# Cluster Analysis

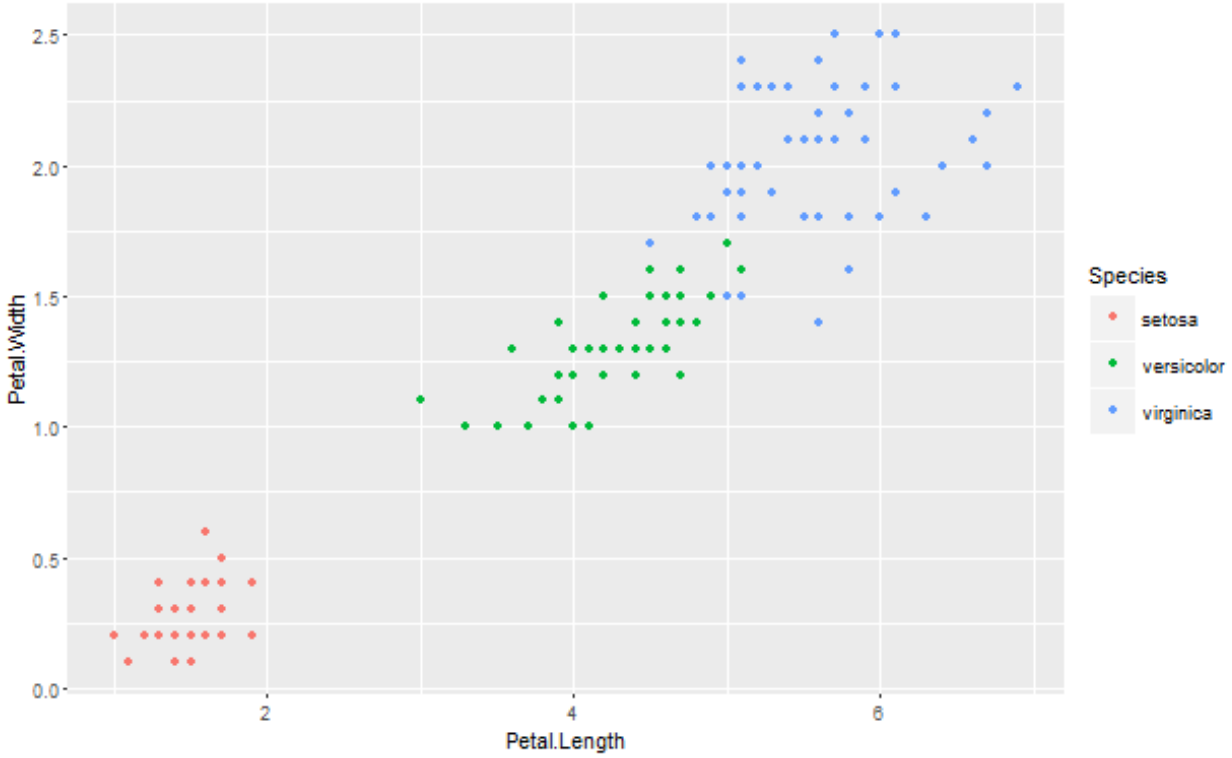
Anahita Zarei, Ph.D.

# Overview

- K-means
- Hierarchical
  - Agglomerative clustering (AGNES - Bottom-up)
  - Divisive hierarchical clustering (DIANA - Top-down)
- Elbow Method
- Silhouette Score

# Clustering – Unsupervised Classification

- Cluster: a collection of data objects
  - Similar to one another within the same cluster
  - Dissimilar to the objects in other clusters
- Clustering is unsupervised classification, i.e. no predefined classes
- Assessment of clustering quality is application-dependent and to an extent subjective
- Typical applications
  - As a stand-alone tool to get insight into data distribution
  - As a preprocessing step for other algorithms (e.g. Use of K means clustering in RBF models)



# Examples of Clustering Applications (as a stand-alone tool)

- Marketing: Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs
- Urban planning: Identifying groups of houses according to their house type, value, and geographical location
- Politics: Help campaign managers to identify voters with similar interests
- Insurance: Identifying groups of motor insurance policy holders with a high average claim cost

# K-means Clustering

- The k-means clustering algorithm is a simple yet powerful tool for obtaining clusters and cluster centers.
- The goal of k-means clustering is to
  1. Select centers $\mu_1,\ldots,\mu_k$ for each cluster
  2. Assign input data points $x_1,\ldots,x_N$ into K cluster sets $S_1,\ldots S_K$
- The centers are representative of the data if every data point in cluster $S_k$ is close to its corresponding center $\mu_k$.
- For cluster $S_k$ with center $\mu_k$, define the squared error measure to quantify the quality of the cluster:

$$\sum_{\mathbf{x}_n \in S_k} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$$

- The k-means error function just sums this cluster error over all clusters:

$$\sum_{k=1}^{K} \sum_{\mathbf{x}_n \in S_k} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$$

# K-means Clustering Algorithm

- Objective: Minimize the distance between $x_n$ and the closest enter $\mu_k$ by splitting $x_1, \cdots, x_N$ into clusters $S_1, \cdots, S_K$

$$\text{Minimize} \sum_{k=1}^{K} \sum_{\mathbf{x}_n \in S_k} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$$

- Finding the global minimum of the above cost function is intractable (similar to the NN case).
- We seek an iterative approach to find a local minimum.

# Lloyd's algorithm

- Select initial centroids at random.
- Assign each object to the cluster with the nearest centroid:

$$S_k \leftarrow \{\mathbf{x}_n : \|\mathbf{x}_n - \boldsymbol{\mu}_k\| \leq \text{all } \|\mathbf{x}_n - \boldsymbol{\mu}_\ell\|\}$$

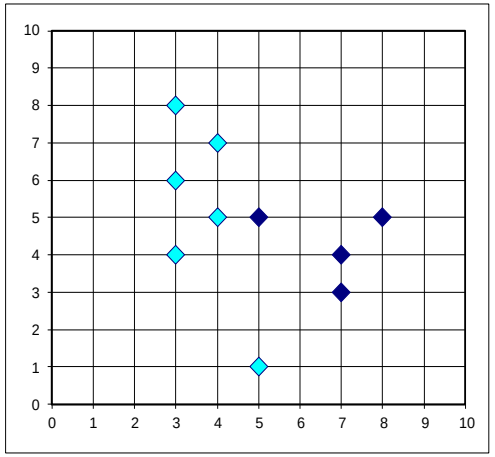
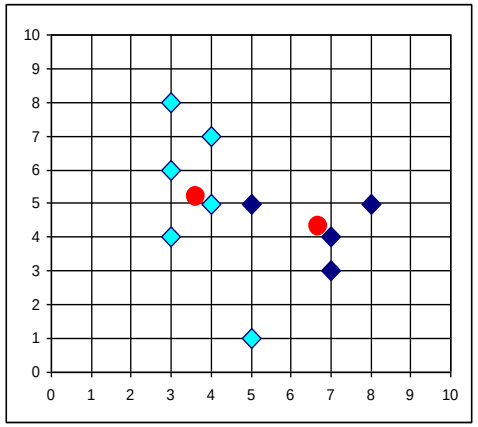
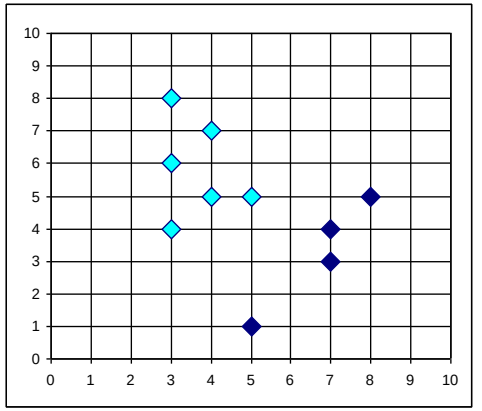
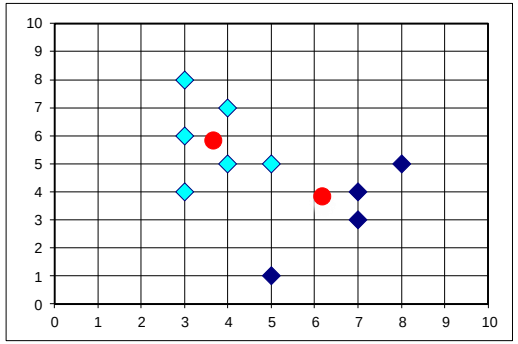
- Compute each centroid as the mean of the objects assigned to it:

$$\boldsymbol{\mu}_k \leftarrow \frac{1}{|S_k|} \sum_{\mathbf{x}_n \in S_k} \mathbf{x}_n$$

- Repeat previous 2 steps until error stops decreasing.

# Example

- $x_1 = \frac{3\times3+2\times4+5}{6} = 3.67$
- $y_1 = \frac{8+7+6+5+4+1}{6} = 5.17$
- $x_2 = \frac{5+2\times7+8}{4} = 6.75$
- $y_2 = \frac{3+4+2\times5}{4} = 4.25$

- $x_1 = \frac{3\times3+2\times4+5}{6} = 3.67$
- $y_1 = \frac{8+7+6+2\times5+4}{6} = 5.83$
- $x_2 = \frac{5+2\times7+8}{4} = 6.75$
- $y_2 = \frac{1+3+4+5}{4} = 3.25$

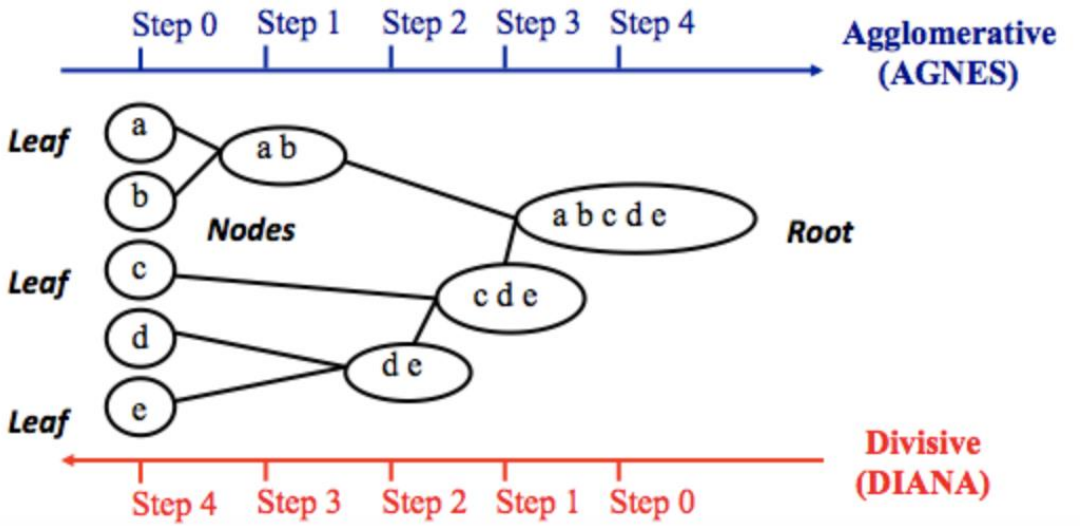
# Pros and Cons of k-means clustering

- Advantage: Scalability
  - K-means is fast. In K-means only the distance between points and cluster center is calculated (as opposed to pairwise distance between all points in hierarchical clustering). It has a linear complexity $O(n)$

- Disadvantage: Flexibility
  - The number of clusters must be predetermined which is not always a trivial task.
  - Using k-means with the mixed variables (continuous and categorical) isn't trivial.
  - K-means is not reproducible. It starts with random choice of cluster centers and it may yield different clustering results on different runs of the algorithm.

# Hierarchical Clustering Algorithms

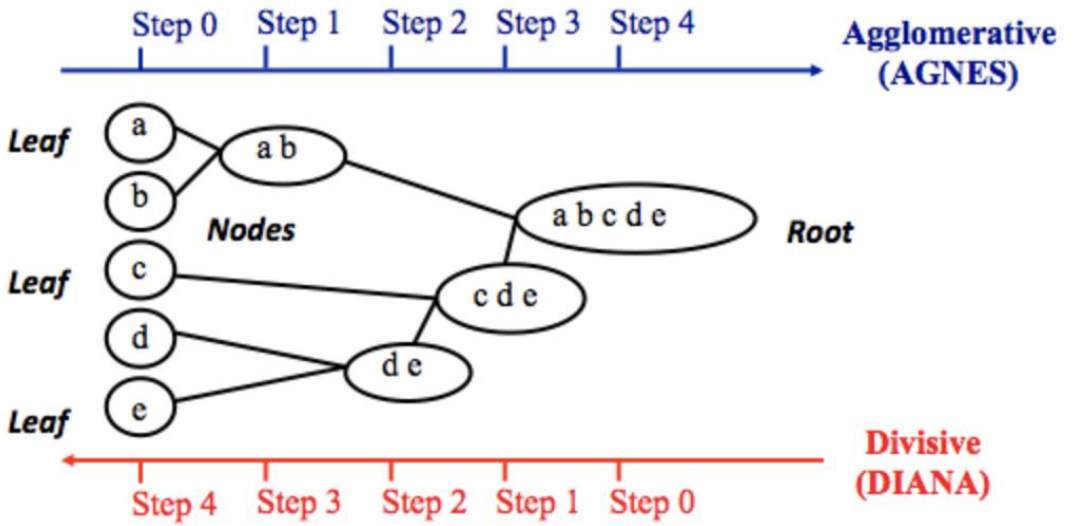
- **Agglomerative clustering:** It is Also known as AGNES (Agglomerative Nesting) and works in a bottom-up manner. At each step of the algorithm, the two clusters that are the most similar are combined into a new bigger cluster.
- **Divisive hierarchical clustering:** It's also known as DIANA (Divise Analysis) and it works in a top-down manner. At each step of iteration, the most heterogeneous cluster is divided into two.
- See a demo for AGNES here.
- AGNES is better in forming small clusters, DIANA is better in forming larger clusters.




From: UC Analytics R Guide

# AGNES - DIANA

| | AGNES | DIANA |
|---|---|---|
| After Step 1 | ab, c, d, e<br>(4 clusters) | cde, ab<br>(2 clusters) |
| After Step 2 | ab, de, c<br>(3 clusters) | de, c, ab<br>(3 clusters) |
| After Step 3 | ab, cde<br>(2 clusters) | d,e,c,ab<br>(4 clusters) |
| After Step 4 | abcde<br>(1 cluster) | a,b,c,d,e<br>(5 clusters) |



# Linkage Methods

Depending on how we measure the dissimilarity between two clusters of observations the formed clusters will have different structures:

**Maximum or complete linkage clustering:** It computes all pairwise dissimilarities between the elements in cluster 1 and the elements in cluster 2, and considers the largest value of these dissimilarities as the distance between the two clusters. It tends to produce more **compact clusters**.

**Minimum or single linkage clustering:** It computes all pairwise dissimilarities between the elements in cluster 1 and the elements in cluster 2, and considers the smallest of these dissimilarities as a linkage criterion. It tends to produce **loose clusters**.

**Mean linkage clustering:** It computes all pairwise dissimilarities between the elements in cluster 1 and the elements in cluster 2, and considers the average of these dissimilarities as the distance between the two clusters.

**Centroid linkage clustering:** It computes the dissimilarity between the centroid for cluster 1 (a mean vector of length p variables) and the centroid for cluster 2.

**Ward's method:** It minimizes the total within-cluster variance. At each step the pair of clusters with minimum between-cluster distance are merged.

# Code Example

```
> library(dplyr)
> library(cluster)
> d = longley %>%
+   dplyr::select(GNP.deflator, Armed.Forces, Population, Employed)
>
> str(d)
'data.frame':   16 obs. of  4 variables:
 $ GNP.deflator: num  83 88.5 88.2 89.5 96.2 ...
 $ Armed.Forces: num  159 146 162 165 310 ...
 $ Population  : num  108 109 110 111 112 ...
 $ Employed    : num  60.3 61.1 60.2 61.2 63.2 ...
```

We use library ”cluster” in R to examine some hierarchical clustering techniques on Longley data set.

# Normalizing the Data

We would like all features to carry the same weight. Therefore, we will normalize them using the Gaussian method.

```
> # Normalizing the data
> scaleObj= preProcess(d, method = c("center","scale"))
> dNorm = predict(scaleObj, d)
>
> str(dNorm)
```

```
'data.frame':   16 obs. of  4 variables:
 $ GNP.deflator: num  -1.731 -1.221 -1.249 -1.129 -0.508 ...
 $ Armed.Forces: num  -1.461 -1.653 -1.424 -1.375 0.707 ...
 $ Population  : num  -1.411 -1.264 -1.1 -0.934 -0.769 ...
 $ Employed    : num  -1.422 -1.194 -1.465 -1.176 -0.597 ...
```

# Finding the Distance

- We compute the dissimilarity values with function ”daisy”. You can also use the “dist” function.
- ”daisy” has the advantage of being able to find the distance of mixed type (continuous and categorical) data set using the gower metric.

```
# Dissimilarity matrix
# Compute all the pairwise dissimilarities (distances) between observations in the data set.
myDist = daisy(dNorm, metric = "euclidean")
myDist
```

```
Dissimilarities :
         1947      1948      1949      1950      1951      1952      1953      1954      1955      1956      1957      1958      1959      1960      1961
1948 0.6084877
1949 0.5764766 0.3922549
1950 0.8115925 0.4423595 0.3580622
1951 2.7002272 2.5855716 2.4401692 2.2546236
1952 3.4357343 3.3448383 3.1855944 3.0064979 0.7619770
1953 3.6096402 3.4713180 3.3418511 3.1202450 0.9610065 0.4775321
1954 3.3716913 3.2088729 3.0508489 2.8336428 0.7953758 0.5786752 0.4869698
1955 3.4410520 3.1854215 3.1074946 2.8224331 1.1992304 1.2279212 0.8662986 0.8014388
1956 3.8054172 3.4735180 3.4552302 3.1356036 1.8400445 1.8825067 1.4808986 1.4746252 0.6970873
1957 4.1169998 3.7455834 3.7321128 3.4076900 2.2131336 2.2241812 1.8244810 1.7831146 1.0686932 0.4467386
1958 4.0320261 3.6293266 3.5865972 3.2737292 2.2726760 2.3458016 2.0162436 1.8286810 1.2609706 0.8889399 0.6100274
1959 4.4938475 4.0687755 4.0713111 3.7390367 2.8206577 2.8624697 2.4771375 2.3823657 1.7105074 1.1120593 0.6869557 0.6754056
1960 4.8542676 4.4214840 4.4327499 4.0961314 3.2253144 3.2411802 2.8400175 2.7630917 2.0916312 1.4765046 1.0554866 1.0614421 0.4180766
1961 5.1206877 4.6928153 4.6838651 4.3526635 3.4654825 3.4406045 3.0413838 2.9459319 2.3292640 1.7663594 1.3440130 1.2564347 0.7323206 0.3977260
1962 5.6521299 5.2462369 5.2306581 4.8986508 3.8585939 3.7368072 3.3123729 3.2751249 2.6855715 2.1331425 1.7323687 1.7556727 1.2402662 0.8960593 0.6082645
```

- Note that only the lower triangle of the matrix is displayed, since it doesn't matter which two observation we consider.
- Also, note that myDist is a dissimilarity object and all of the clustering functions will recognize this. However, if you need to use it with anything other than the clustering functions, you'll need to use as.matrix to convert it to a regular matrix.

```
> nrow(myDist)
NULL
> nrow(as.matrix(myDist))
[1] 16
```

# AGNES Clustering

- The agnes function can take either the data matrix or dissimilarity matrix, depending on the value of the diss argument.

```
# Clustering with agnes
agnesClust =  agnes(myDist, diss = T, method = "complete")
```

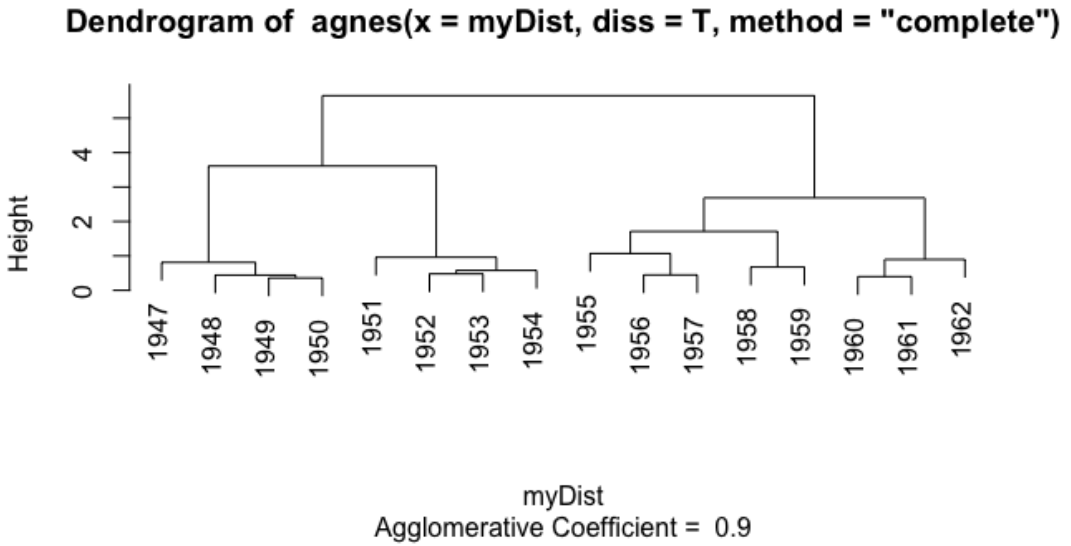
- Agglomerative coefficient measures the amount of clustering structure found (values closer to 1 suggest strong clustering structure).

```
> agnesClust$ac
[1] 0.8952904
```

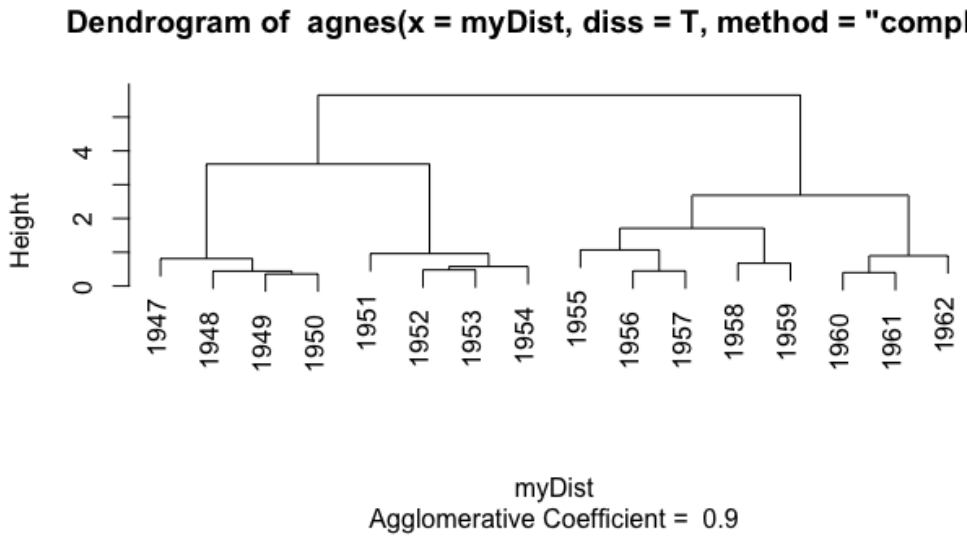
# Dendrogram

- The main graphical tool for looking at a hierarchical cluster solution is known as a dendogram.
- This is a tree-like display that lists the objects which are clustered along the x-axis, and the distance at which the cluster was formed along the y-axis.
- Since the y-axis represents how close together observations were when they were merged into clusters, clusters whose branches are close together probably aren't very reliable. But if there's a big difference along the y-axis between the last merged cluster and the currently merged one, that indicates that the clusters formed are probably doing a better job in showing the data structure.

```
pltree(agnesClust,main = "Dendrogram of complete linkage", cex = 0.5 )
```



# Dendrogram




- Looking at this dendogram, there are clearly two very distinct groups.
- The left hand group seems to consist of two more distinct cluster, while most of the observations in the right hand group are clustering together at about the same height.
- For this data set, it looks like either two or three groups might be an interesting place to start investigating.

# Comparing Different Linkage Methods

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

myAc = sapply(m, function(myMethod){
  agnes(myDist, method = myMethod)$ac
})

myAc
```

```
> myAc
  average    single  complete      ward
0.8480666 0.7806937 0.8952904 0.9290746
```

Based on the above values Ward's method provides us with the strongest clustering structure of the four methods.

# Cluster Membership

We can use “cutree” to see how many elements are in each of the clusters.

```
> (cutree(agnesClust,3))
 [1] 1 1 1 1 2 2 2 2 3 3 3 3 3 3 3 3
```

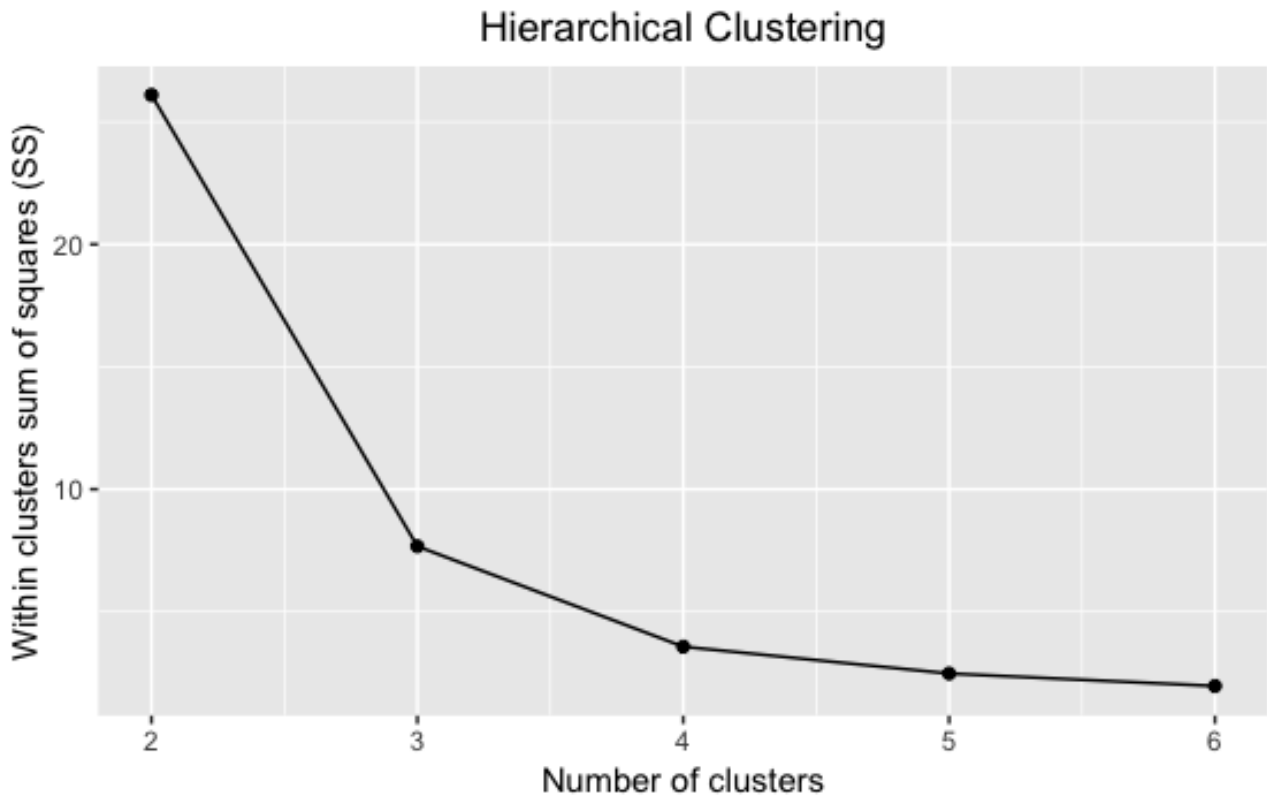
```
sapply(2:5, function(i){
  table(cutree(agnesClust,i))
})
```

```
> table(cutree(agnesClust,3))

1 2 3
4 4 8
```

```
[[1]]

1 2
8 8

[[2]]

1 2 3
4 4 8

[[3]]

1 2 3 4
4 4 5 3

[[4]]

1 2 3 4 5
4 4 3 2 3
```

# Elbow Method

- One method to choose the number of clusters is to pick a k so that adding another cluster doesn't give much better modeling of the data.
- Recall that the total within-cluster sum of square (wss) measures the compactness of the clustering and therefore we want it to be as small as possible.
- If we plot wss vs. k, the first clusters will have a large wss (since there are lots of points in them), but at some point the marginal gain will drop, giving an angle in the graph.
- The number of clusters is chosen at this point, aka "elbow criterion".
- In this plot, k = 3 or k = 4 seem like a reasonable choice.
- fviz_nbclust can also be used for elbow plots.



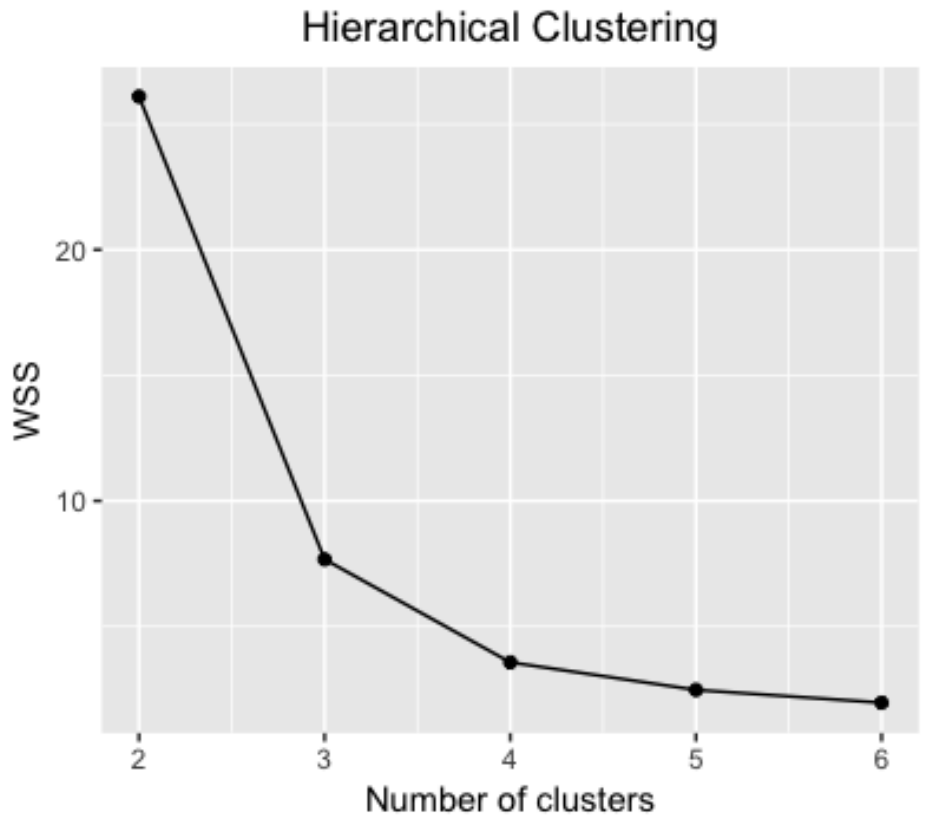
```
> cluster.stats(d = myDist, clustering = cutree(agnesClust,k=3))["within.cluster.ss"]
$within.cluster.ss
[1] 7.67187
```

# Elbow Plot

```
wss = sapply(2:6, function(i) {
  unlist(cluster.stats(d = myDist, clustering = cutree(agnesClust,k=i)))["within.cluster.ss"] }
  )
```

```
> (wssDF = data.frame(k = c(2:6), within.cluster.ss = wss))
  k within.cluster.ss
1 2         26.097297
2 3          7.671870
3 4          3.571774
4 5          2.474945
5 6          1.965529
```

# Elbow Plot




```
wssDF %>%
ggplot(aes(x=k, y=within.cluster.ss)) +
  geom_point()+
  geom_line()+
  ggtitle("Hierarchical Clustering") +
  labs(x = "Number of clusters", y = "WSS") +
  theme(plot.title = element_text(hjust = 0.5))
```

# Silhouette Score

- Silhouette Score is a measure that indicates how well each observation fits into the cluster that it's been assigned to.
- It’s calculated by comparing how close the object is to other objects in its own cluster with how close it is to objects in other clusters.

| Average Silhouette Score | Cluster Quality |
|---|---|
| $0.7 < x < 1$ | Strong Structure |
| $0.5 < x < 0.7$ | Reasonable Structure |
| $0.25 < x < 0.5$ | Weak Structure |
| $x < 0.25$ | No Substantial Structure |

# Average Silhouette Plot

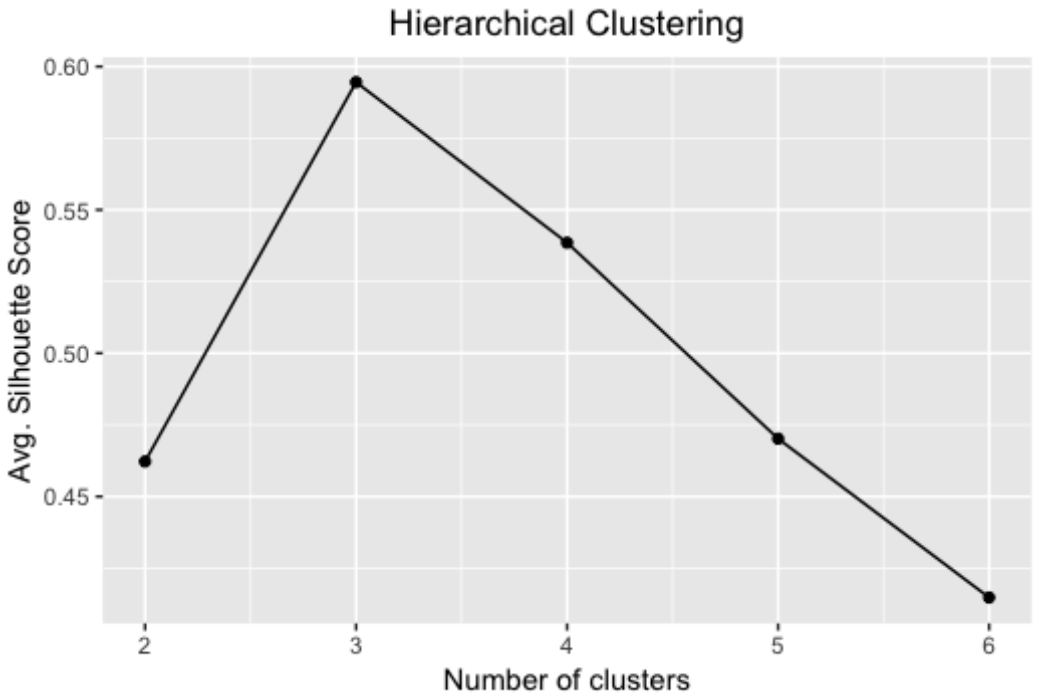
```
> silhouette(cutree(agnesClust, k = 3), myDist)
```

```
      cluster neighbor  sil_width
 [1,]       1        2  0.7970560
 [2,]       1        2  0.8474192
 [3,]       1        2  0.8528050
 [4,]       1        2  0.8083504
 [5,]       2        1  0.6635658
 [6,]       2        3  0.7686951
 [7,]       2        3  0.7124846
 [8,]       2        3  0.7123789
 [9,]       3        2 -0.3949492
[10,]       3        2  0.2708907
[11,]       3        2  0.5067488
[12,]       3        2  0.4930176
[13,]       3        2  0.6435917
[14,]       3        2  0.6497967
[15,]       3        2  0.6261928
[16,]       3        2  0.5547416
attr(,"Ordered")
[1] FALSE
attr(,"call")
silhouette.default(x = cutree(agnesClust, k = 3), dist = myDist)
attr(,"class")
[1] "silhouette"
```

```
silScore = sapply(2:6, function(i){
  mean(silhouette(cutree(agnesClust, k = i), myDist)[,'sil_width'])
})
```

```
> (silScoreDF = data.frame(k = 2:6, silScore))
  k  silScore
1 2 0.4622284
2 3 0.5945491
3 4 0.5385505
4 5 0.4701848
5 6 0.4147656
```

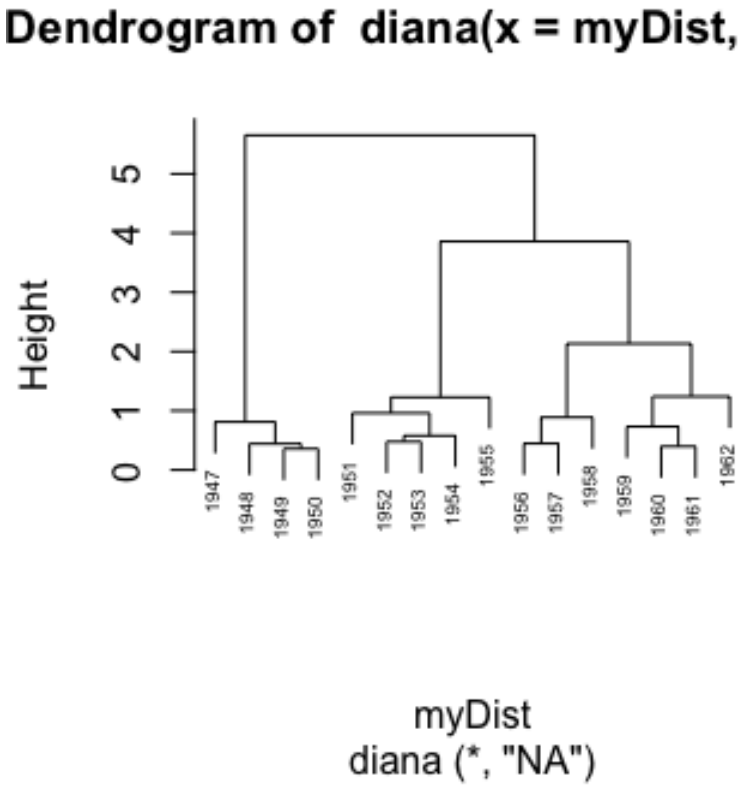
# Average Silhouette Plot

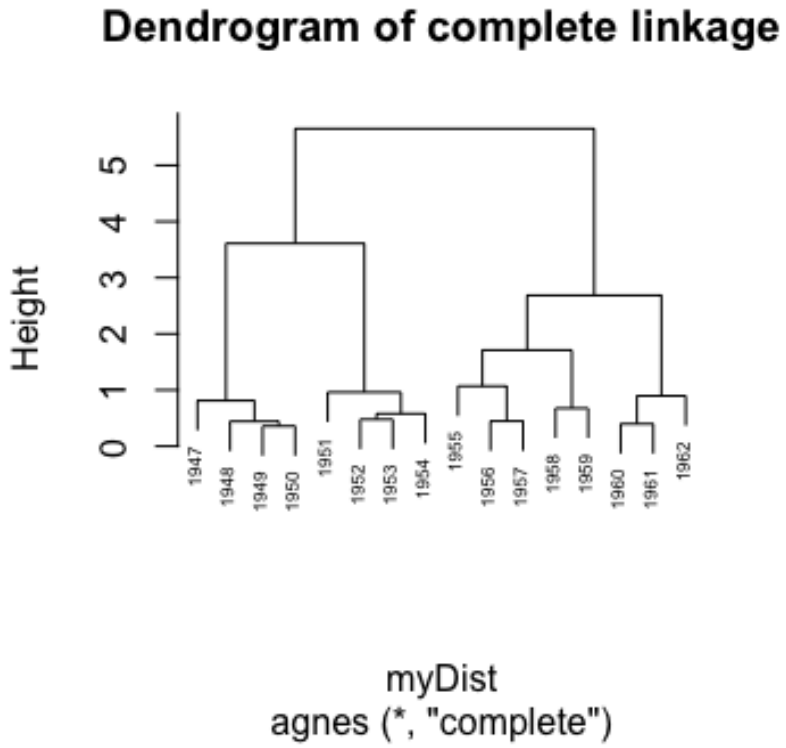



```
silScoreDF %>%
  ggplot(aes(x = k, y = silScore))+
  geom_point()+
  geom_line()+
  ggtitle("Hierarchical Clustering") +
  labs(x = "Number of clusters", y = "Avg. Silhouette Score") +
  theme(plot.title = element_text(hjust = 0.5))
```

# DIANA Clustering

```
> dianaClust = diana(myDist, diss = T)
> pltree(dianaClust, cex = 0.5 )
```

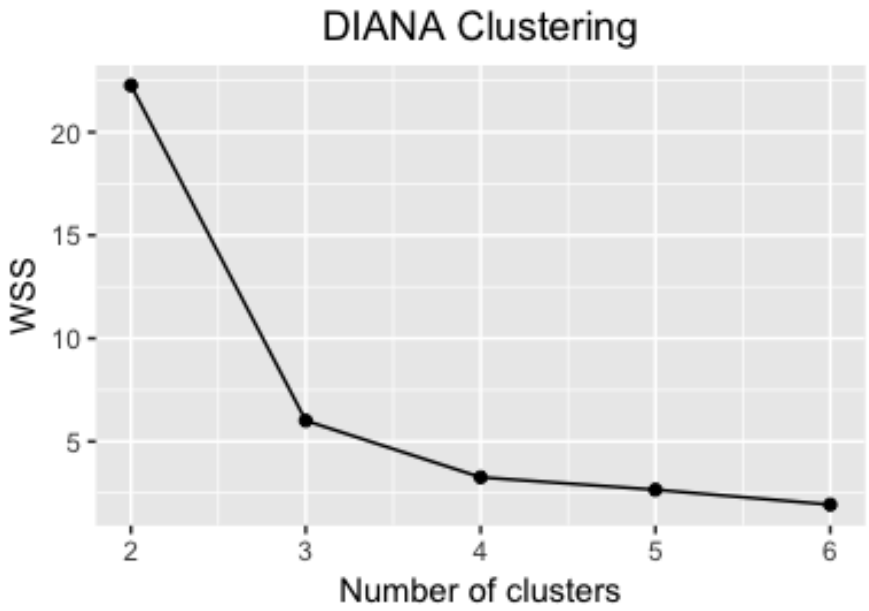


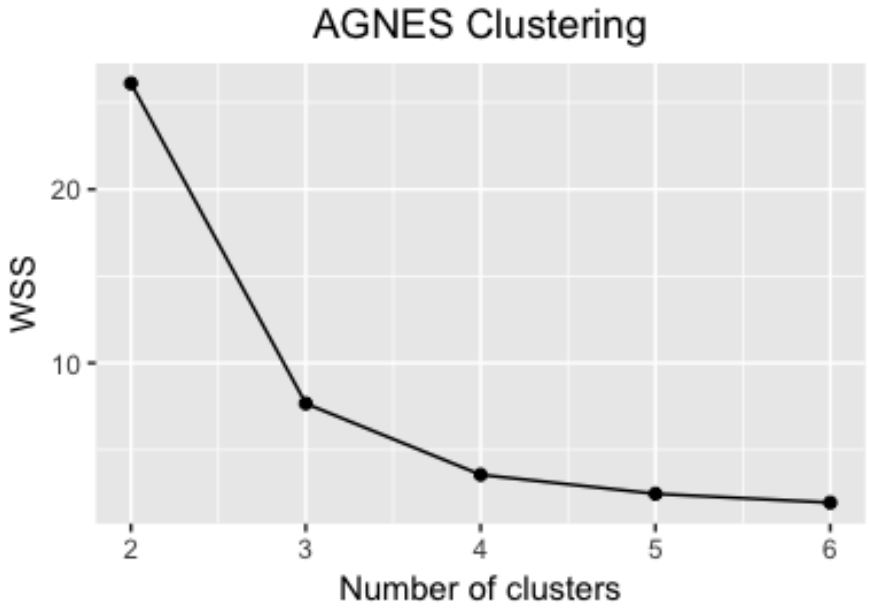



# Elbow Plot

Comparing the two plots, we note for k = 3, DIANA WSS is smaller for AGNES WSS.

```
wss = sapply(2:6, function(i) {
  unlist(cluster.stats(d = myDist, clustering = cutree(dianaClust,k=i)))["within.cluster.ss"] }
)

(wssDF = data.frame(k = c(2:6), within.cluster.ss = wss))
wssDF %>%
  ggplot(aes(x=k, y=within.cluster.ss)) +
  geom_point()+
  geom_line()+
  ggtitle("DIANA Clustering") +
  labs(x = "Number of clusters", y = "WSS") +
  theme(plot.title = element_text(hjust = 0.5))
```



# Average Silhouette Plot

At k = 3, DIANA has a much better Silhouette score than AGNES.

```
silScore = sapply(2:6, function(i){
  mean(silhouette(cutree(dianaClust, k = i), myDist)[,'sil_width'])
})

(silScoreDF = data.frame(k = 2:6, silScore))

silScoreDF %>%
  ggplot(aes(x = k, y = silScore))+
  geom_point()+
  geom_line()+
  ggtitle("DIANA Clustering") +
  labs(x = "Number of clusters", y = "Avg. Silhouette Score") +
  theme(plot.title = element_text(hjust = 0.5))
```