

# 2020-1 Data Analysis with Applications

## Lecture 18. Clustering

---

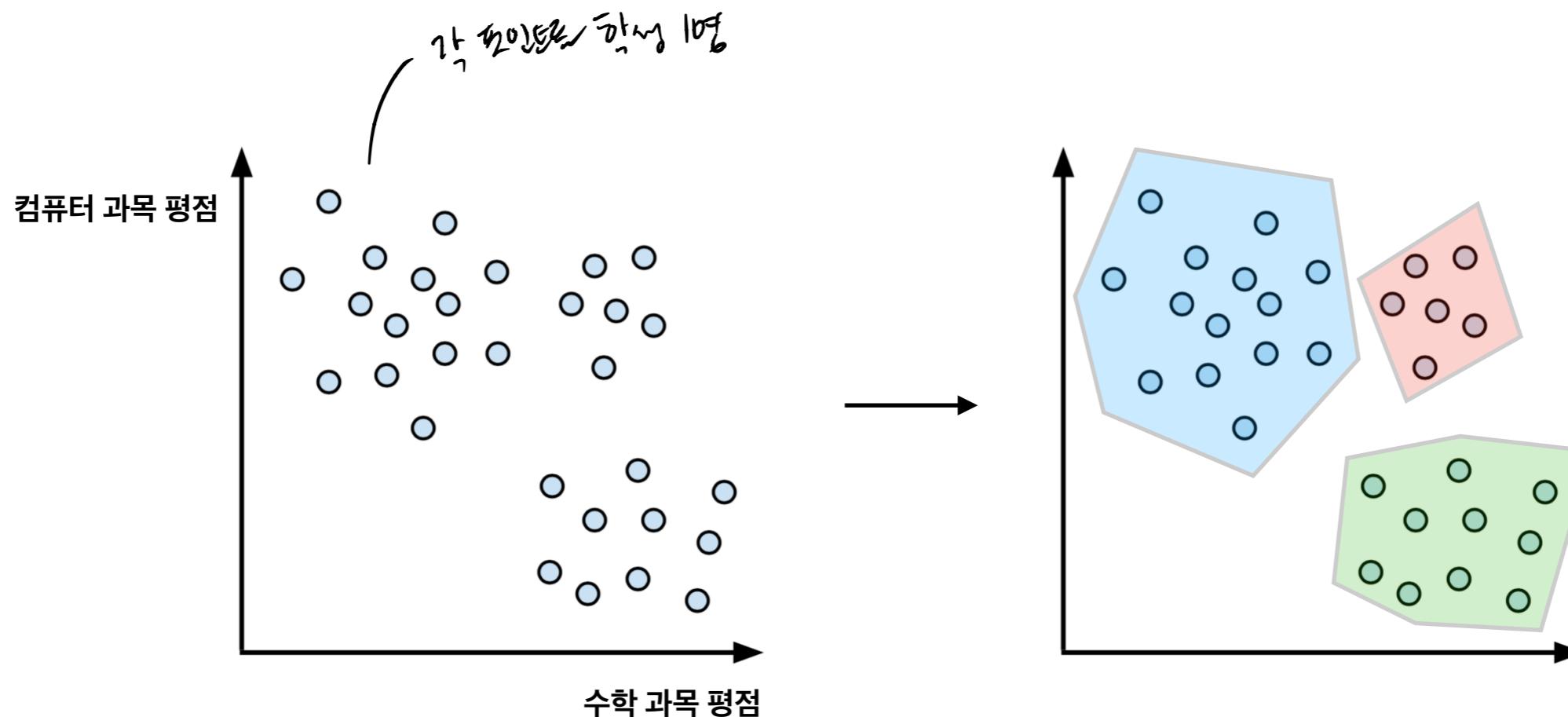
Department of Industrial and Information Systems Engineering,  
Soongsil University

# Introduction to Clustering

---

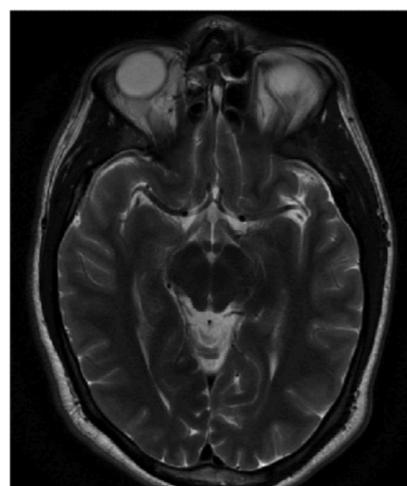
# Clustering

- Clustering은 주어진 데이터셋으로부터 cluster (or subgroup)를 찾는 기법을 말한다.
- 데이터들을 서로 겹치지 않는 subgroup들로 partition하며, 각 subgroup 내의 데이터들은 서로 최대한 유사하게 (similar), 서로 다른 subgroup에 속한 데이터들은 서로 최대한 다르게 (dissimilar) 만드는 partition을 찾는다.

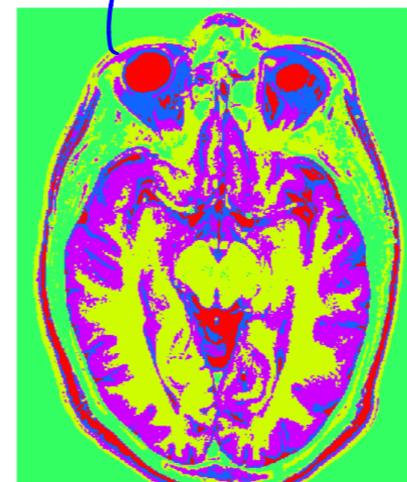
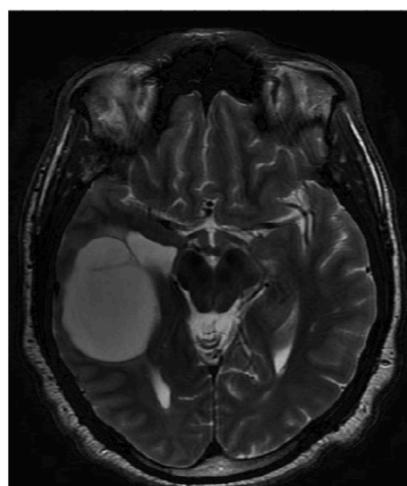


# Clustering Applications

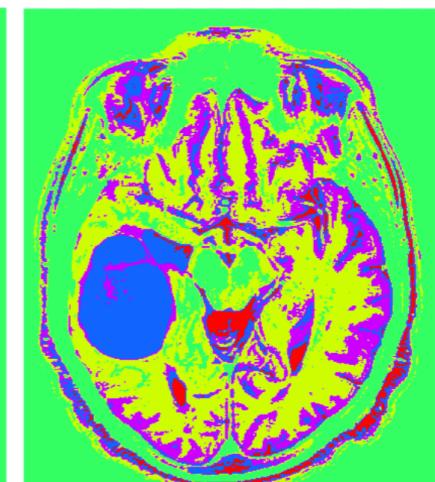
- Market segmentation
  - 고객의 구매패턴이나 거주지, 직업, 수입 등의 정보를 활용하여 고객들을 여러 그룹으로 나눈 후 각 그룹의 특성에 맞는 광고 및 마케팅 전략을 수립할 수 있다.  
→ 광고를 더 넓힐 수 있다.
- Image segmentation
  - 픽셀 단위의 이미지 데이터를 여러 segment로 분할하여 원래 이미지를 단순화시킴으로써 더 의미있는 정보를 추출하거나 분석이 용이하게 만들 수 있다.
  - 의료 영상 이미지에서의 cancer detection, 감시카메라 이미지에서 비일상적인 패턴 포착 등에 활용된다.



Original MRI Images



의료 영상 분석



Segmented Images

# Clustering Algorithms

- 대표적인 clustering 알고리즘으로는 *k-means clustering*과 *hierarchical clustering*이 있다.  
*k가 정해져고 cluster  $\Rightarrow$  data 사이의 거리도 빠르게 계산.*
- *k-means clustering*은 cluster의 수  $k$ 가 미리 주었을 때  $k$ 개의 cluster를 찾는 알고리즘으로 계산량과 계산시간 측면에서 매우 효율적이기 때문에 자주 활용된다.
- *Hierarchical clustering*은 cluster의 수를 미리 정하지 않으며, clustering의 결과를 tree의 형태로 시각화할 수 있다는 장점이 있다.

# *k*-Means Clustering

---

# $k$ -Means Clustering

- $k$ -means clustering에서는 cluster의 수  $K$ 가 주어졌을 때 전체 데이터의 집합을  $K$  개의 부분집합  $C_1, C_2, \dots, C_K$ 로 partition한다.
- $k$ -means clustering은 각 cluster에 속한 데이터 사이의 변동 (within-cluster variation)을 최소화하는 것이 목적이다. Cluster  $C_k$ 의 within-cluster variation을  $W(C_k)$ 로 정의한다면  $k$ -means clustering은 다음 목적함수를 최소화하는 partition  $C_1, C_2, \dots, C_K$ 를 찾는 것과 같다.

변동

$$\min_{C_1, \dots, C_K} \{ \sum_{k=1}^K W(C_k) \}$$

- 일반적으로 within-cluster variation은 cluster 내의 모든 데이터 사이의 squared Euclidean distance의 합으로 계산한다. 즉,  $W(C_k)$ 는 다음 식과 같다.

함이 작을수록 partition이 품질적으로 높다.

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

- $|C_k|$ 는 cluster  $C_k$ 에 속한 데이터의 수,  $x_{ij}$ 는  $i$ 번째 데이터의  $j$ 번째 feature의 값,  $p$ 는 feature의 수를 의미한다.

# $k$ -Means Clustering

- 앞의 두 식을 결합하면,  $k$ -means clustering은 다음 문제를 푸는 것과 같다.

$$\cancel{\min}_{C_1, \dots, C_k} \left\{ \sum_{k=1}^K \frac{1}{|C_k|} \sum_{i, i' \in C_k} \sum_{j=1}^p (x_{ij} - x_{i'j})^2 \right\}$$

- 데이터의 수가 많을 때 이 문제의 최적해를 구하는 것은 이론적으로 거의 불가능하다.  $n$ 개의 데이터로 만들 수 있는  $K$ 개의 partition의 수는  $K^n$ 에 가까울 정도로 매우 크다. 예를 들어  $n = 50, K = 3$ 일 때 가능한 partition의 수는 약  $4.08 \times 10^{18}$ 개이다.
- 따라서 빠른 시간 내에 좋은 해를 찾기 위한 heuristic algorithm을 사용한다.

최적해를 찾기 위해 고마끔은 해를 찾는 algorithm

ex) selectim

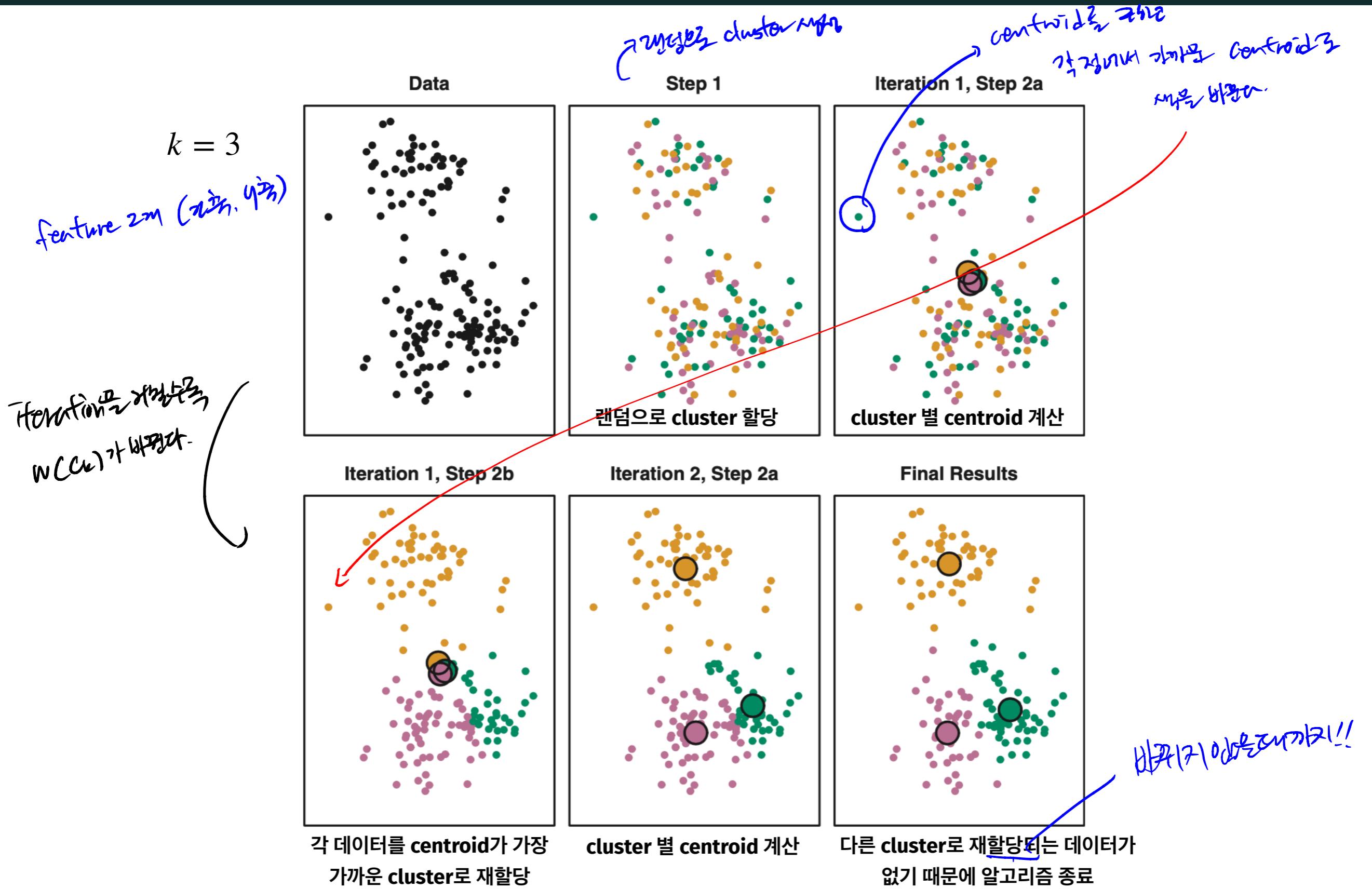
# *k*-Means Clustering

## ***k*-Means Clustering**

1. 각 데이터에 1부터  $K$ 까지의 cluster number를 랜덤으로 할당한다.
2. 다음을 cluster의 변동이 없을 때까지 반복한다.
  - (a) 각 cluster의 무게중심(centroid)을 계산한다.
  - (b) 각 데이터를 데이터로부터 가장 가까운 centroid를 가지는 cluster에 할당한다. 이때 centroid까지의 거리는 Euclidean distance를 사용한다.

- Cluster의 centroid는 cluster에 속한 모든 데이터의 평균으로 계산한다.
  - 예)  $C_1 = \{(3,6,4), (3,2,2), (0,4,3)\}$  일때,  $C_1$ 의 centroid = (2,4,3).
- Step 2를 반복할수록 within-cluster variation은 점점 감소하며, within-cluster variation이 더 이상 감소하지 않는 순간 알고리즘이 종료된다.

# $k$ -Means Clustering



# $k$ -Means Clustering

\*  
\* 초기 랜덤으로 cluster로 할당하는 경우에 따라 성능이 많이 차이날 수 있다.

- $k$ -means clustering의 성능은 Step 1에서 랜덤으로 만들어지는 첫 cluster의 형태에 따라서 크게 달라질 수 있다.
- 따라서  $k$ -means를 적용할 때에는 알고리즘을 여러 번 실행한 후 목적함수 값이 가장 작은 solution을 선택하는 것이 합리적이다.



- 앞의 예에 대해서  $k$ -means를 6번 실행한 결과이다. 서로 다른 랜덤 cluster로부터 알고리즘이 진행되기 때문에 최종적으로 구해진 cluster가 가지는 목적함수 값의 차이가 큼을 확인할 수 있다.

# Hierarchical Clustering

---

# Hierarchical Clustering

- Hierarchical clustering은 다음과 같은 매우 간단한 알고리즘에 의해 수행된다.

↳ cluster를 계속 바꾸는 것

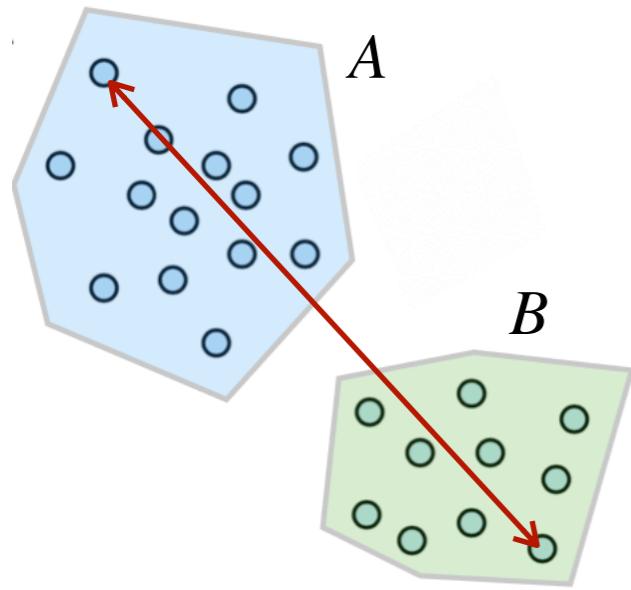
## Hierarchical Clustering

1. 주어진  $n$ 개의 데이터에 대해서, 각 데이터 하나로 cluster 하나를 만든다. 즉,  $n$ 개의 cluster가 생성된다.
  2. 다음을 전체 cluster의 수가 하나가 될때까지 반복한다.
    - (a) 모든 cluster 사이의 dissimilarity 값을 계산하고, dissimilarity 값이 가장 작은 두 cluster를 하나로 합친다(merge).
- 즉  $n$ 개의 cluster로부터 시작하여, cluster를 반복적으로 merge하여 최종적으로 모든 데이터를 포함하는 하나의 cluster를 만든다.

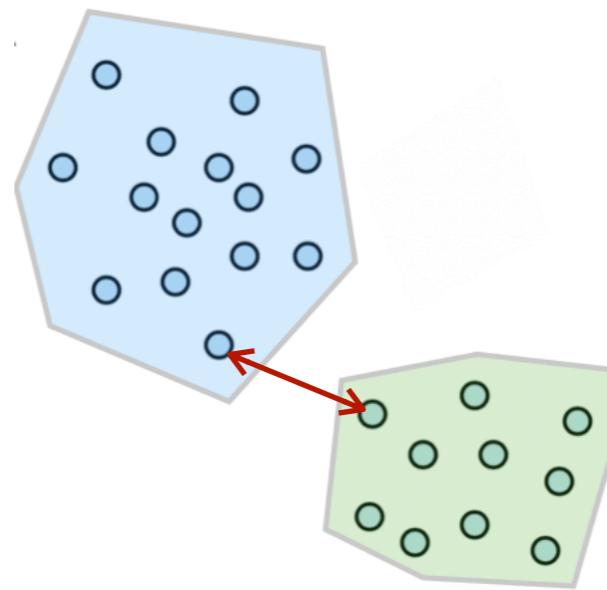
# Hierarchical Clustering

- 알고리즘의 수행 시에 두 cluster 간의 dissimilarity를 반복적으로 계산해야 한다.
- *[data cluster 기준]* 두 데이터 사이의 dissimilarity는 일반적으로 Euclidean distance를 사용하지만, 두 cluster 간의 dissimilarity를 계산하기 위한 기준이 필요하다. 일반적으로 다음과 같은 기준이 사용된다.
  - **complete linkage**: cluster A와 B에 속한 데이터 사이의 모든 dissimilarity 값 중에서 가장 큰 값을 사용한다.
  - **single linkage**: cluster A와 B에 속한 데이터 사이의 모든 dissimilarity 값 중에서 가장 작은 값을 사용한다.
  - **average linkage**: cluster A와 B에 속한 데이터 사이의 모든 dissimilarity 값들의 평균을 사용한다. ↗ min distance의 평균!
  - **centroid linkage**: cluster A의 centroid와 cluster B의 centroid 사이의 dissimilarity 값을 사용한다.

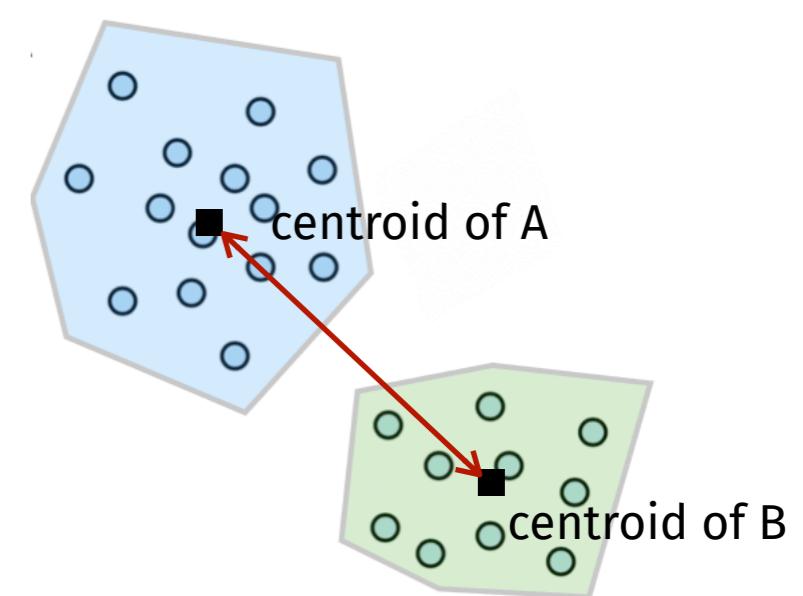
# Hierarchical Clustering



**complete linkage**

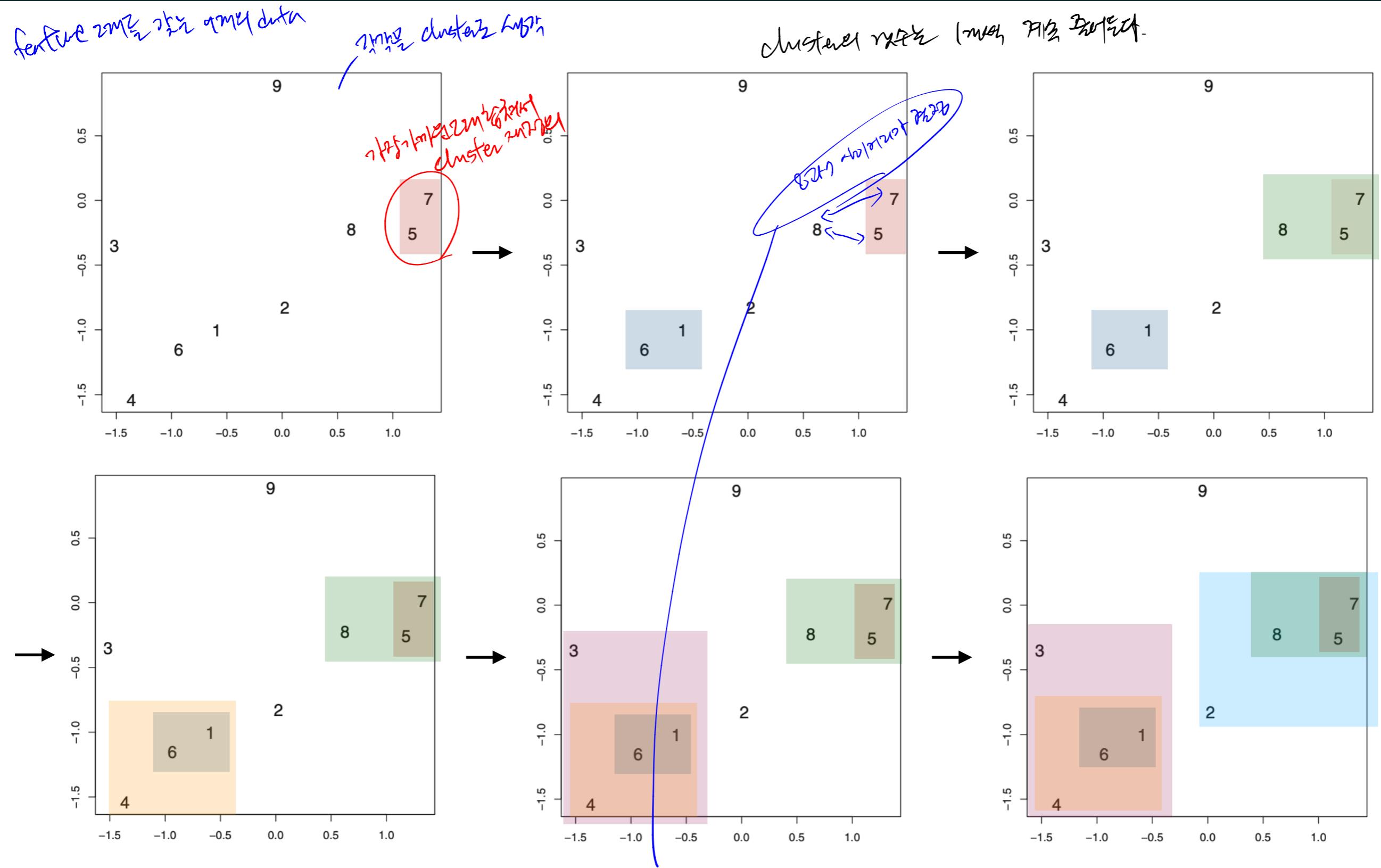


**single linkage**



**centroid linkage**

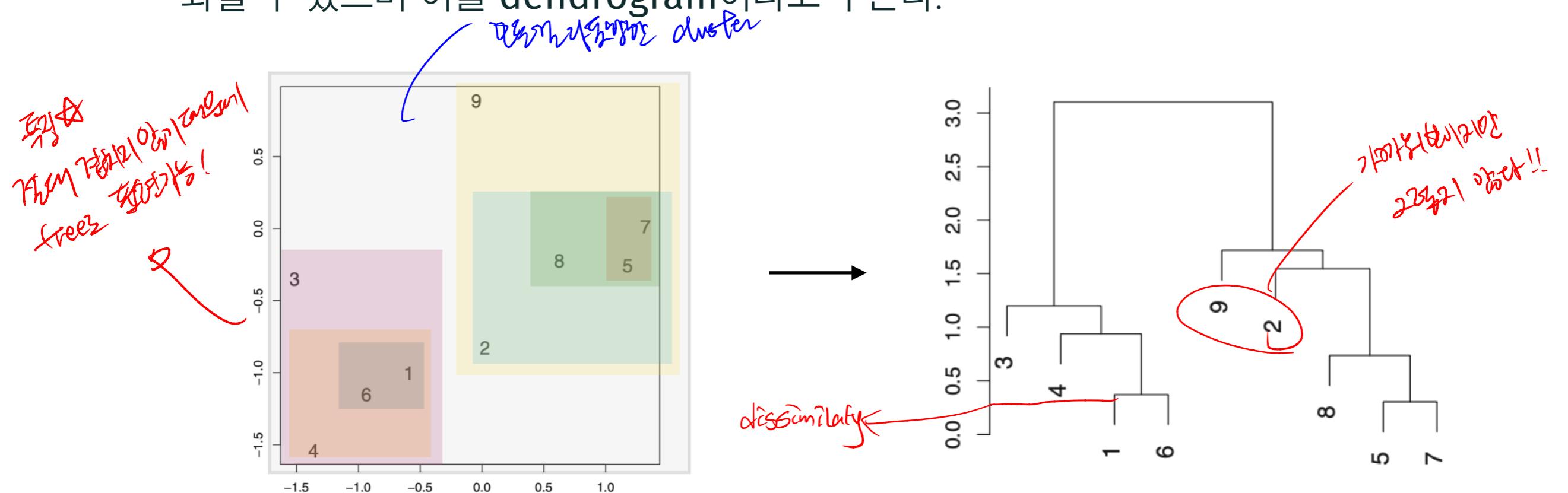
# Hierarchical Clustering



Euclidean distance와 complete linkage를 사용한 hierarchical clustering의 수행 예

# Hierarchical Clustering

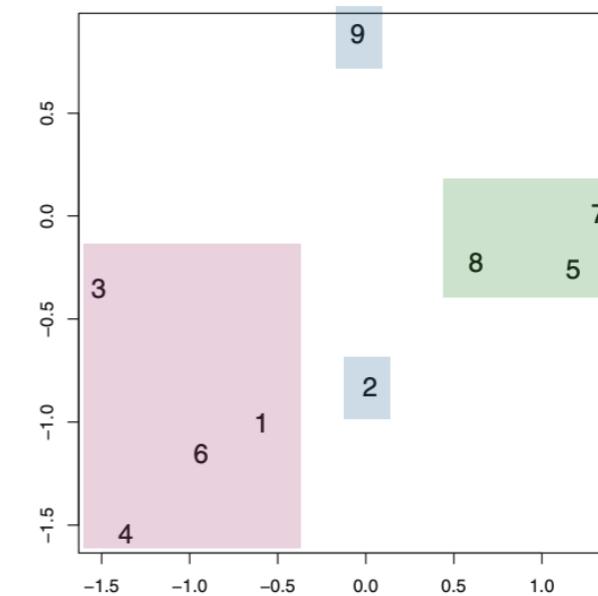
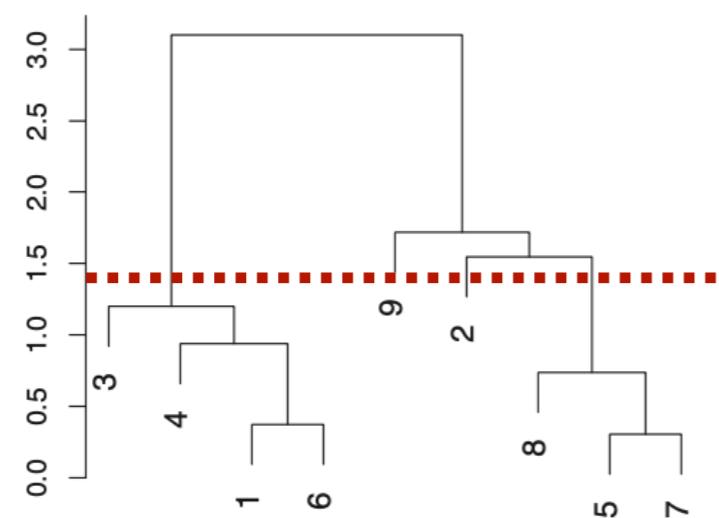
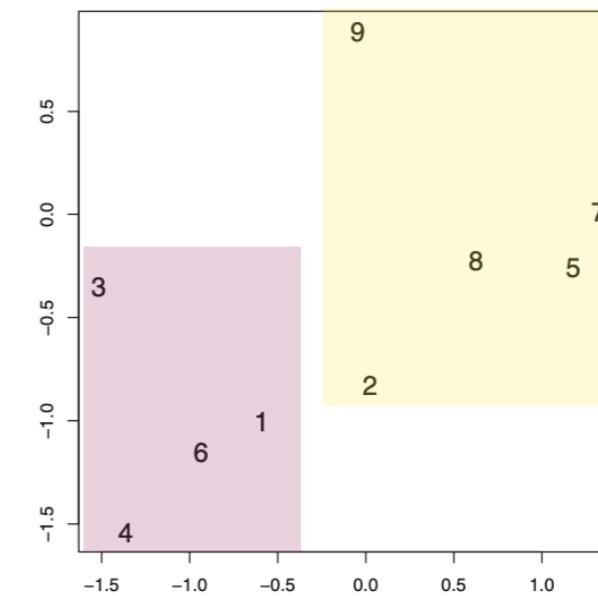
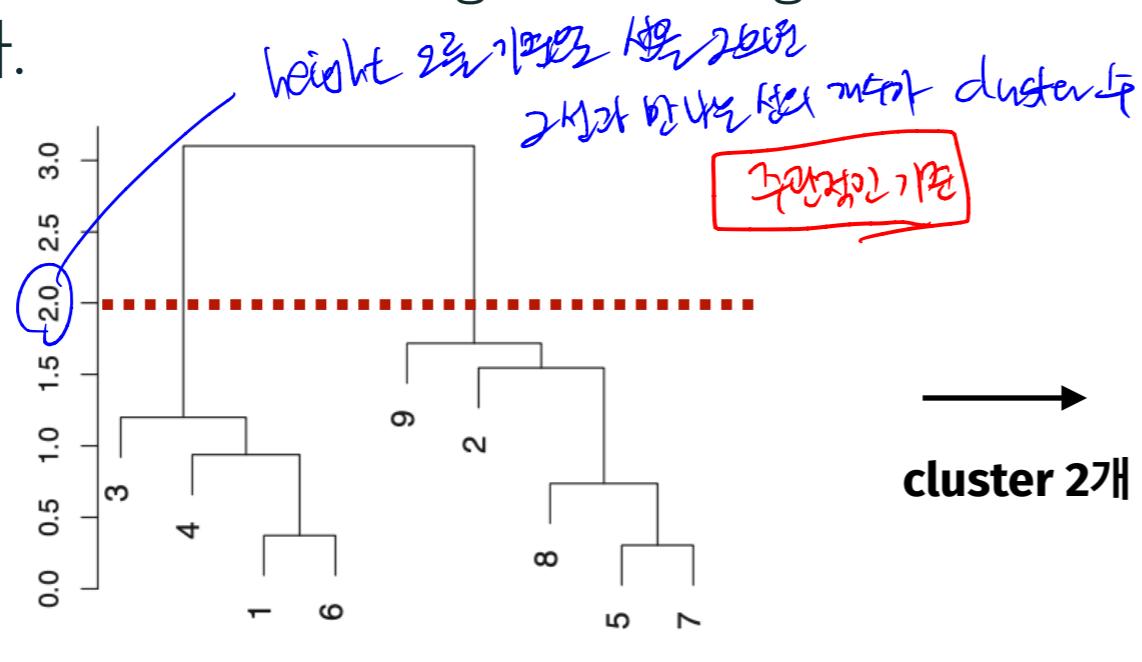
- Hierarchical clustering에서는 알고리즘이 진행됨에 따라 새로 생성되는 cluster가 이전에 생성된 cluster들을 내부에 포함하게 된다. 이러한 구조는 tree의 형태로 시각화할 수 있으며 이를 **dendrogram**이라고 부른다.



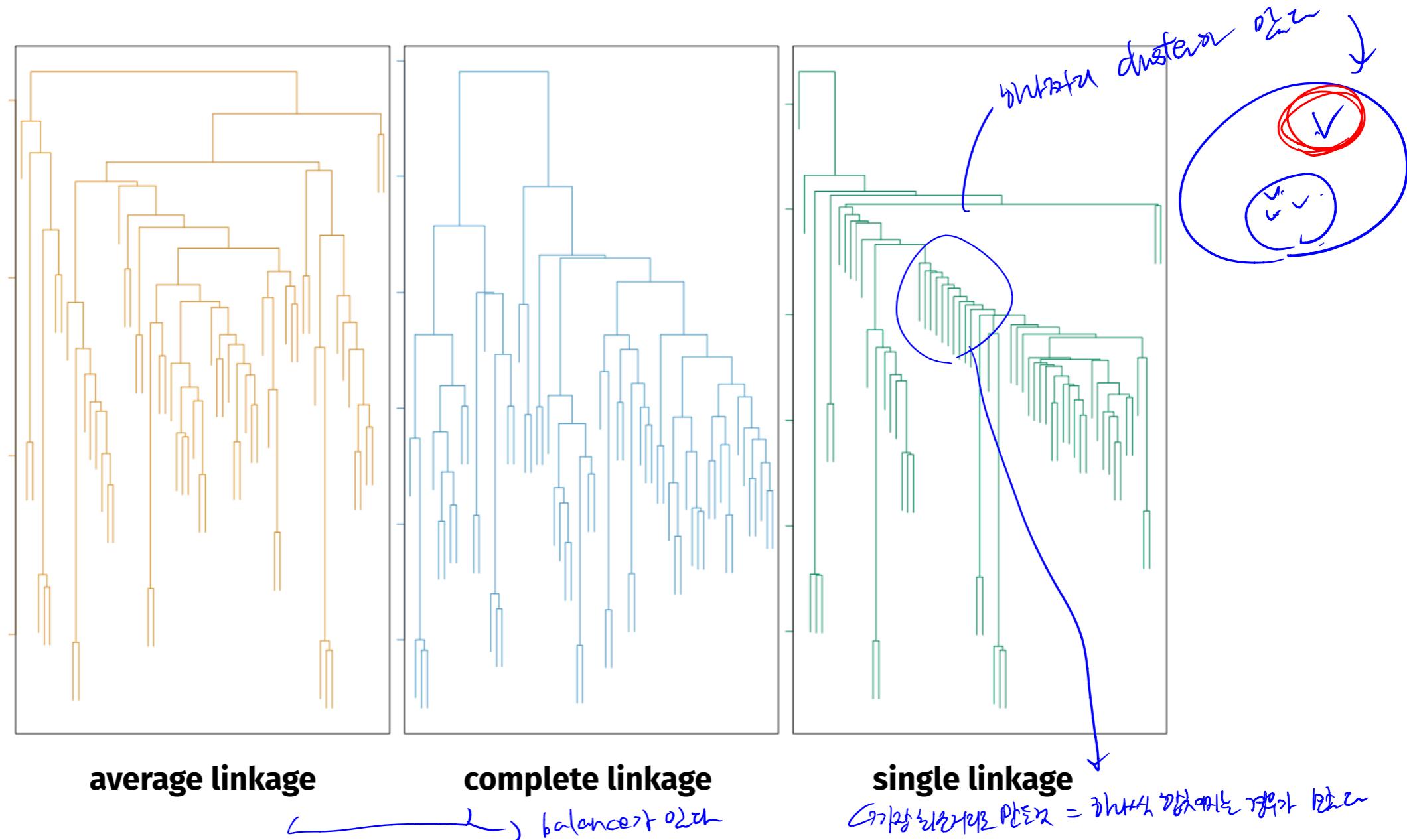
- Dendrogram의 **leaf node**는 각 데이터를 의미하고, y-축의 값 (**height**)은 두 cluster 사이의 dissimilarity 값을 의미한다. 예를 들어 cluster {3}과 cluster {4, 1, 6} 사이의 dissimilarity는 대략 1.2 정도의 값을 가진다.
- x-축에서 가까운 두 데이터가 실제로 가까운 것은 아닐 수도 있음에 주의해야 한다. 9와 2를 비교해보자.

# Hierarchical Clustering

- Dendrogram은 hierarchical clustering 알고리즘 수행 중에 반복적으로 생성되는 cluster에 대한 정보를 모두 포함하고 있기 때문에, 이로부터 적절한 수의 cluster를 찾아야 한다. Dendrogram의 height 값을 기준으로 원하는 수의 cluster를 구할 수 있다.



# Hierarchical Clustering



- Cluster 간의 dissimilarity 기준에 따라 cluster의 형태가 크게 달라질 수 있다. 일반적으로 average, complete linkage를 사용했을 때 balanced cluster가 얻어지는 경향이 있다.

# Practical Issues

- Clustering의 결과는 여러 요인에 의해서 크게 달라질 수 있다. 실제 clustering을 적용할 시 다음 사항들을 고려해야 한다.
- 데이터 표준화: 각 feature들 사이에 단위가 크게 다르다면 scale이 큰 feature에 의해서만 cluster가 결정될 수 있기 때문에, 이 경우에는 데이터를 표준화시키는 것이 필요하다.
  - Hierarchical clustering: cluster 사이의 dissimilarity 기준을 어떤 것을 사용할지, dendrogram으로부터 최종 cluster를 얻기 위한 height를 어떤 값을 사용하지 결정해야 한다.
    - ① dissimilarity
    - ② height
  - k-means clustering: cluster의 수  $k$ 를 어떤 값을 사용할지 결정해야 한다.
    - ① 빠른 학습
- 일반적으로 다수의 옵션에 대해 clustering을 반복적으로 수행해보고, domain knowledge를 기반으로 가장 의미있는 cluster를 선택한다.

↓  
데이터 속에는 domain이 있고  
domain의 특성은 있다.

# R Practice

---

# Clustering on Simulated Data

- 아래와 같이 두 class를 가지는 데이터를 랜덤으로 생성한 후 clustering 알고리즘을 적용해보자.

feature가 2개인 데이터  
zoom 가능

```
# 두 개의 feature를 가지는 200개의 데이터 생성
set.seed(123)
x <- matrix(rnorm(200*2), ncol = 2)
```

```
# 첫 100개 데이터의 두 feature 값을 일정한 크기로 증가 및 감소시켜 데이터셋을 두 클래스로 나눈다.
```

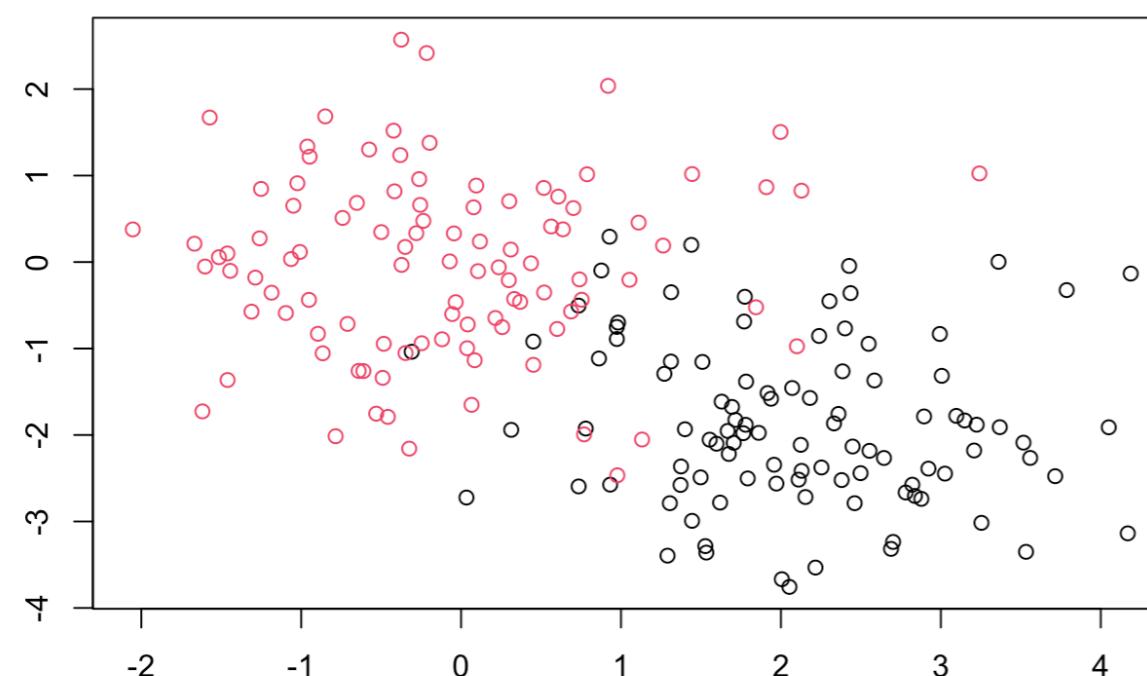
```
x[1:100,1] <- x[1:100,1] + 2
x[1:100,2] <- x[1:100,2] - 2
```

) 차원 1번, 2번을 100% 이외로 나누기

```
# 데이터 시각화
```

```
plot(x, col=c(rep(1,100), rep(2,100))), xlab="", ylab="")
```

zoom (화면)

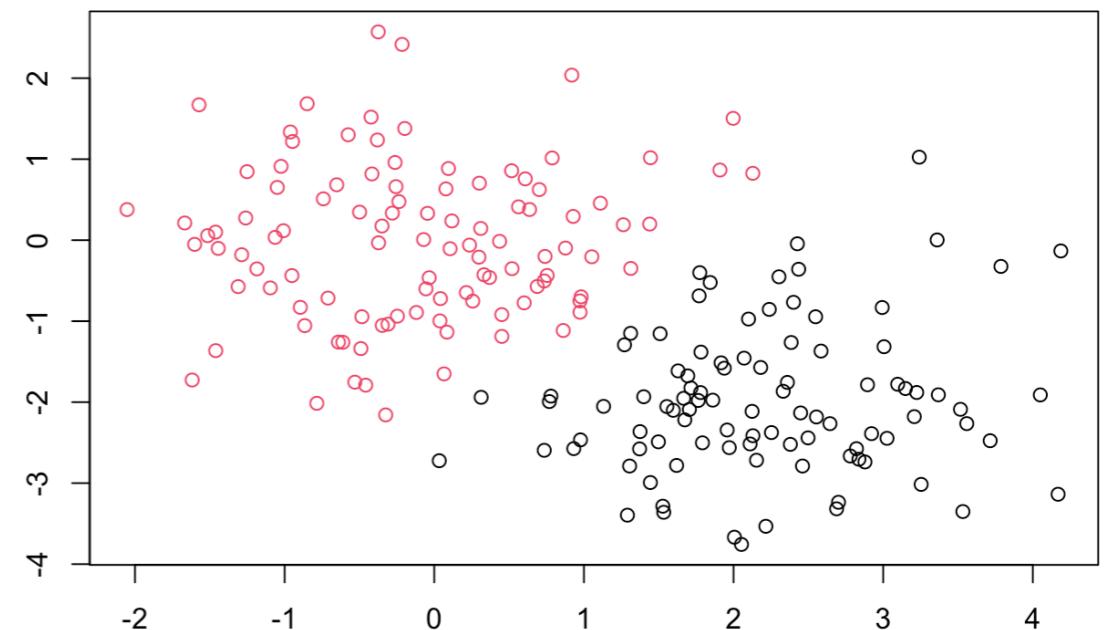


- 두 class를 완벽하게 분할하기는 어려울 것으로 예상된다.

# *k*-means Clustering

- kmeans() 함수를 이용하여 k-means clustering을 적용할 수 있으며 다음 옵션을 사용한다.
    - centers: cluster의 수
    - nstart: 반복실행 횟수 (반복 시마다 랜덤으로 선택되는 초기 cluster가 바뀐다.)

- 실제 두 클래스와 매우 유사한 형태의 cluster가 생성되었음을 확인할 수 있다.



# *k*-means Clustering

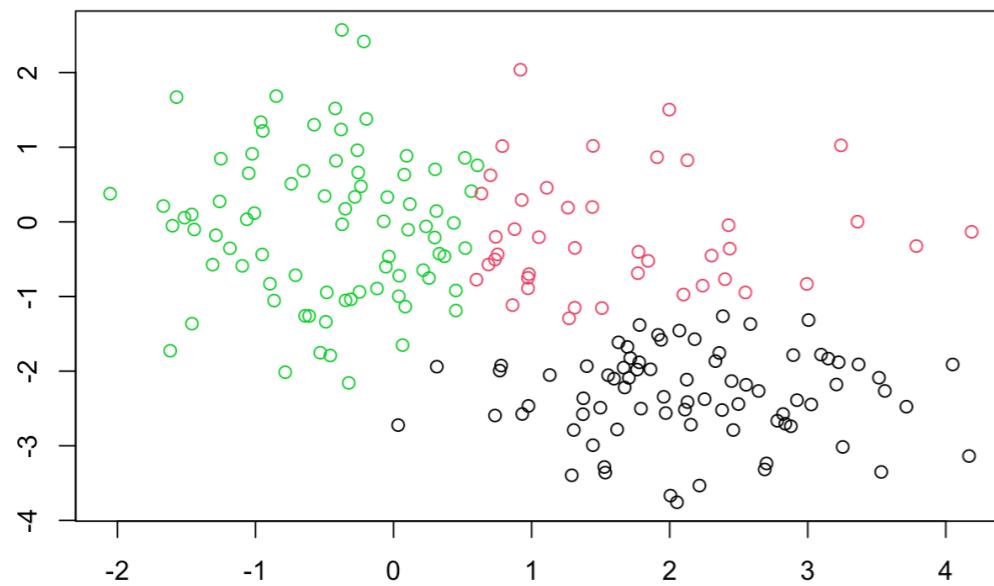
- 각 Cluster의 size,  $W(C_k)$  값 (withinss), 목적함수 값 (tot.withinss), center의 값 등을 확인할 수 있다.

# *k*-means Clustering

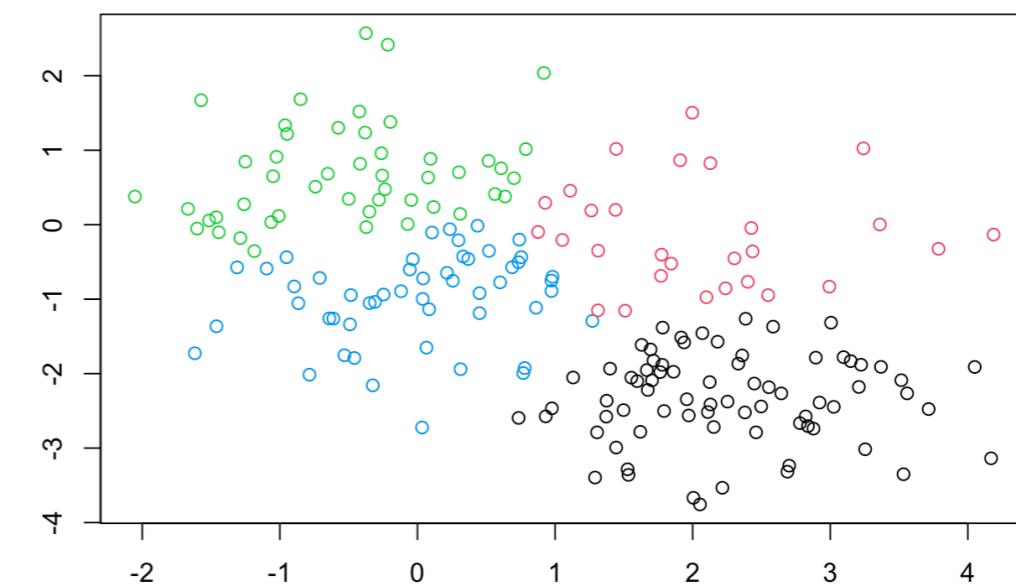
- 실제 clustering을 적용할 때에는 cluster의 수를 모르는 것이 일반적이기 때문에 다른 *k*값에 대해서도 *k*-means를 적용하여 결과를 비교해보자.

```
# k=3 일때 k-means clustering
km3 <- kmeans(x, centers = 3, nstart = 20)
plot(x, col=km3$cluster, xlab="", ylab="")
1.2번과 2번 세그를 3번으로 바꿔버려!!

# k=4 일때 k-means clustering
km4 <- kmeans(x, centers = 4, nstart = 20)
plot(x, col=km4$cluster, xlab="", ylab="")
```



$k = 3$



$k = 4$

# Hierarchical Clustering

- hclust() 함수를 이용하여 hierarchical clustering을 적용할 수 있다. dist() 함수는 Euclidean distance를 계산해주며, method 옵션으로 complete, average, single, centroid linkage를 각각 적용할 수 있다.

dist(~, "")  
↳ 쪽지, 폴더 등 가능 선택 가능

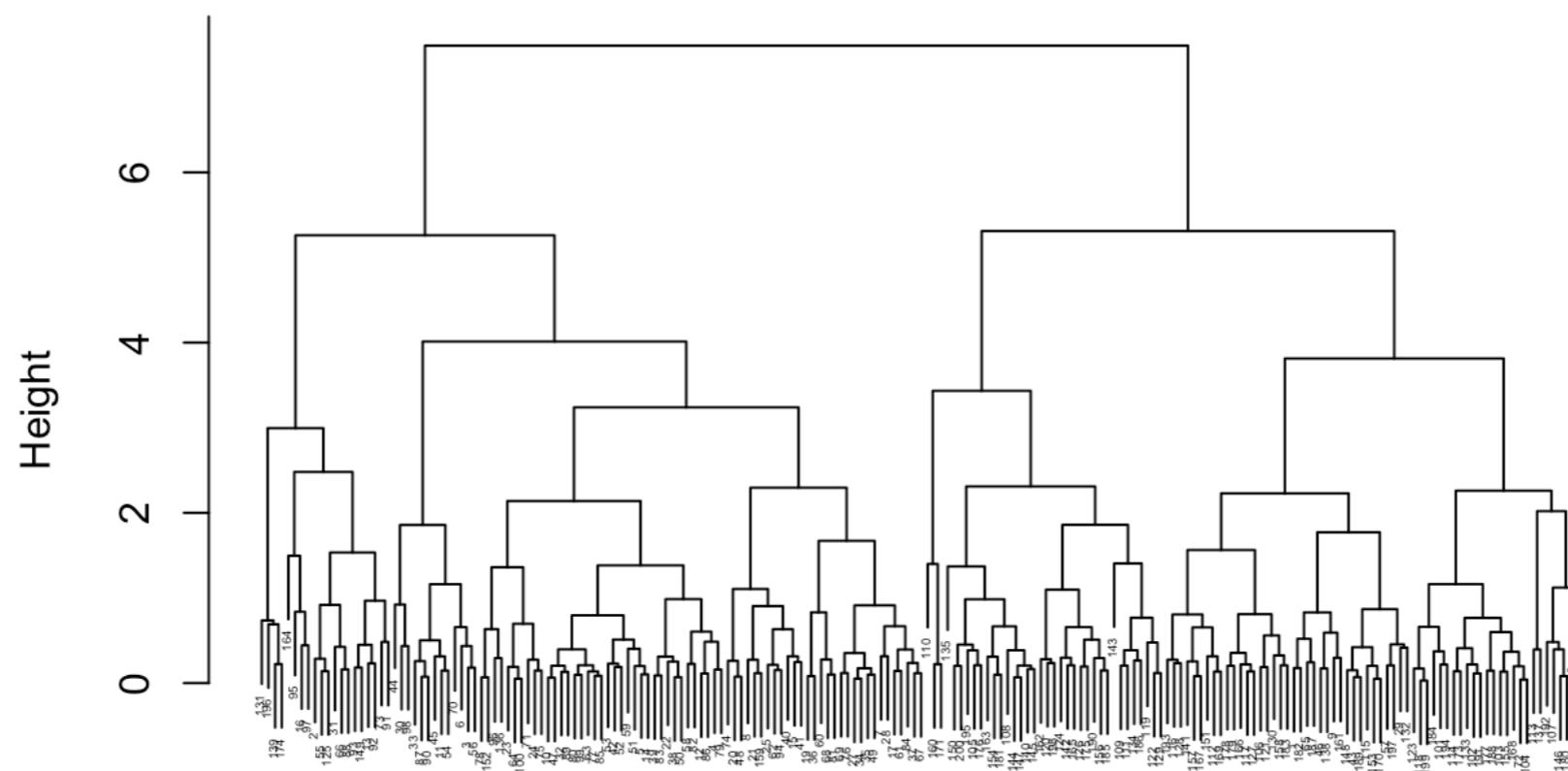
모든 사이를 포함한다.

```
# hierarchical clustering using Euclidean distance and complete linkage
hc_compl <- hclust(dist(x), method = "complete")
plot(hc_compl, cex=0.3, xlab="", sub="")
```

Label 끝까지 설정

일반화로 label=FALSE

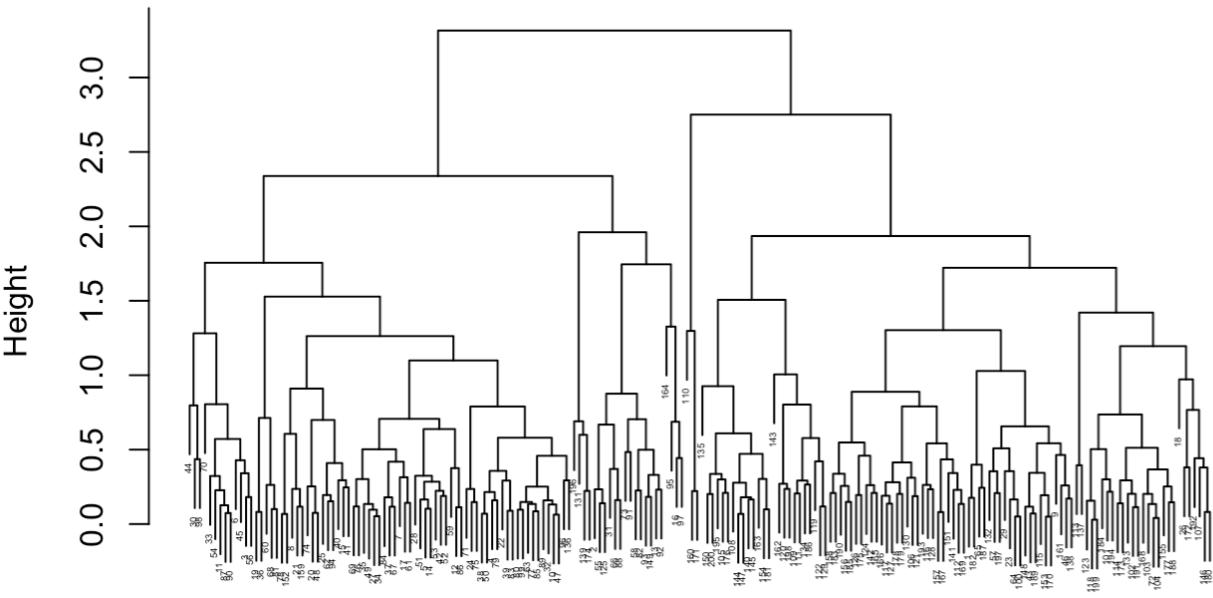
Cluster Dendrogram



# Hierarchical Clustering

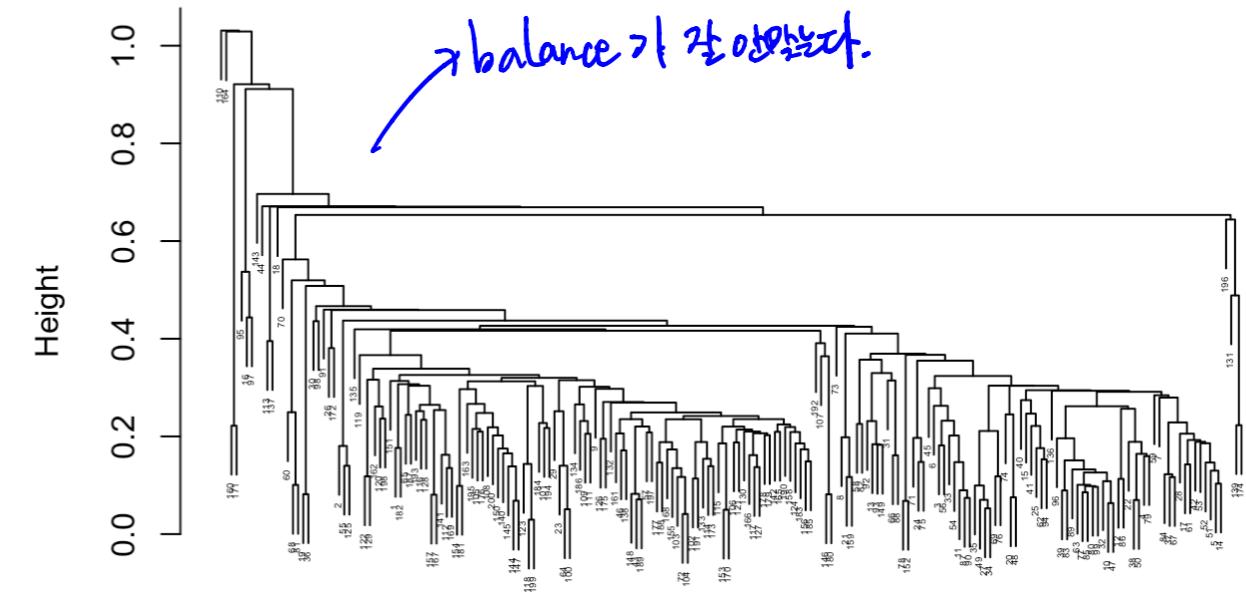
```
# average linkage  
hc_avg <- hclust(dist(x), method = "average")  
plot(hc_avg, cex=0.3, xlab="", sub="")  
  
# single linkage  
hc_sin <- hclust(dist(x), method = "single")  
plot(hc_sin, cex=0.3, xlab="", sub="")
```

Cluster Dendrogram



average linkage

Cluster Dendrogram



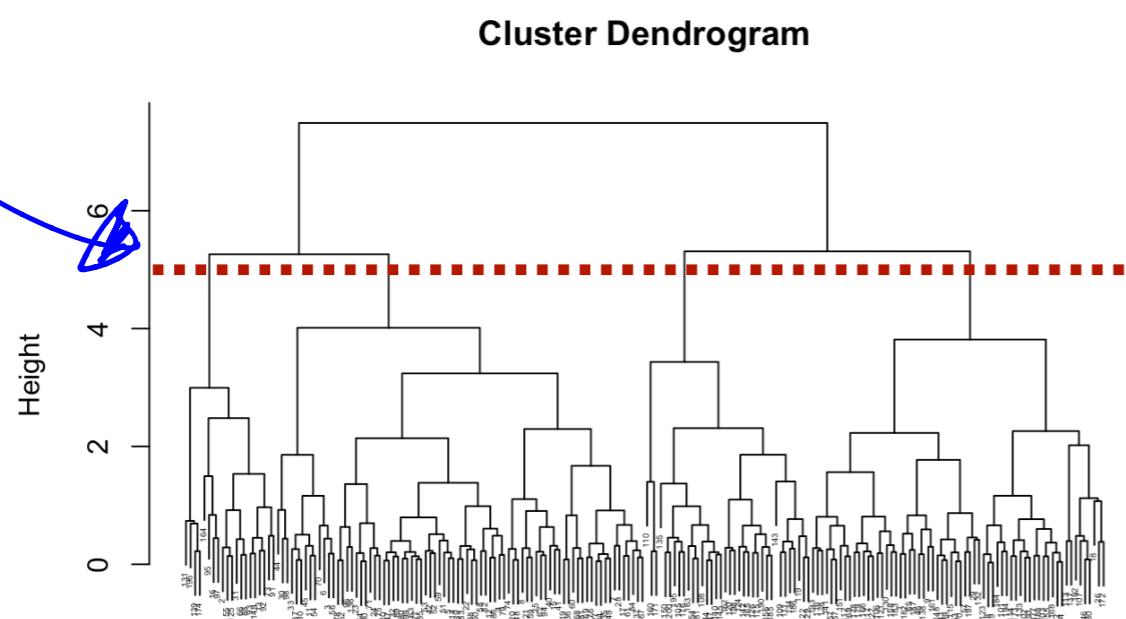
single linkage

balance가 잘 안됩니다.

- average, complete linkage가 single linkage에 비해 더 balanced된 cluster를 생성한다.

# Hierarchical Clustering

- `cutree()` 함수를 이용하여 dendrogram으로부터 원하는 수의 cluster를 얻을 수 있다. 이때 다음 옵션을 사용한다.
    - `h`: tree를 자르는 height 값
    - `k`: cluster의 수

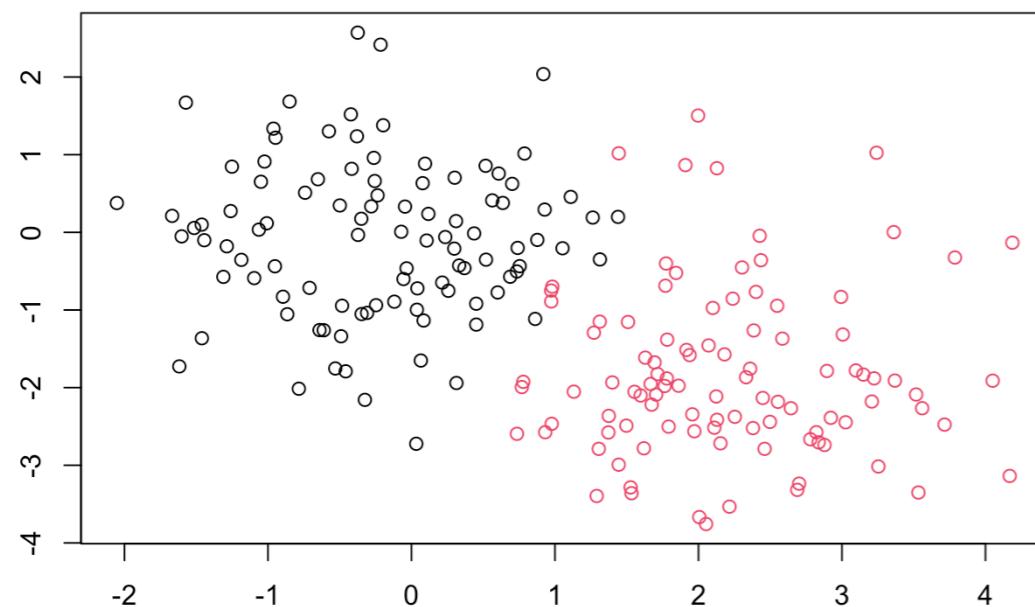


- height = 5를 기준으로 4개의 cluster가 만들어진다.

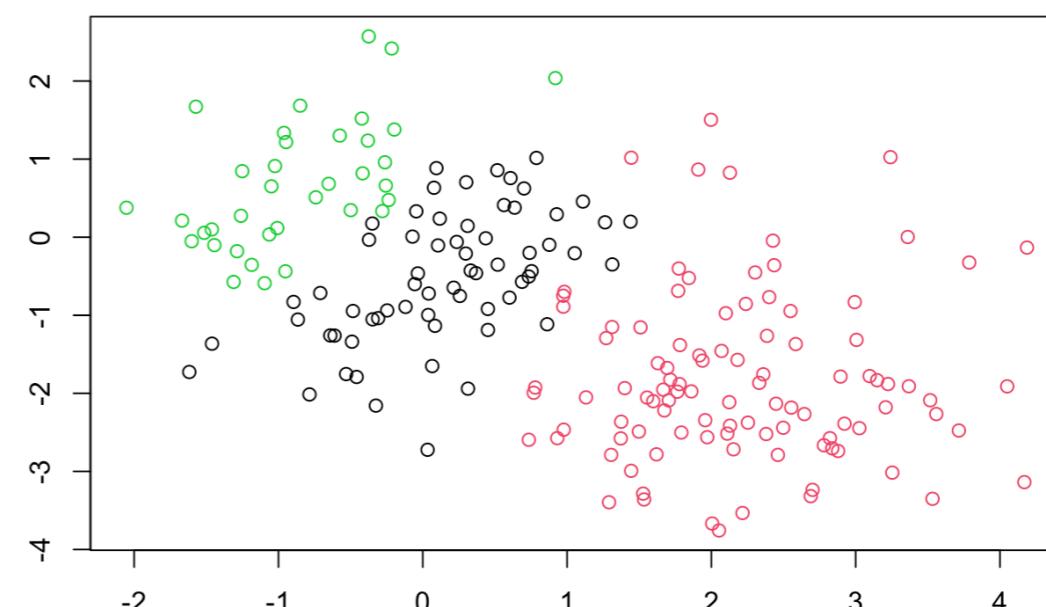
# Hierarchical Clustering

```
# 2개의 cluster 생성  
cl2 <- cutree(hc_compl, k = 2)  
plot(x, col=cl2, xlab="", ylab="")  
# 3개의 cluster 생성  
cl3 <- cutree(hc_compl, k = 3)  
plot(x, col=cl3, xlab="", ylab="")
```

cluster를 2개로 만들면  
↳ cl2는 cluster로 구성되어 있다.



$k = 2$



$k = 3$

- $k = 2$  일때는  $k$ -means 와 비슷한 결과를 얻었으나,  $k = 3$ 일때는  $k$ -means와 크게 다른 결과를 얻었다.