Kmeans ++, Heirarchical Clustering]

→ Kmeans ++

→ Heirarchical clustering

Recap

↳ Amazon customer segmentation

n_clicks	n_visits	amount_spent	amount_discount	days_since_registration
130	65	213.905831	31.600751	233
543	46	639.223004	5.689175	228
520	102	1157.402763	844.321606	247
702	83	1195.903634	850.041757	148
221	84	180.754616	64.283300	243

→ Clustering :

↳ Points belong to different populations

→ A good clustering algorithm will be able to detect these.

Kmeans ++

There is an issue with the lloyd's algo.

→ Because, centroids are randomly initialized we may get different results some times.

→ animation

Solution

→ choose centers smartly.

idea:

→ 1st centroid initialised randomly

→ then choose 2nd center as the farthest point from first

→ n^{th} centroid is the point which is farthest from it's nearest existing centroid

→ animation

What about outliers?

→ Remove oultiers for learning centers.

↳ outlier removal depends on human judgement

→ Instead of choosing outliers we could choose furthest point as centroid with prob proportional to distance

```
for iteration in range(10):
    c1 = np.random.choice(pts)
    probs = []
    for i in range(len(pts)):
        probs.append(dist(c1, pts[i]))
    probs = np.array(probs) / sum(probs)

    c2 = np.random.choice(pts, probs)
```

Limitations of Kmeans

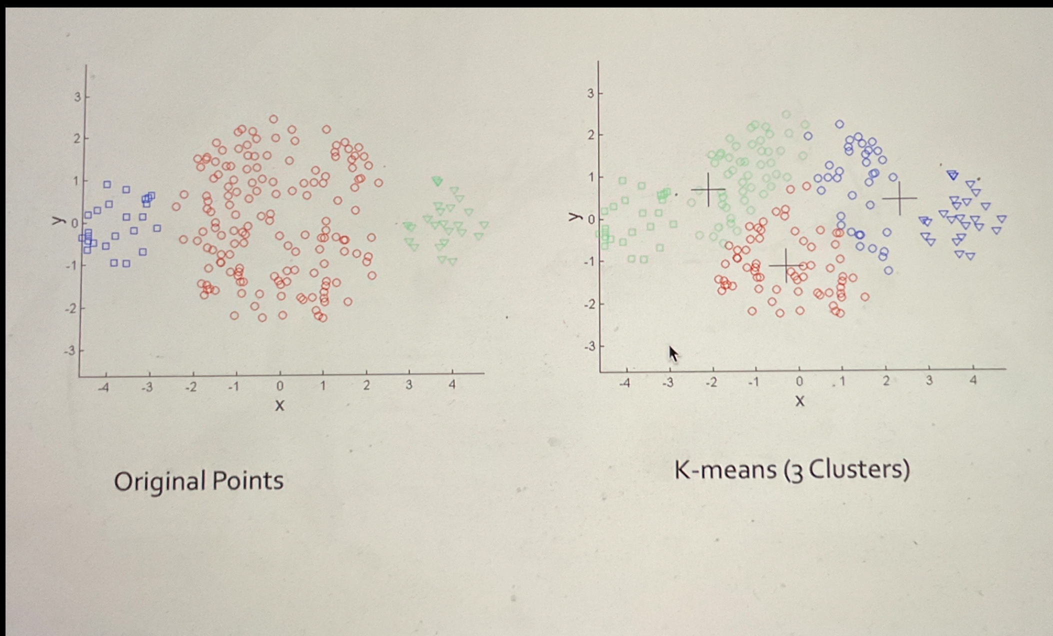
→ Need to decide "K"

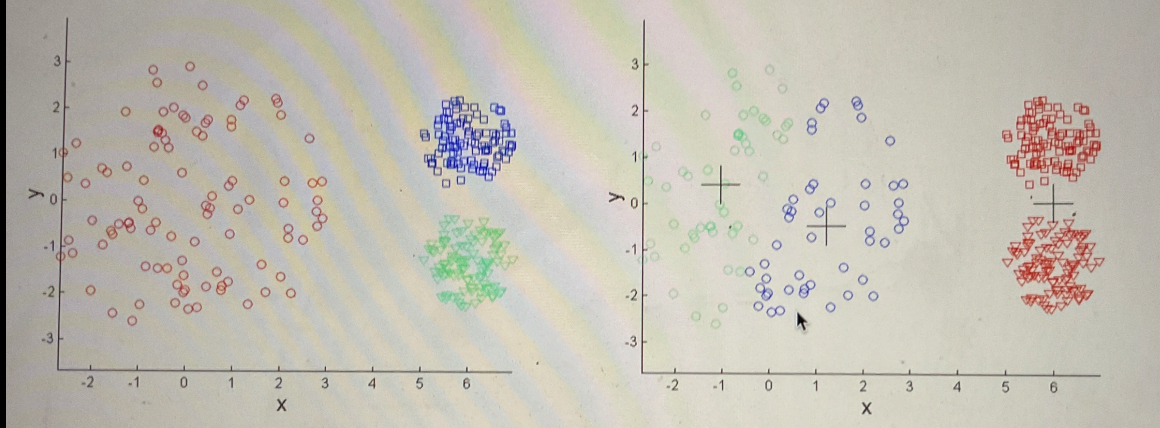
→ May not be best for ______ clusters.

→ Diff size

→ diff density

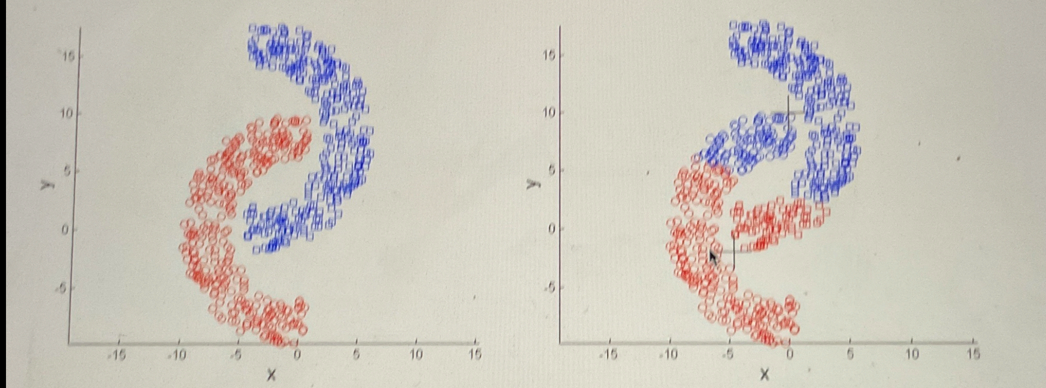
→ non globular (hyper-spheres)





Original Points

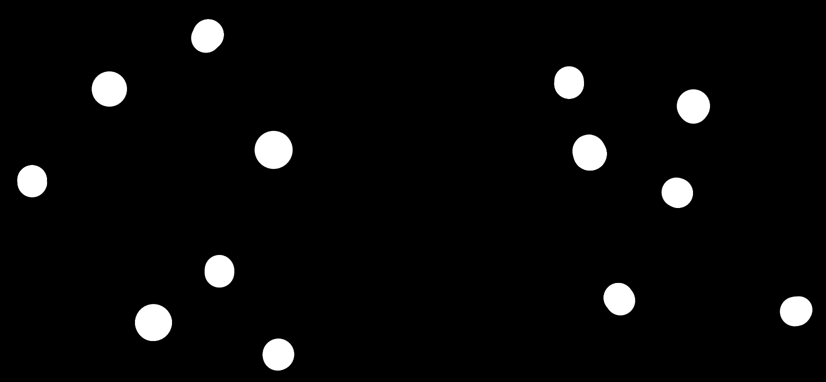
K-means (3 Clusters)



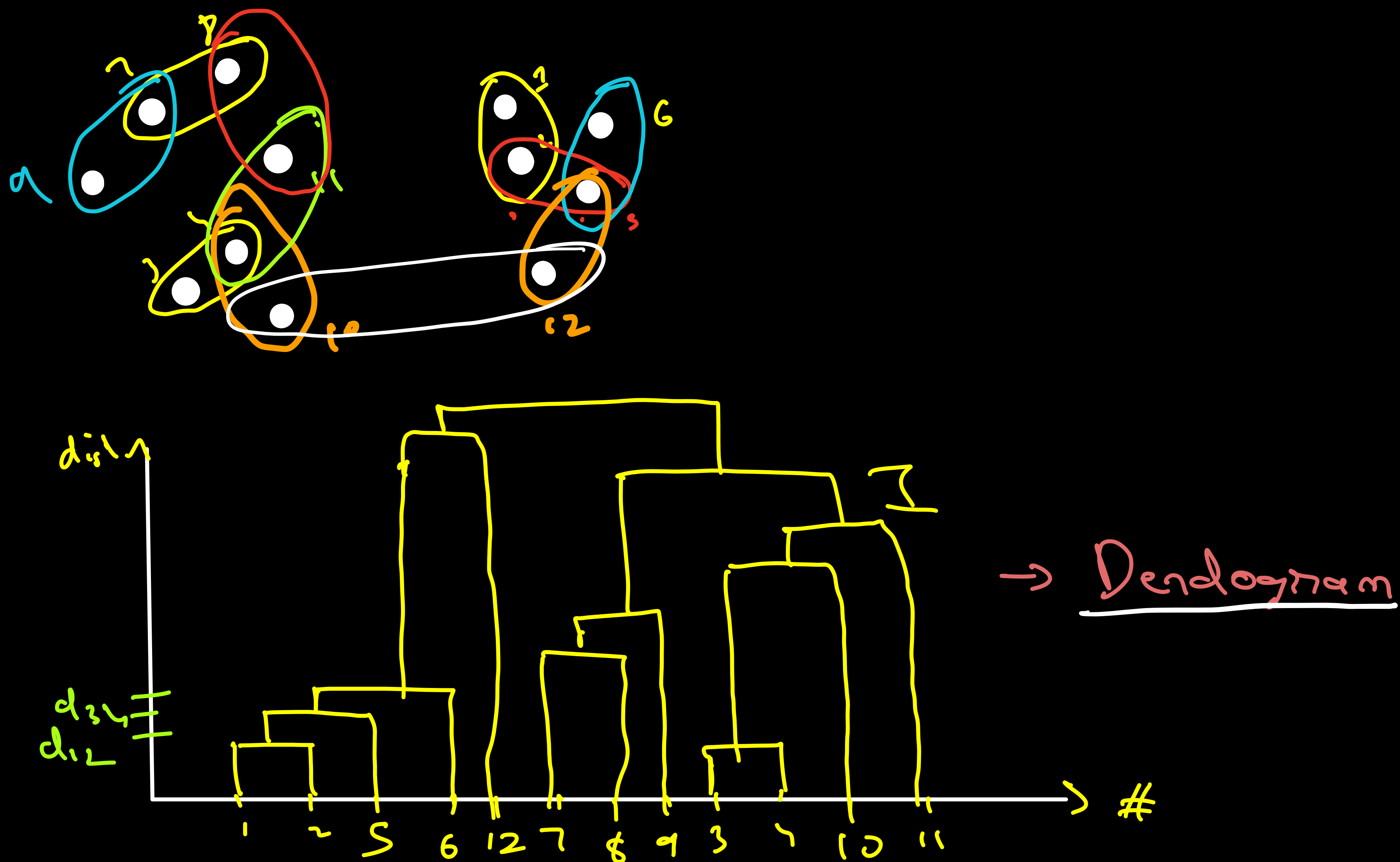
Original Points

K-means (2 Clusters)

Heirarchical Clustering

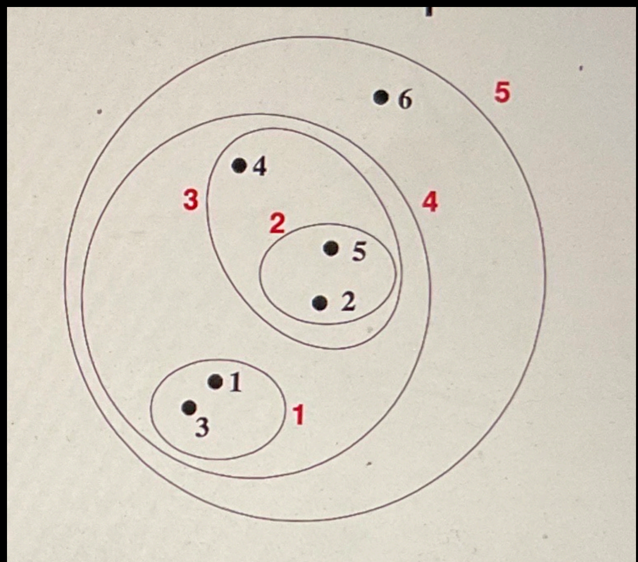


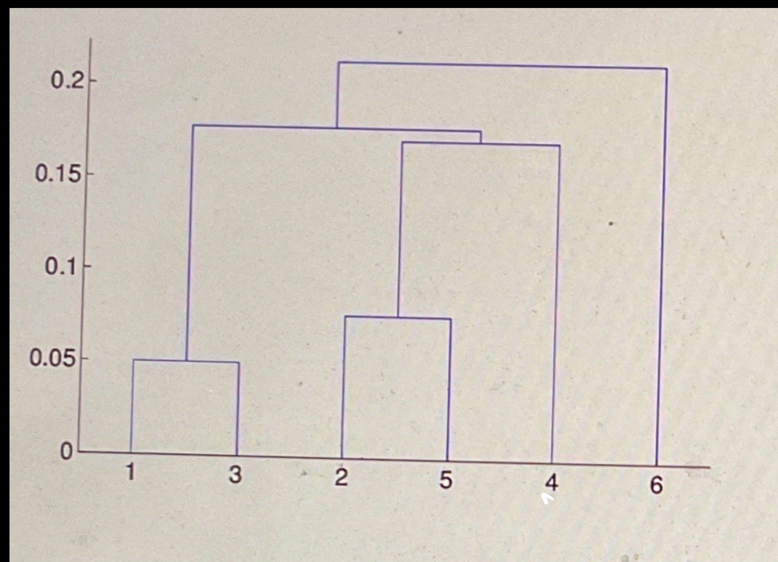
- compute some form of distance b/w all points.
- combine 2 pts with min distance
- treat the sub-clusters as pts and repeat till only one cluster remains


dist
d_{34}
d_{12}
1 2 5 6 12 7 8 9 3 4 10 11
#
→ Dendogram

(another name)

→ Agglomerative clustering:





points

dendogram

distance b/w clusters:

→ min

→ animation

→ max

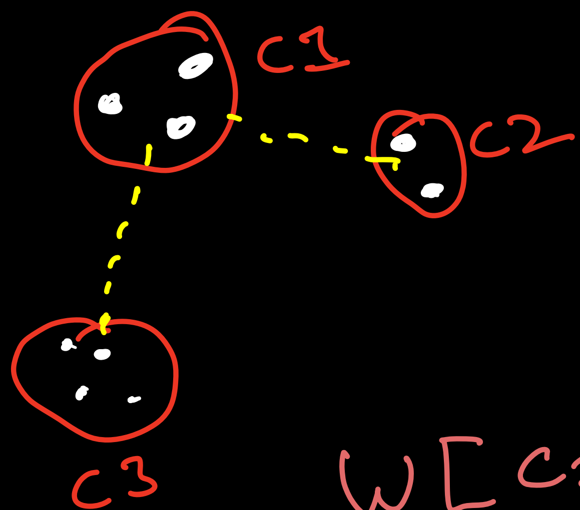
→ avg

→ any other func^n, eg: squared, cosine etc..

→ wards method

Pros and cons in post read.

Ward's distance / Ward's linkage



Using avg dist

C1 + C2 , C3

Ward's distance

$W[C1, C2] =$

$$\boxed{WCSS(C1+C2) - WCSS(C1) + WCSS(C2)}$$

How much will ESS / WCSS increase

ESS → error; WCSS → within cluster

when I merge 2 clusters.

→ We will merge clusters where there is min increase in ESS / WCSS

→ very effective metric in practice.

Pros:

→ no need to pre-decide 'K'

→ heirarchy may actually be present in real world

Cons:

→ Sensitive to choice of linkage fxn.

→ Offline only. ← Can only be used for analysis

Term:

Proximity matrix:

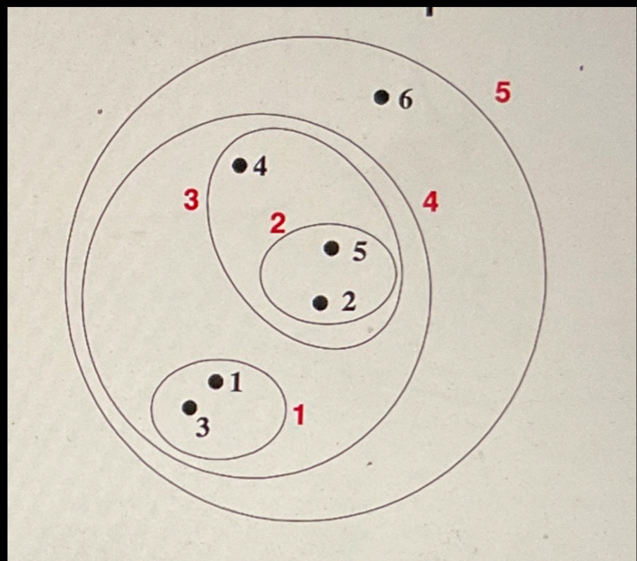
distance (c_i, c_j)

$$
\begin{array}{c|cccc|}
 & c_1 & c_2 & c_3 & \cdots\ c_n \\
\hline
c_1 & 0 & 0.25 & 1.3 & \cdots \\
c_2 & 0.31 & 0 & \vdots & \vdots \\
c_3 & \vdots & \vdots & 0 & \vdots \\
\vdots & \vdots & & & \\
c_n & \cdots & \cdots & \cdots & 0 \\
\hline
\end{array}
$$

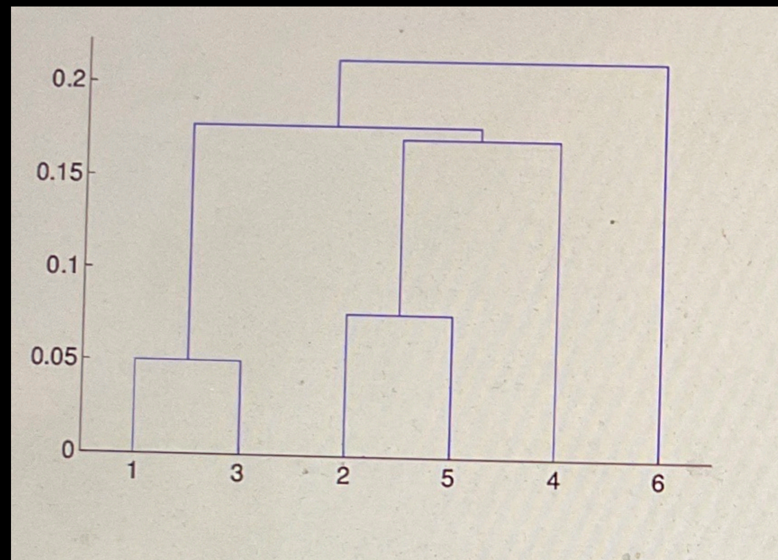
→ code

Heirarchical Clustering

→ Agglomerative clustering:



points



dendogram

distance b/w clusters:

→ min

→ animation

→ max
→ avg

} pros and cons in post read.

→ any other func^n, eg: squared, cosine etc..

Pros:

→ no need to pre-decide 'K'

→ heirarchy may actually be present in real world

Term:

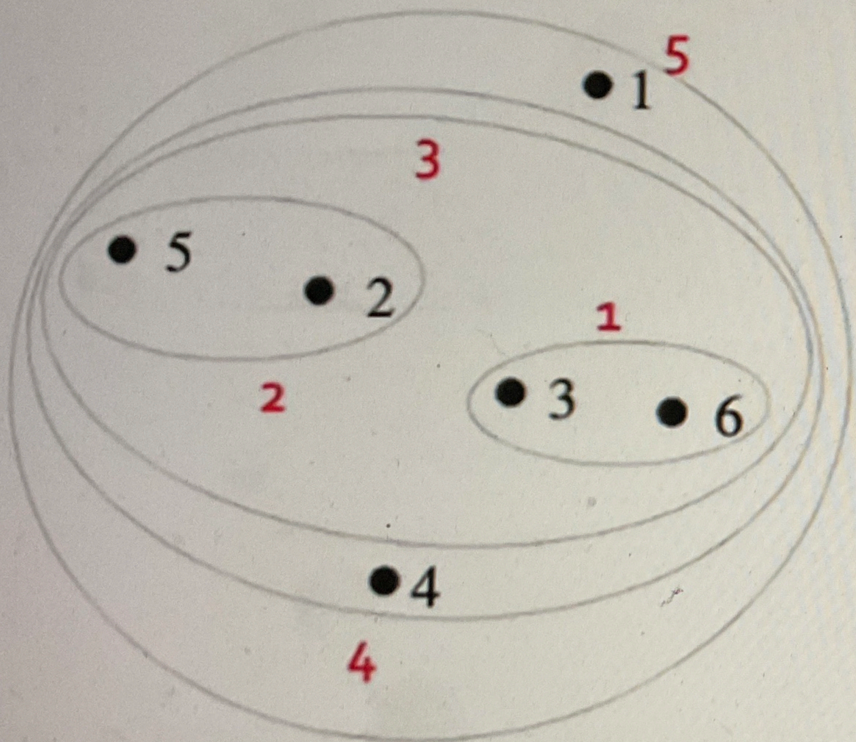
Proximity matrix:

	c_1	c_2	c_3	..	c_n
c_1	0	0.25	1.3		..
c_2	0.31	0	.		.
c_3	.	.	0		.
.	.				
c_n	.	.	-	-	0

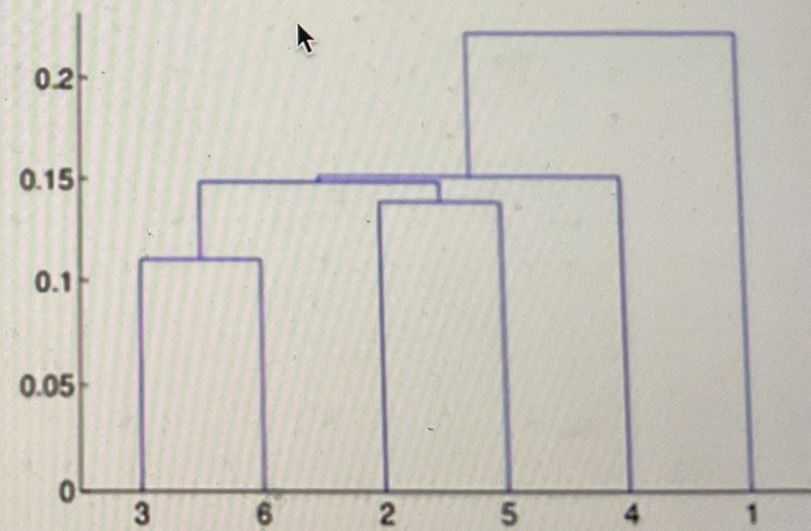
distance (c_i, c_j)

→ code

	1	2	3	4	5	6
1	0	.24	.22	.37	.34	.23
2	.24	0	.15	.20	.14	.25
3	.22	.15	0	.15	.28	.11
4	.37	.20	.15	0	.29	.22
5	.34	.14	.28	.29	0	.39
6	.23	.25	.11	.22	.39	0



Nested Clusters



Dendrogram