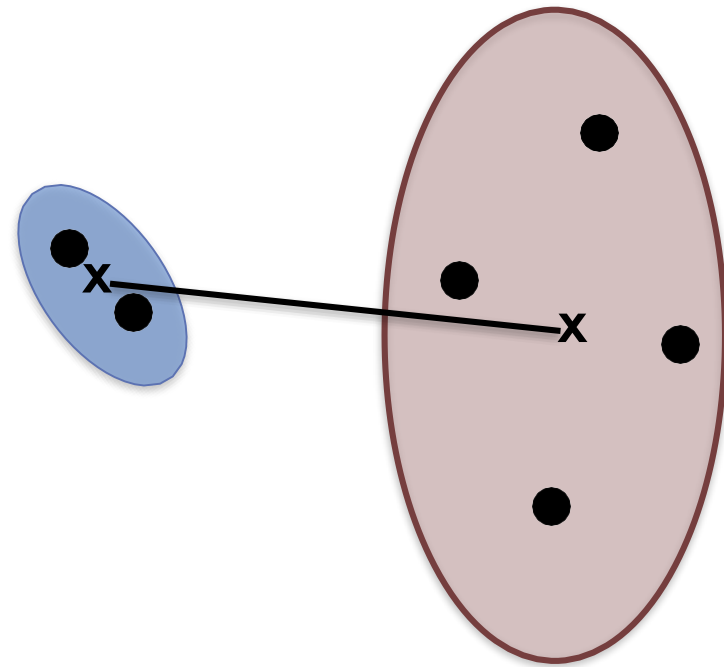
**Complete Linkage:** Distance between the farthest points in the clusters.



# Linkages

---

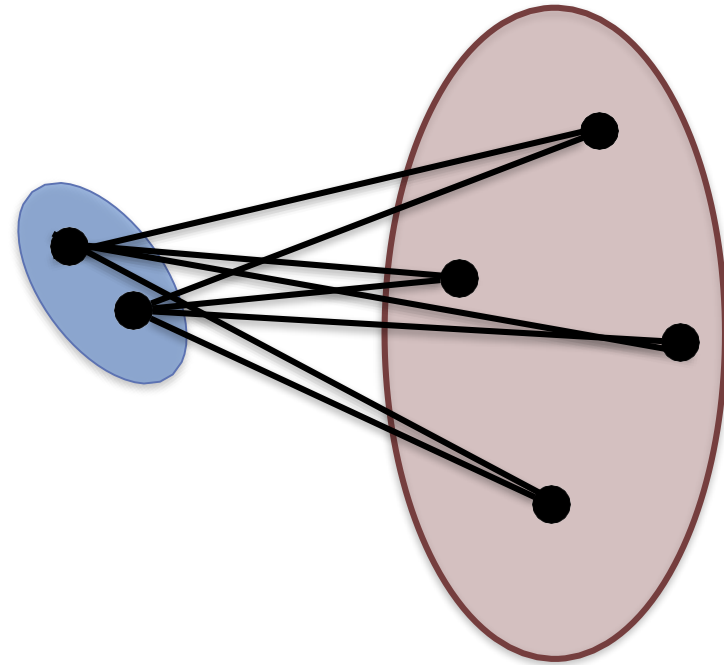
**Centroid Linkage:** Distance between the centroids (means) of each cluster.



# Linkages

---

**Average Linkage:** Average distance between all points in the clusters.

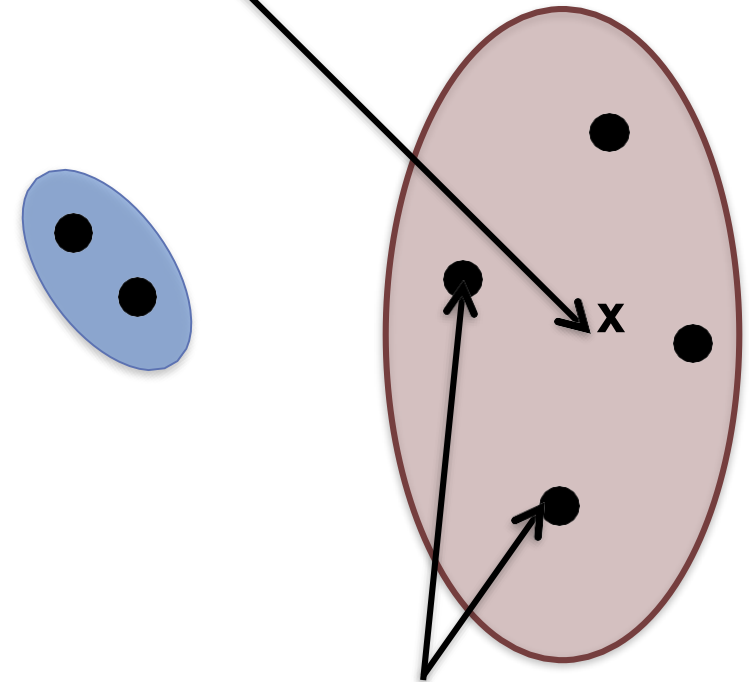


# Linkages

**Ward's Method:** Minimize SSE.

$$\sum_{j=1}^{N_i} \|\mathbf{x}_j - \mathbf{c}_i\|_2$$

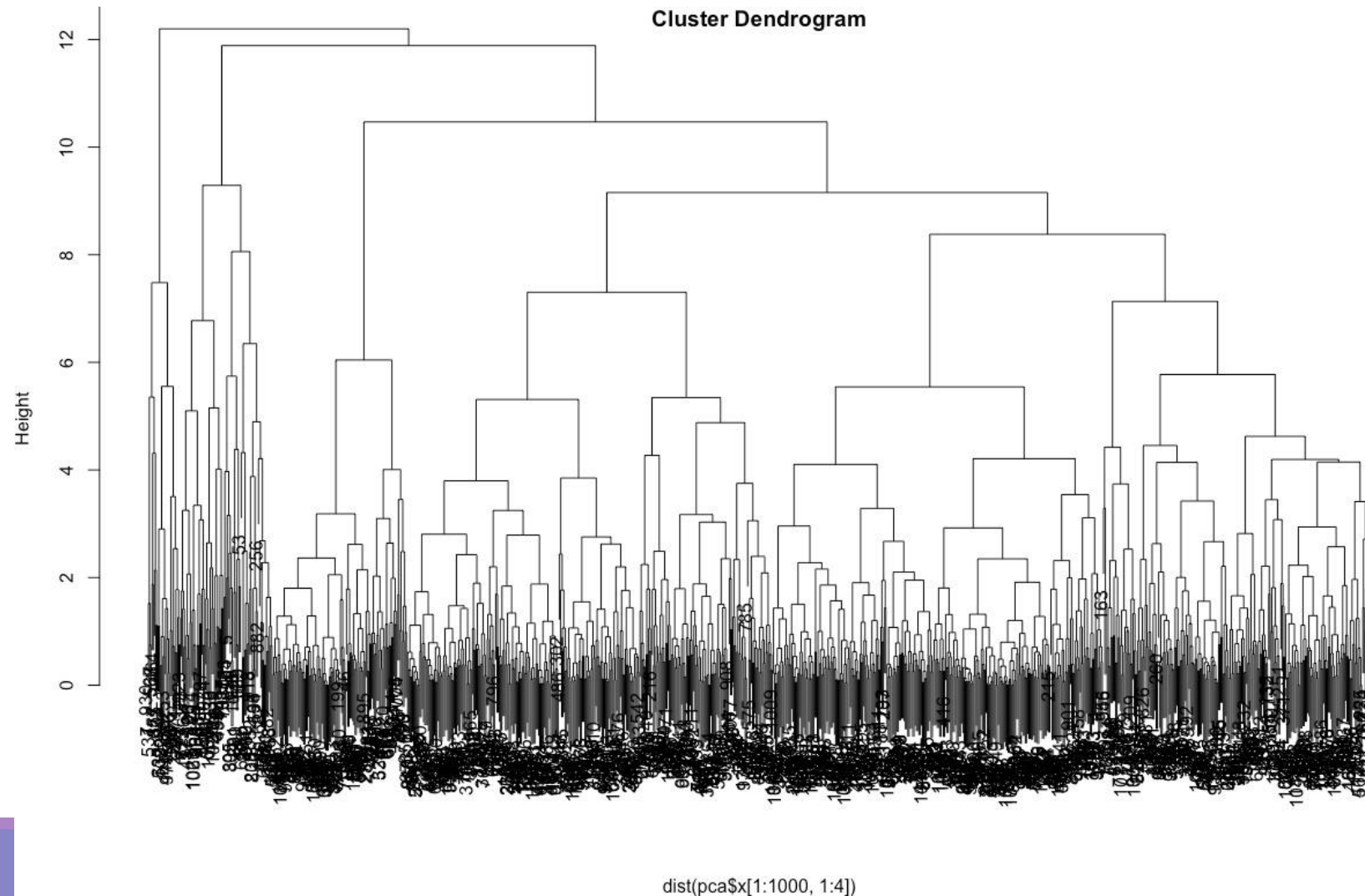
centroid for cluster i,  $\mathbf{c}_i$



data points in cluster i:

$\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{N_i}$

# Hierarchical Clustering Summary





# Hierarchical Clustering Summary

## Disadvantages

- Computationally intensive, large storage requirements, **not good for large datasets**
- **Lacks global objective** function: only makes decision based on local criteria.
- **Merging decisions are final.** Once a point is assigned to a cluster, it stays there.
- **Poor performance on noisy** or high-dimensional **data** like text.

## Advantages

- **Creates hierarchy (dendrogram)** that can help choose the number of clusters and examine how those clusters relate to each other.
- **Do NOT need to know number of clusters apriori**

# US Arrest data

---

# Number of different hierarchical packages in R

---

We will focus on the cluster package (start with agnes...Agglomerative “Nesting” Hierarchical Clustering) and the Nbclust package

In the cluster package, recommend calculating distances outside of package (more options!!). In this case, you will pass the clustering algorithm the DISTANCE matrix!

# Options

---

## DISTANCE (IN DIST)

Euclidean

Manhattan

Maximum

Canberra

Binary

Minkowski

## LINKAGE (IN CLUSTER ALGORITHM)

Average

Single

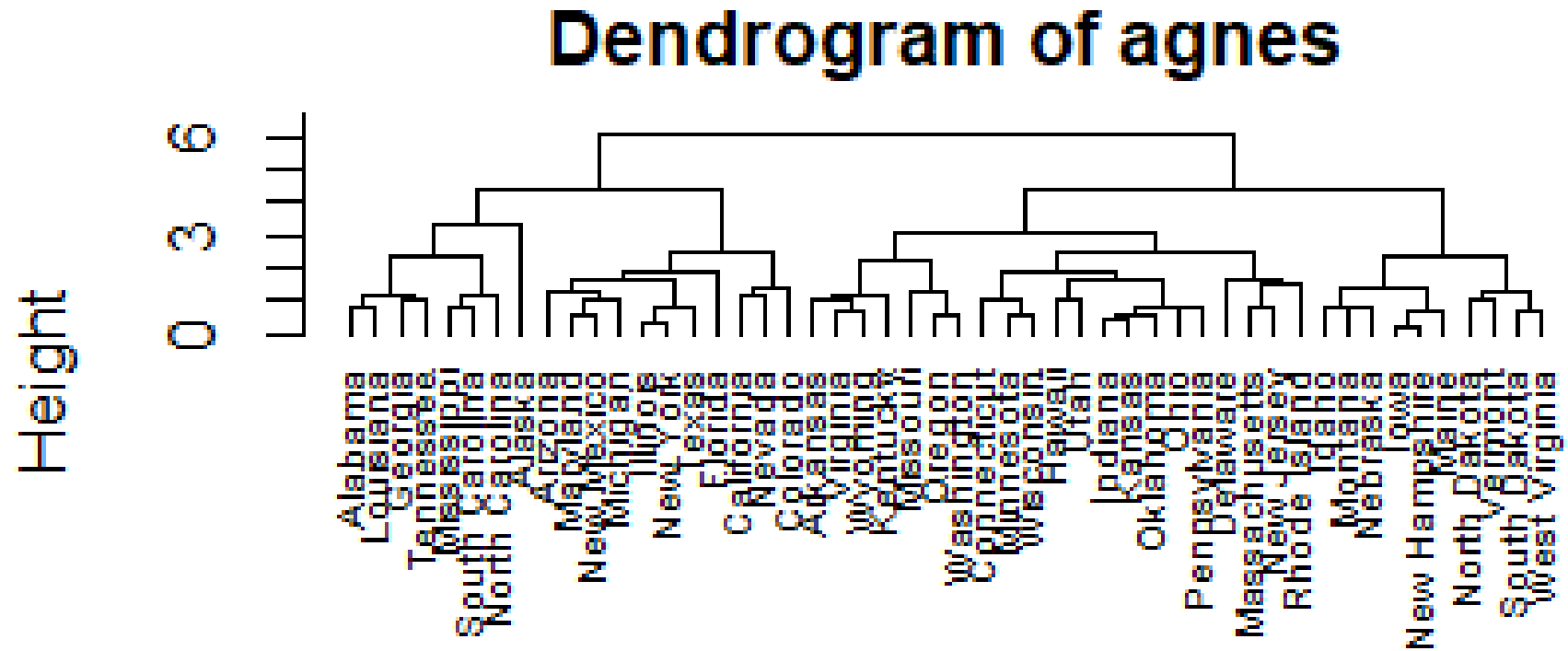
Complete

Ward

Weighted

Some more advanced options...

```
dist.assault=dist(arrest.scal,method = "euclidean")  
h1.comp.eucl=agnes(dist.assault,method="complete")  
pltree(h1.comp.eucl, cex = 0.6, hang = -1, main = "Dendrogram of agnes")
```

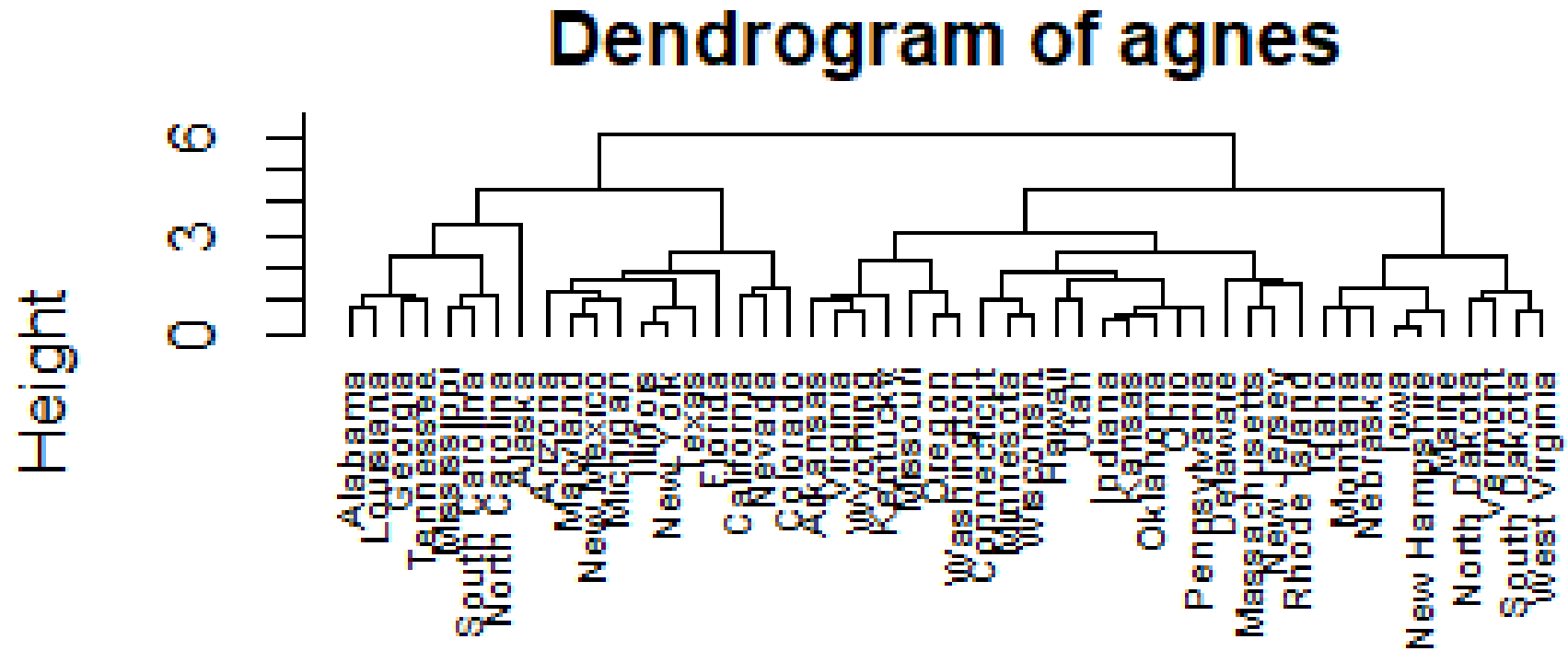


dist.assault  
agnes (\*, "complete")

```
dist.assault=dist(arrest.scal,method = "euclidean")
```

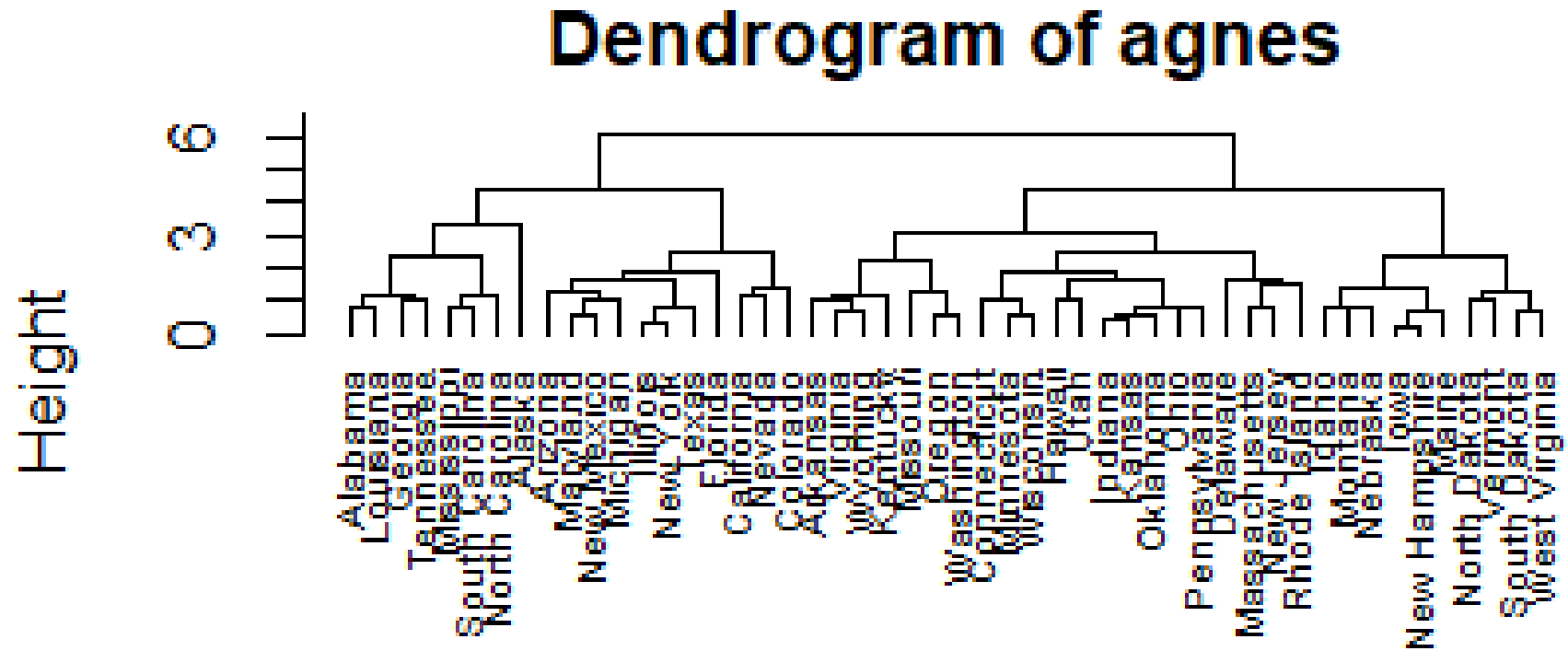
```
h1.comp.eucl=agnes(dist.assault,method="complete")
```

```
pltree(h1.comp.eucl, cex = 0.6, hang = -1, main = "Dendrogram of agnes")
```



dist.assault  
agnes (\*, "complete")

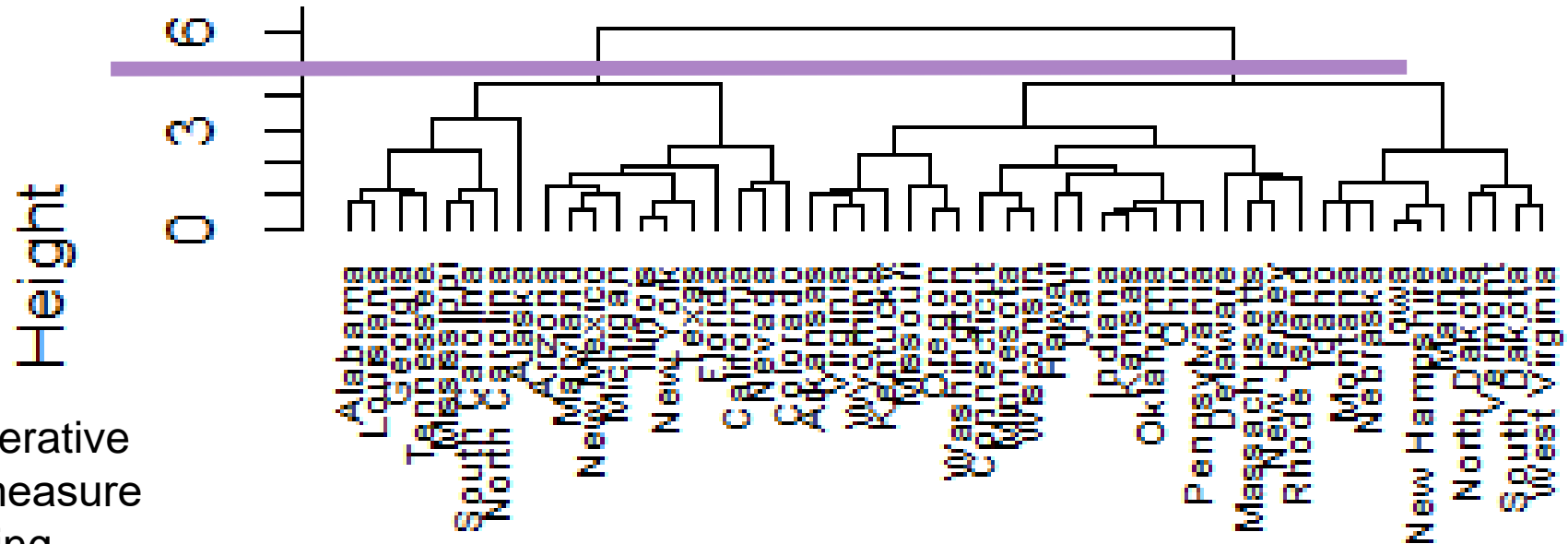
```
dist.assault=dist(arrest.scal,method = "euclidean")  
h1.comp.eucl=agnes(dist.assault,method="complete")  
pltree(h1.comp.eucl, cex = 0.6, hang = -1, main = "Dendrogram of agnes")
```



dist.assault  
agnes (\*, "complete")

```
dist.assault=dist(arrest.scal,method = "euclidean")  
h1.comp.eucl=agnes(dist.assault,method="complete")  
pltree(h1.comp.eucl, cex = 0.6, hang = -1, main = "Dendrogram of agnes")
```

## Dendrogram of agnes



```
h1.comp.eucl$ac  
[1] 0.8531583
```

This is called the agglomerative coefficient...provides a measure of how strong the clustering structure is (want values close to 1)

```
dist.assault  
agnes (*, "complete")
```



# Results from hierarchical clustering

---

You will most likely get different results when

1. You use different distance metrics
2. You use different linkage methods
3. Use scaled data versus PCA versus different types of standardization

# Create algorithm to run through possibilities:

---

```
m <- c( "average", "single", "complete", "ward")
names(m) <- c( "average", "single", "complete", "ward")

# function to compute coefficient
ac <- function(x) {
  agnes(dist.assault, method = x)$ac
}

map_dbl(m, ac)
```

average	single	complete	ward
0.7379371	0.6276128	0.8531583	0.9346210

# CAUTION

---

Want best algorithm for business sense (creates the most sensible clusters). This might not be the best mathematical cluster!! You will need to explore profiles to see which makes the most sense!!

# Create clusters in hierarchical

---

```
h2=agnes(dist.assault,method="ward")  
h2_clus <- cutree(h2, k = 2)  
H2_clus
```

```
[1] 1 1 1 2 1 1 2 2 1 1 2 2 1 2 2 2 2 1 2 1 2 1 2 1 2 2 2 1 2  
[30] 2 1 1 1 2 2 2 2 2 2 1 2 1 1 2 2 2 2 2 2 2
```

# NbClust with hierarchical clustering

---

```
NbClust(arrest.scal,distance="euclidean",method="complete",min.nc=2,max.nc = 4)
```

```
*****
```

```
*
```

```
* Among all indices:
```

```
* 9 proposed 2 as the best number of clusters
```

```
* 9 proposed 3 as the best number of clusters
```

```
* 5 proposed 4 as the best number of clusters
```

```
***** Conclusion *****
```

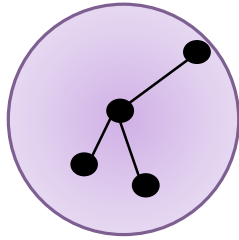
```
* According to the majority rule, the best number of  
clusters is 2
```

# DBSCAN

---

Density-based spatial clustering of applications with noise

Groups together points that are close to each other based on a distance measure, a minimum number of points (“neighbors”) and a “neighborhood about each point”



Points not “near” other points will be deemed an “outlier”

A “cluster” of points must have a minimum number of points around it to be considered a cluster

# Example how it works....

---

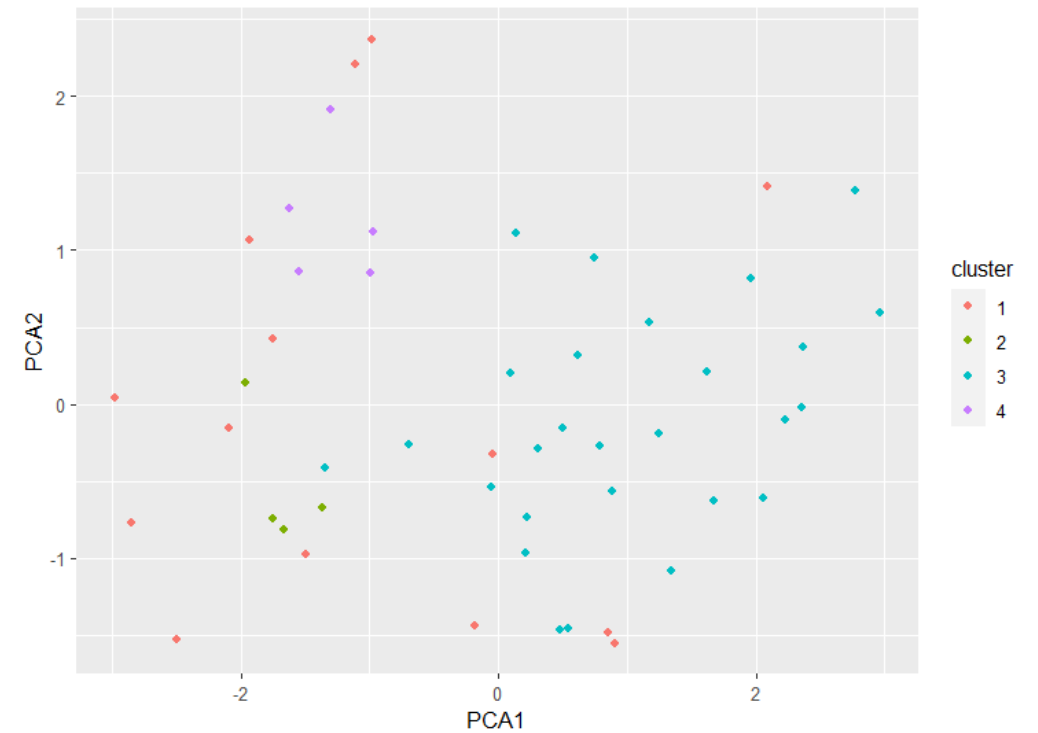
<https://www.kdnuggets.com/2020/04/dbscan-clustering-algorithm-machine-learning.html>

# HDBSCAN (hierarchical)

---

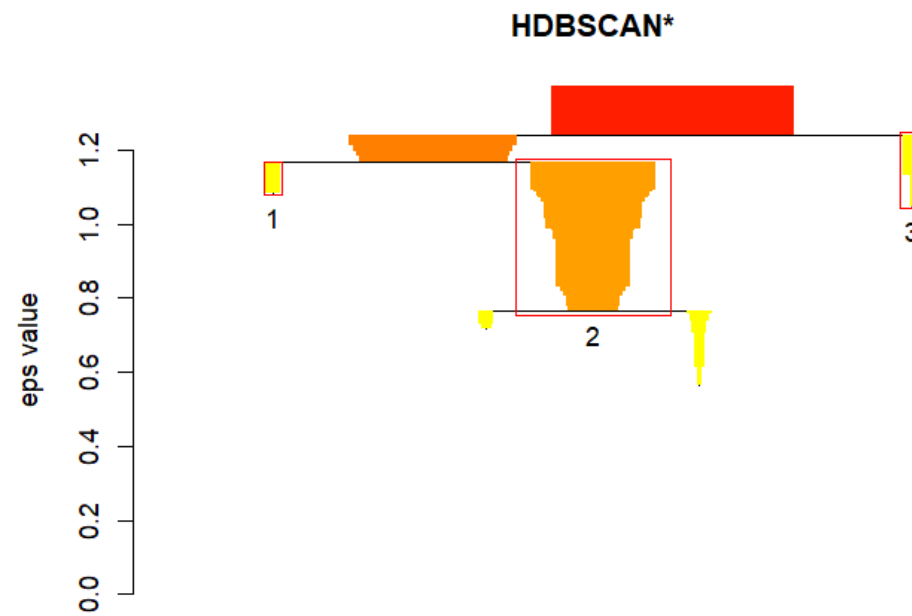
minPts – usually try  $d + 1$  ( $d$ =number of dimensions...here  $d=4$ .... $d=4$  works better than  $d=5$ )

```
scan1<-hdbscan(arrest.scal,minPts=4)
pca_ex=prcomp(arrest.scal,scale=F)
scan1data=cbind.data.frame(pca_ex$x[,1],pca_ex$x[,2],as.factor(scan1$cluster+1))
colnames(scan1data)=c("PCA1","PCA2","cluster")
ggplot(scan1data,aes(x=PCA1,y=PCA2,color=cluster))
+geom_point()+ scale_fill_brewer(palette = "Dark2")
```





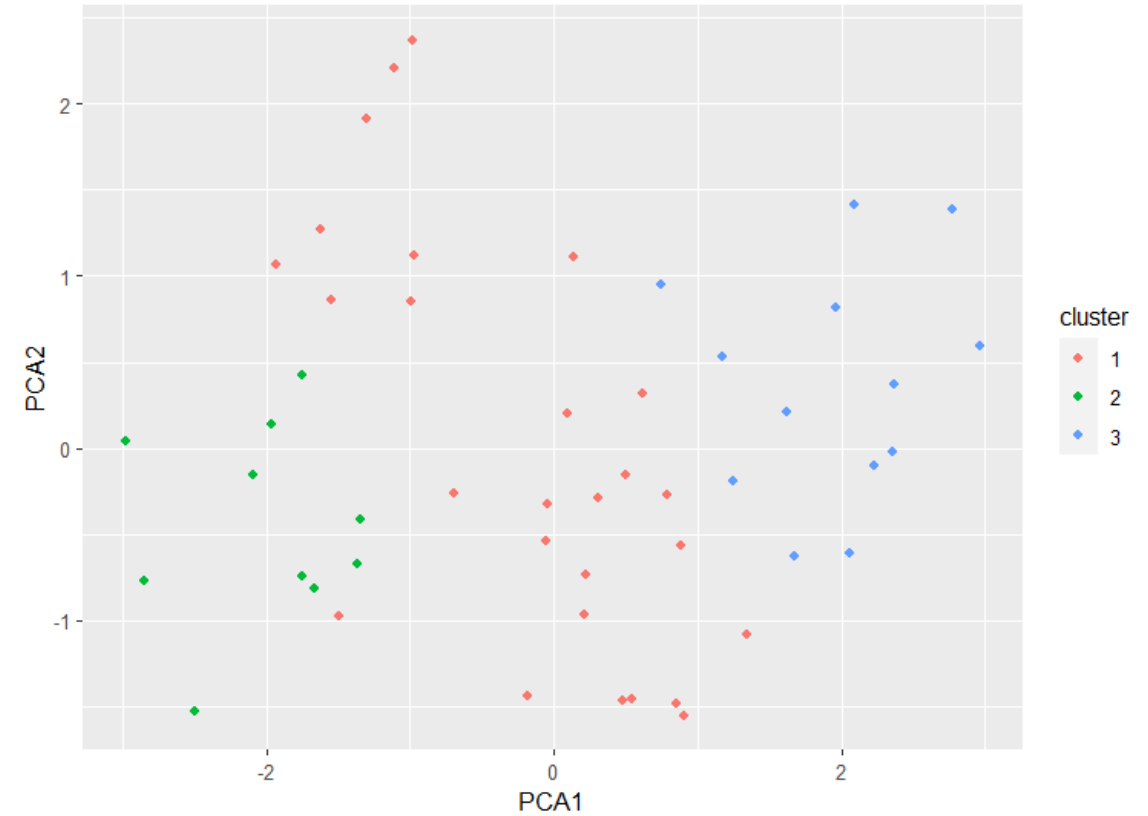
```
plot(scan1,show_flat=T)
```



[https://hdbscan.readthedocs.io/en/latest/how\\_hdbscan\\_works.html](https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html)

```
d=dist(arrest.scal,method = "canberra")
res.dbscan=dbscan(d,eps=1.2,minPts=4)
res.dbscan
```

```
scan1data=cbind.data.frame(pca_ex$x[,1],pca_ex$x[,2],a
s.factor(res.dbscan$cluster+1))
colnames(scan1data)=c("PCA1","PCA2","cluster")
ggplot(scan1data,aes(x=PCA1,y=PCA2,color=cluster))+g
eom_point()+ scale_fill_brewer(palette = "Dark2")
```



# A few other clustering algorithms....

---

Diana – divisive hierarchical clustering (Cluster package)

Pam – Partitioning around Medoids (Cluster package)

Clara - Clustering Large applications (Cluster package)

Mona - Monothetic Analysis Clustering of Binary Variables (Cluster package)

k-prototypes clustering (clustMixType...can do categorical and numeric variables)

# Variable Clustering

---

# Variable Clustering in R

---

Goal: We would like to cluster variables that are related (reduce redundancies/multicollinearity)

This is a form of dimension reduction (can aid in analysis)

In R, the package is ClustOfVar...you can do hierarchical clustering (agglomerative in R....SAS and Python do divisive) or kmeans

Uses eigenvalues to identify similar variables and to assess the goodness of the partition

Can handle quantitative and qualitative variables (you need to split these into two data matrices first)

# Example

---

TELCO CHURN DATA

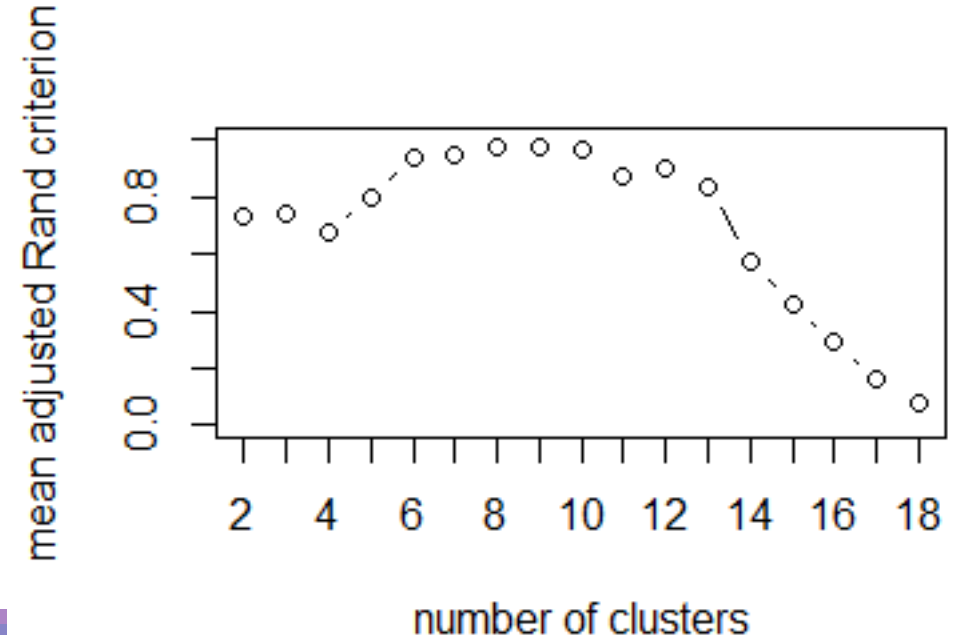
# Preprocessing

---

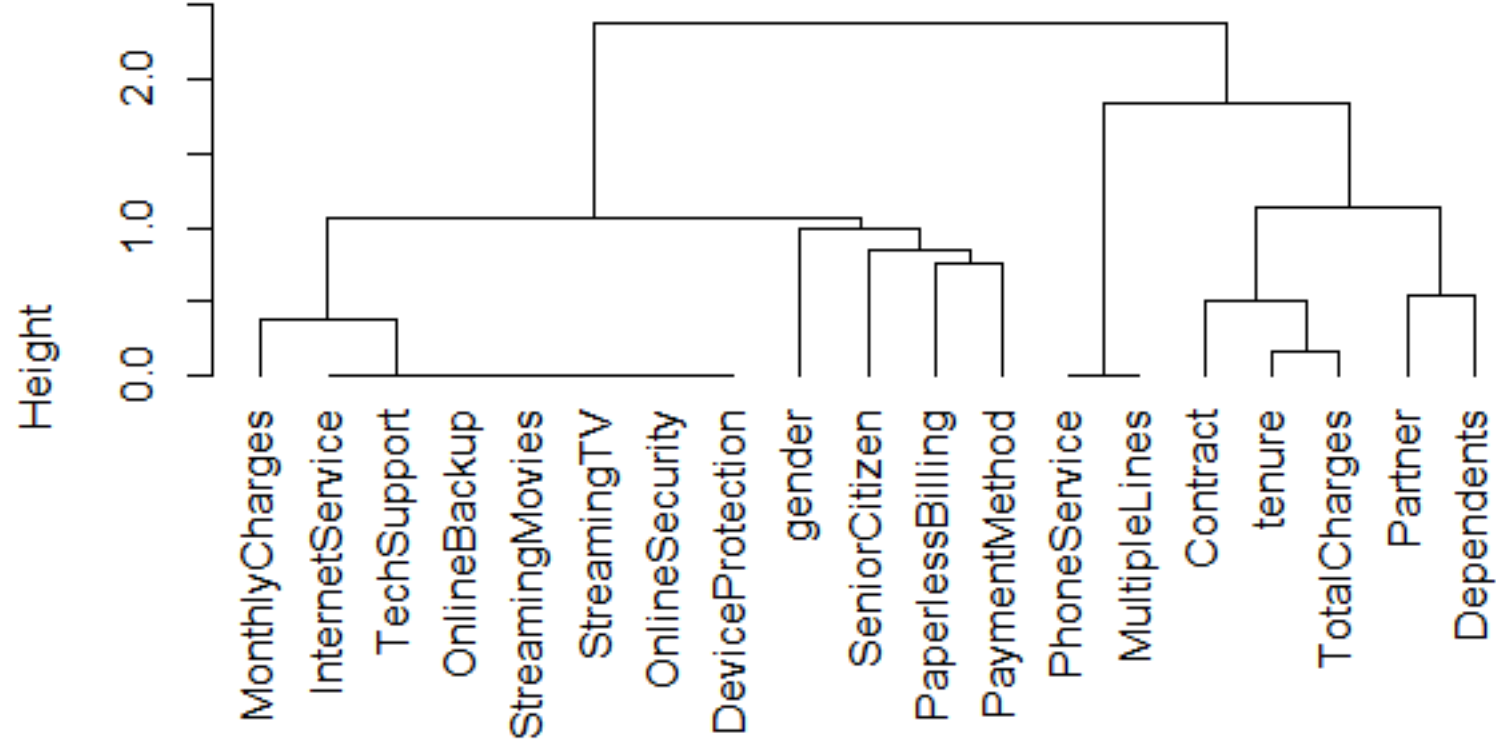
Split into quantitative and qualitative variables (no need to dummy code...all is done within algorithm)

Impute missing values (in this case, makes sense to replace missing values in total charges with 0)

```
telco[is.na(telco)] = 0  
quant.var=telco[,c(3,6,19,20)]  
qual.var=telco[,c(2,4,5,7:18)]  
  
var.clust.h=hclustvar(quant.var, qual.var)  
stab=stability(var.clust.h,B=50)  
plot(stab)
```



**Cluster Dendrogram**





```
h6=cutreevar(var.clust.h, 6)
```

h6\$cluster

SeniorCitizen	tenure	MonthlyCharges
1	2	3
TotalCharges	gender	Partner
2	4	5
Dependents	PhoneService	MultipleLines
5	6	6
InternetService	OnlineSecurity	OnlineBackup
3	3	3
DeviceProtection	TechSupport	StreamingTV
3	3	3
StreamingMovies	Contract	PaperlessBilling
3	2	1
PaymentMethod		
1		

# Questions?

---

