R 語言與機器學習 (四)

丘祐瑋
David Chiu

分群方法簡介

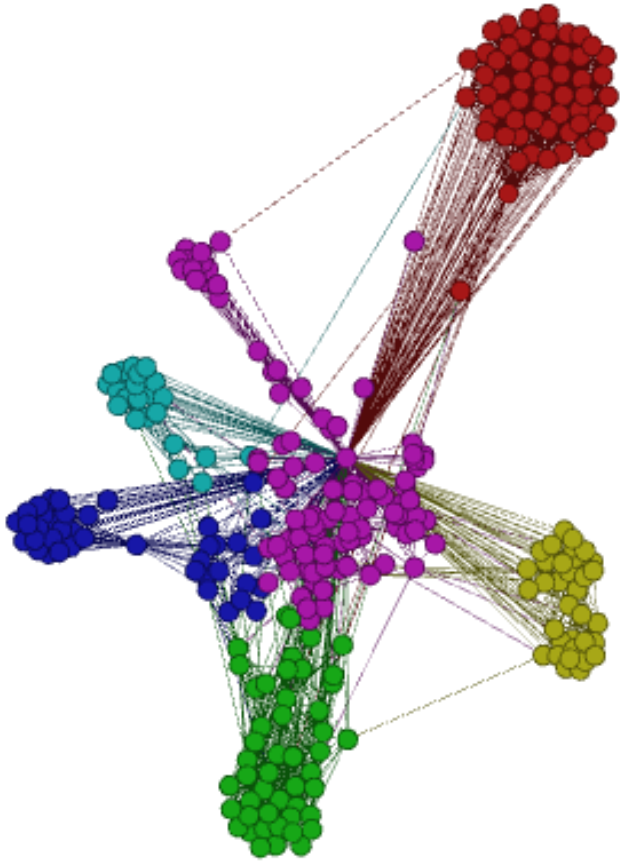
非監督式學習

- 降低維度
  - 產生一有最大變異數的欄位線性組合，可用來降低原本問題的維度與複雜度
  - e.g. 濃縮用到的特徵，編纂成一個新指標

- 分群問題
  - 物以類聚 (近朱者赤、近墨者黑)
  - e.g.分析高危險感染區域

分群應用

- 市場分析
  - 將客戶依行為跟特徵做不同區隔
  - 產品定位
  - 區分市場
- 社會網路分析
  - 找出感染高危險群
- 感染區域分析
  - 找出高危險感染區域

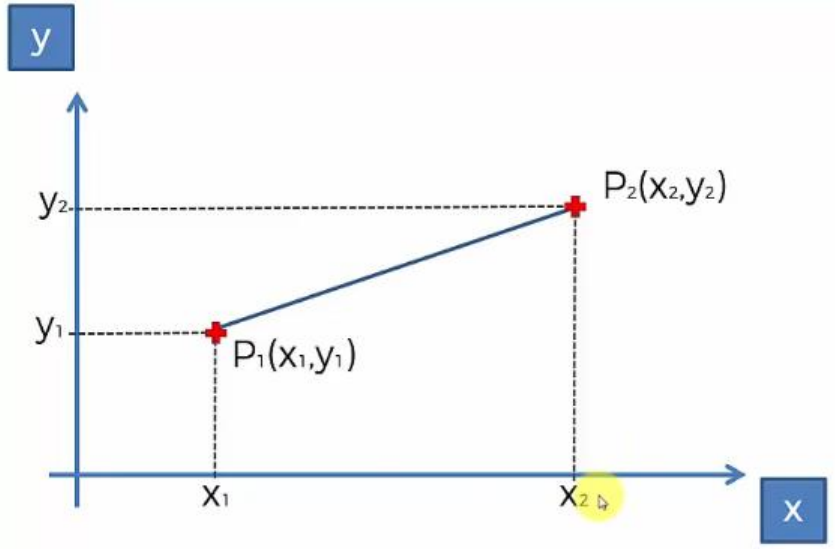


分群問題

- 特色
  - 沒有正確答案 (標籤)
  - 依靠自身屬性相似度，物以類聚

- 如何判斷相似度
  - 以『距離』作為分類的依據，『相對距離』愈近的，『相似程度』愈高，歸類成同一群組。

定義點之間的距離



Euclidean Distance between P_1 and $P_2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

歐式距離
曼哈頓距離
余弦距離
…

使用R 計算距離

```
?dist
x = c(0, 0, 1, 1, 1, 1)
y = c(1, 0, 1, 1, 0, 1)
```

■ 歐氏距離

```
dist(rbind(x,y), method =  "euclidean")
dist(rbind(x,y), method ="minkowski", p=2)
```

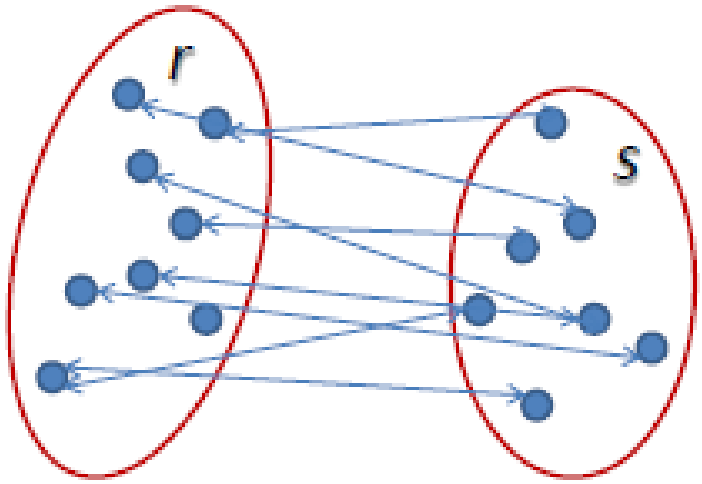
■ 曼哈頓距離

```
dist(rbind(x,y), method =  "manhattan")
dist(rbind(x,y), method ="minkowski", p=1)
```

點間距離與群間距離

- 點間距離能衡量兩點間的距離
  - 相近的點可被視為類似樣本點
- 但如何去計算群與群之間的相似度
  - 單一連結聚合演算法
  - 平均連結聚合演算法
  - 完整連結聚合演算法
  - 沃德法

定義群之間的距離



- 單一連結聚合演算法

$$d(C_i, C_j) = \min_{\mathbf{a} \in C_i, \mathbf{b} \in C_j} d(\mathbf{a}, \mathbf{b})$$

- 完整連結聚合演算法

$$d(C_i, C_j) = \max_{\mathbf{a} \in C_i, \mathbf{b} \in C_j} d(\mathbf{a}, \mathbf{b})$$

- 平均連結聚合演算法

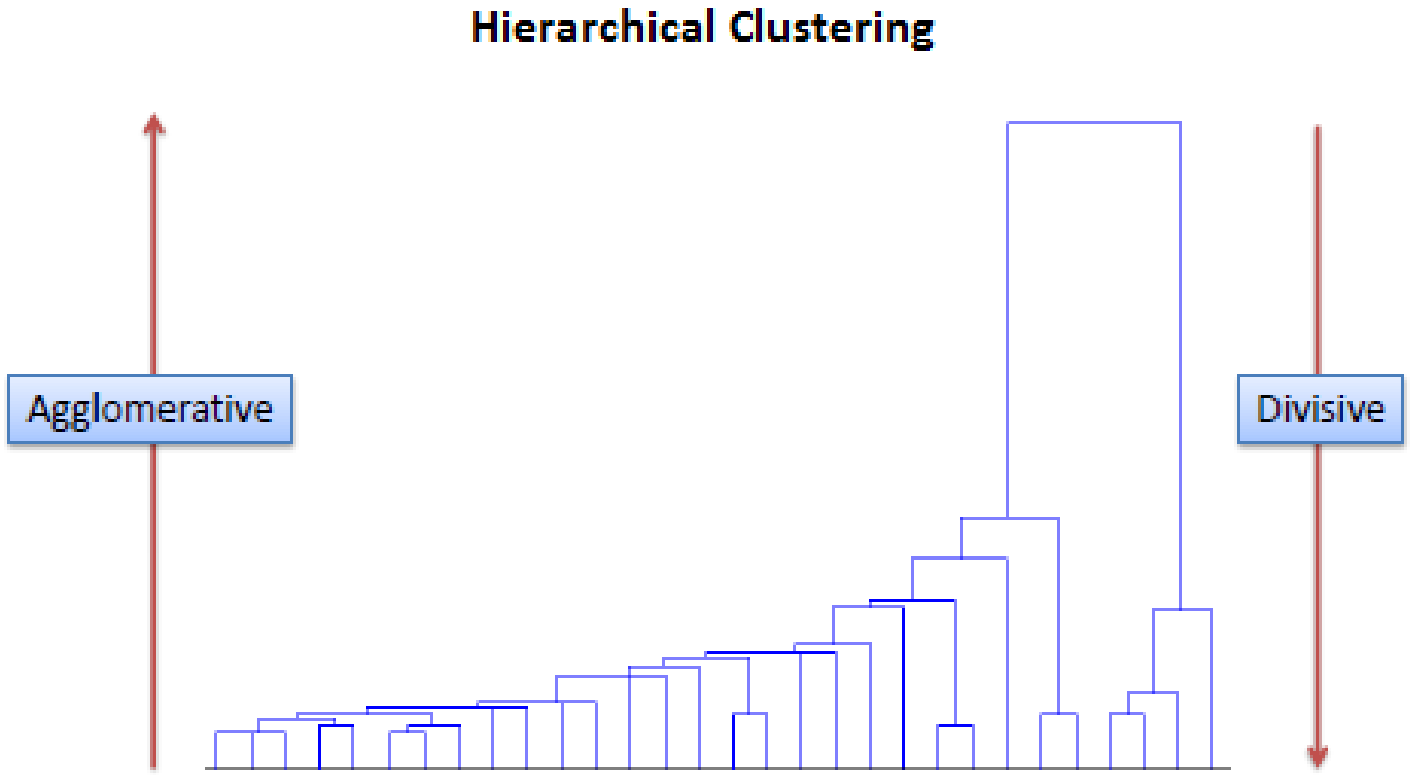
$$d(C_i, C_j) = \sum_{\mathbf{a} \in C_i, \mathbf{b} \in C_j} \frac{d(\mathbf{a}, \mathbf{b})}{|C_i||C_j|},$$

- 沃德法

$$d(C_i, C_j) = \sum_{\mathbf{a} \in C_i \cup C_j} \|\mathbf{a} - \mu\|,$$

階層式分群

- 聚合式、分裂式



聚合式分群

- 聚合式分群
  - 階層式分群法可由樹狀結構的底部開始，將資料或群聚逐次合併
  - 最終合併為一個大的群組
  - 使用hclust

Given:

A set X of objects $\{x_1,\ldots,x_n\}$

A distance function $dist(c_1,c_2)$

for $i = 1$ to n

  $c_i = \{x_i\}$

end for

$C = \{c_1,\ldots,c_n\}$

$l = n+1$

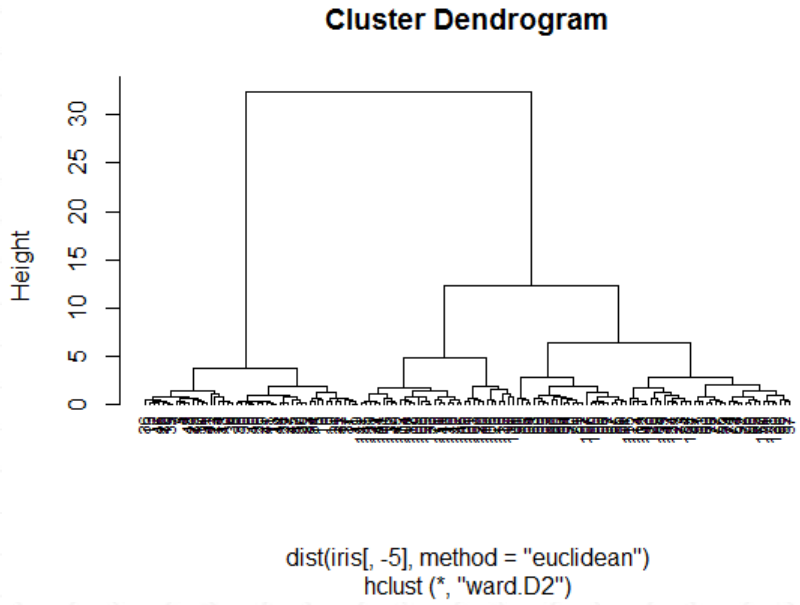
while C.size > 1 **do**

- (c_{min1},c_{min2}) = minimum $dist(c_i,c_j)$ for all c_i,c_j in C
- remove c_{min1} and c_{min2} from C
- add $\{c_{min1},c_{min2}\}$ to C
- $l = l + 1$

end while

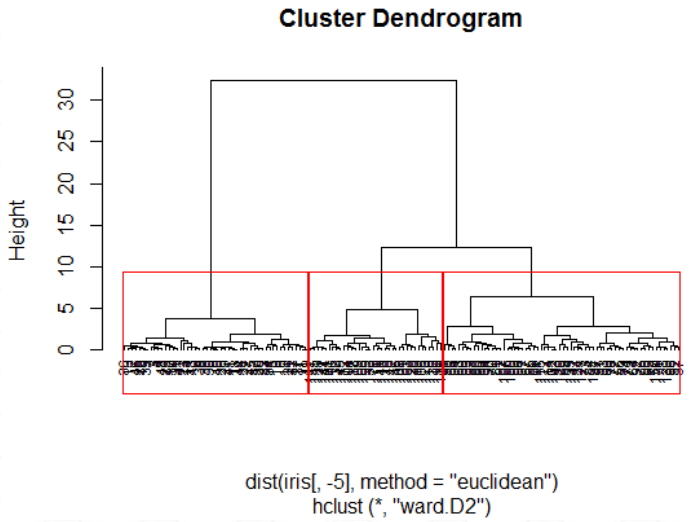
使用hclust 做iris 分群

```
data(iris)
hc <- hclust(dist(iris[,-5], method="euclidean"), method="ward.D2")
plot(hc, hang = -0.01, cex = 0.7)
```



使用cutree樹做分群

```
fit <- cutree(hc, k = 3)
table(fit)
plot(hc, hang = -0.01, cex = 0.7)
rect.hclust(hc, k = 3 , border="red")
```



階層式分群的優點/缺點

- 優點
  - 可以產生視覺化分群結果 (使用plot)
  - 可以等結構產生後，再使用cutree進行分群
  - 不用一開始決定要分多少群

- 缺點
  - 計算速度緩慢(採用遞迴式聚合或分裂)

K-Means 分群

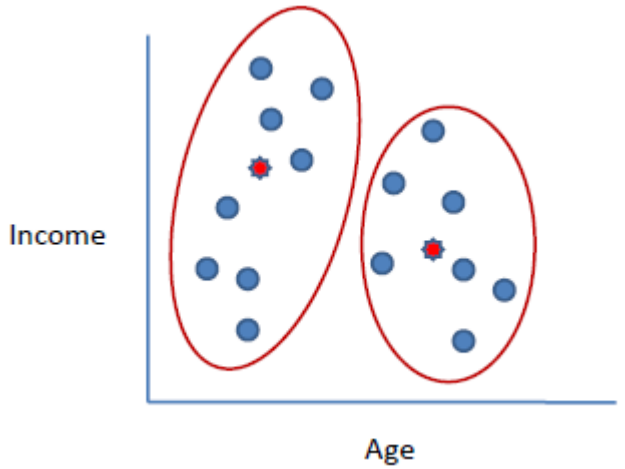
K-Means 分群

- 最小化誤差函數 (Within cluster sum of squares by cluster)
  - 將資料分為k 群
  - 所有資料點 x_j 到其對應群中心 C_i 的距離總合是最小的

number of clusters　number of cases　centroid for cluster *j*　case *i*

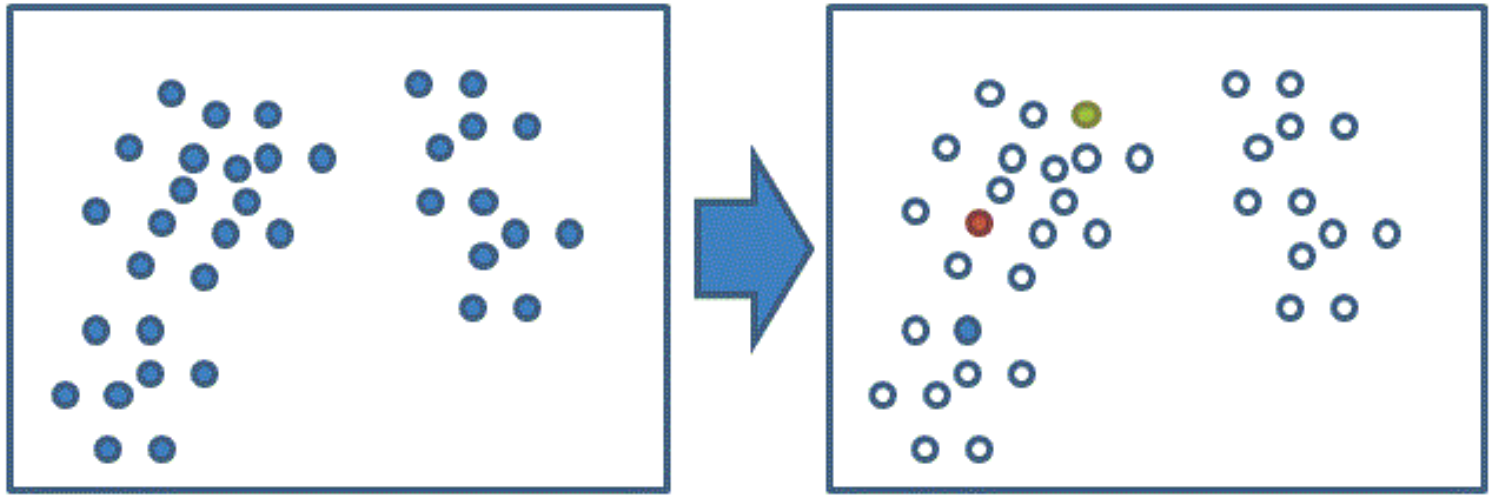
objective function ← $J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2$

Distance function



1. 隨機選取資料組中的k筆群中心

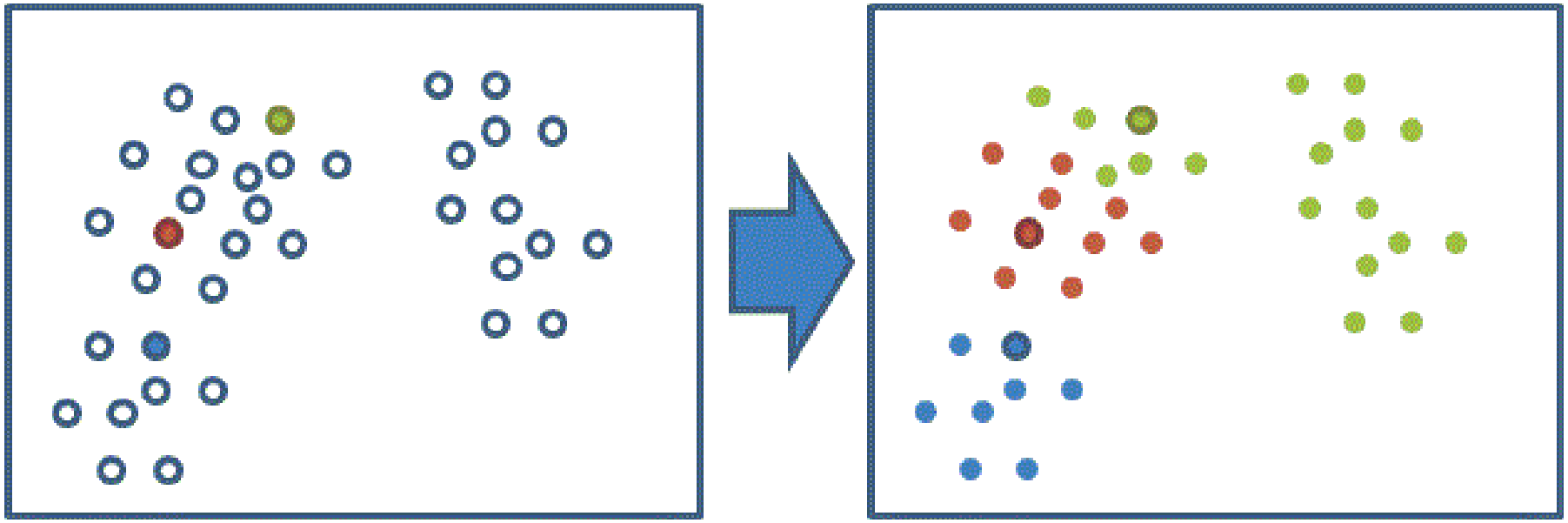
- 隨機選取資料組中的k筆資料當作初始群中心u_1~u_k



初始群中心設定的不好可能導致不會的結果

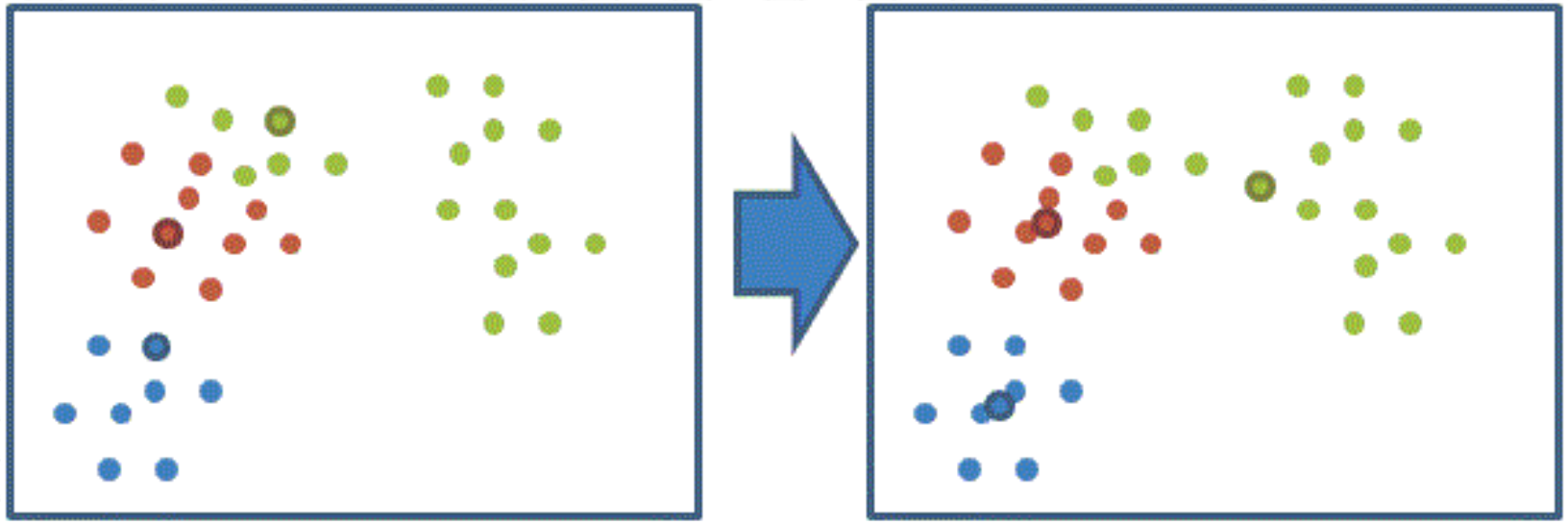
2. 計算每個資料xi 對應到最短距離的群中心

- 計算每個資料xi 對應到最短距離的群中心 (**固定 u_i 求解所屬群 S_i**)



3.利用目前得到的分類重新計算群中心

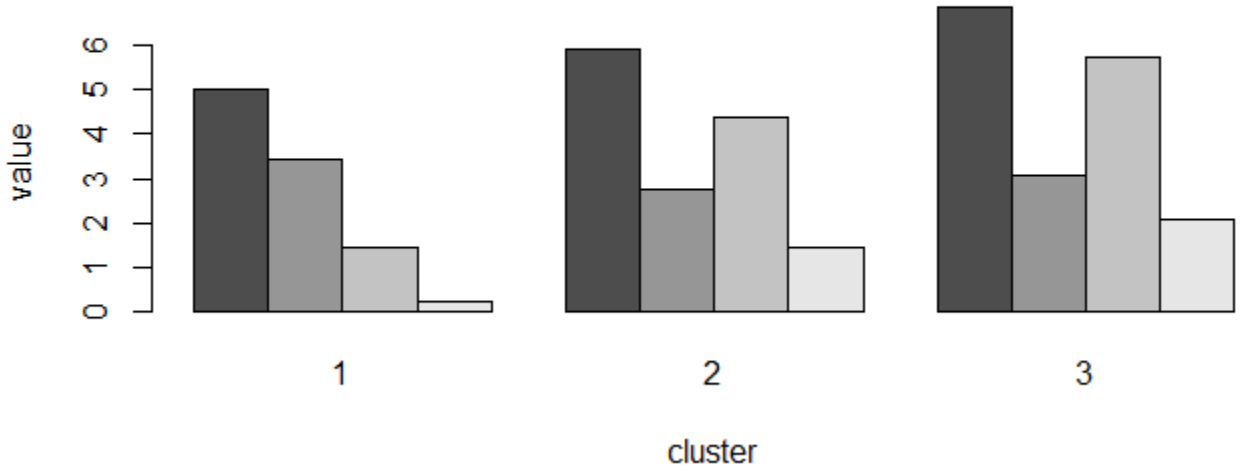
- 利用目前得到的分類重新計算群中心 (**固定 S_i 求解群中心 u_i**)



重複step 2,3直到收斂
(達到最大疊代次數 or 群心中移動距離很小)

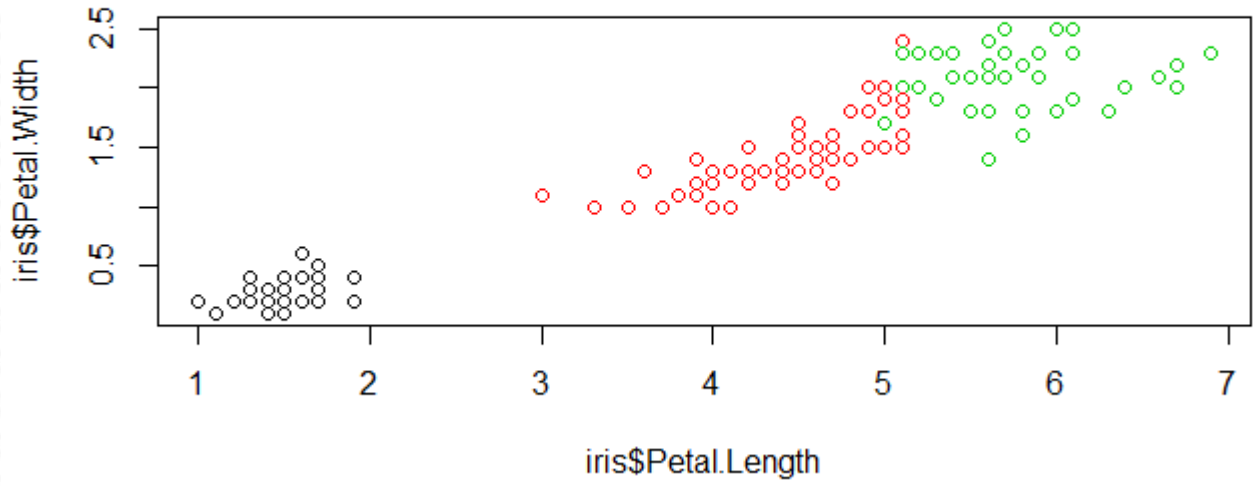
使用kmeans 分群

```
set.seed(22)
fit <- kmeans(iris[,-5], 3)
barplot(t(fit$centers), beside = TRUE,xlab="cluster", ylab="value")
```



繪製分群結果

```
plot(iris$Petal.Length, iris$Petal.Width, col =
fit$cluster)
```



評估分群效果

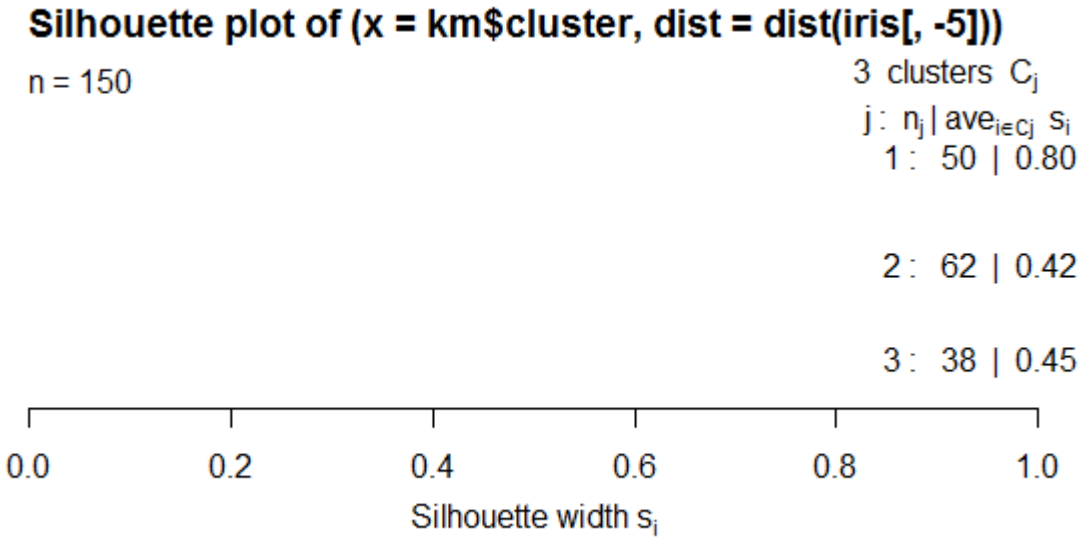
- 定義:
  - 群內之間點的平均距離 (群內的點平均距離越小)
  - 群間之間的平均距離 (群間點的平均距離越大)

$$\text{Silhouette(x)} = \frac{b(x) - a(x)}{\max([b(x), a(x)])}$$

- a(x) 為 x 距離群內其他點的平均距離
- b(x) 是 x 距離其他群內點之間的最小平均距離

計算 Average Silhouette Width

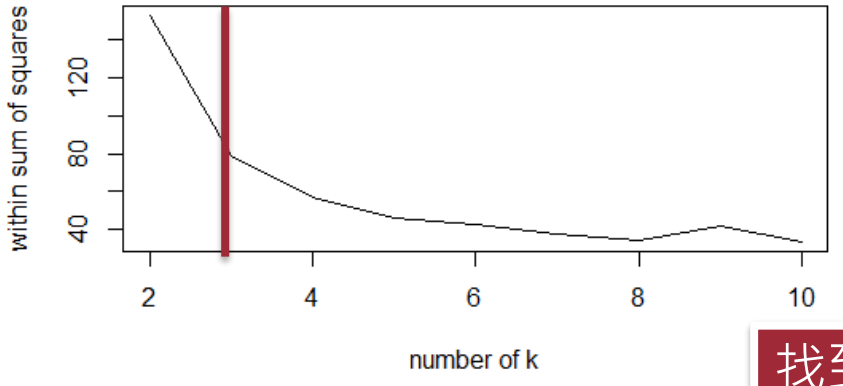
```
set.seed(22)
km  <- kmeans(iris[,-5], 3)
kms <- silhouette(km$cluster,dist(iris[,-5]))
summary(kms)
plot(kms)
```



越接近1 代表群體的品質越好

如何決定K值 (WSS)

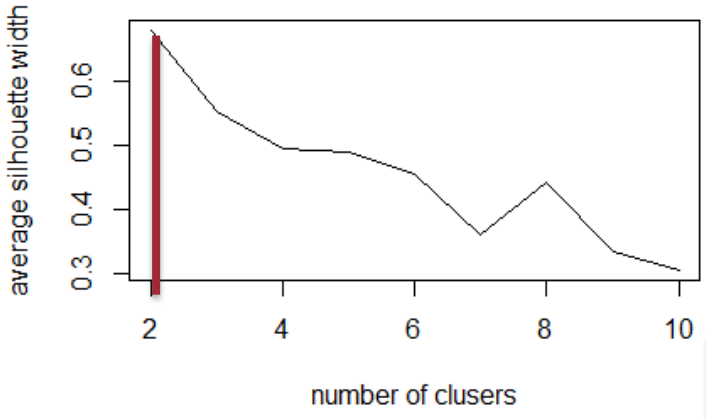
```
nk = 2:10
set.seed(22)
WSS = sapply(nk, function(k) {
  kmeans(iris[,-5], centers=k)$tot.withinss
})
WSS
plot(nk, WSS, type="l", xlab= "number of k", ylab="within sum of
squares")
```



找到坡度改變的那一點

如何決定K值 (Average Silhouette Width)

```
library(fpc)
nk <- 2:10
set.seed(42)
SW = sapply(nk, function(k) {
  cluster.stats(dist(iris[,-5]), kmeans(iris[,-5],
centers=k)$cluster)$avg.silwidth
})
plot(nk, SW, type="l", xlab="number of clusers", ylab="average
silhouette width")
```

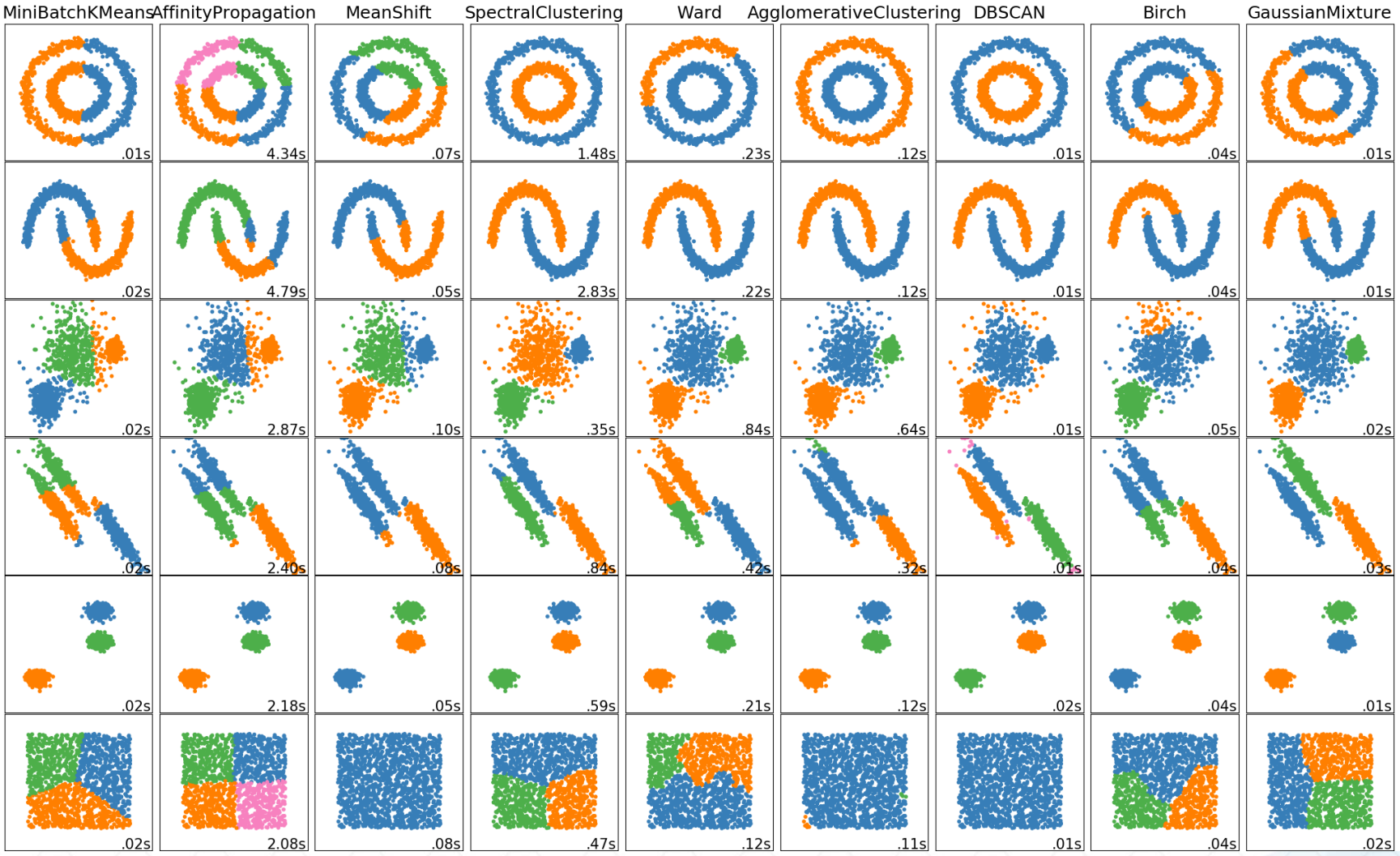


找到最大值那點

比較不同分群演算法

```
single_c   <-  hclust(dist(iris[,-5]), method="single")
hc_single  <-  cutree(single_c, k = 3)

complete_c <-  hclust(dist(iris[,-5]), method="complete")
hc_complete <-  cutree(complete_c, k = 3)

set.seed(42)
km <- kmeans(iris[,-5], 3)

cs <- cluster.stats(dist(iris[,-5]), km$cluster)
cs[c("within.cluster.ss","avg.silwidth")]

sapply(list(kmeans = km$cluster, hc_single = hc_single, hc_complete =
hc_complete), function(c) cluster.stats(dist(iris[,-5]),
c)[c("within.cluster.ss","avg.silwidth")])
```

分群演算法



http://scikit-learn.org/stable/modules/clustering.html

發病區域分析

運用視覺化分析找出霍亂疫情元兇

- 1854 年，霍亂疫情爆發，造成十天之內死了五百多人
- Dr. John Snow 將所有病患的住家位置點在地圖上，發現病例聚集在一口井附近

取得登革熱資料

```
url <- 'https://raw.githubusercontent.com/ywchiu/cdc_course/master/data/Dengue_2015_09_Tainan.csv''
Dengue <- read.csv(url)
head(Dengue)
```

	X <int>	最小統計區中心點X <dbl>	最小統計區中心點Y <dbl>	發病日 <fctr>
1	1	120.1610	22.99350	2015-09-01
2	2	120.2261	23.01015	2015-09-01
3	3	120.2190	23.01516	2015-09-01
4	4	120.2077	23.00104	2015-09-01
5	5	120.2136	22.98719	2015-09-01
6	6	120.2107	23.00696	2015-09-01

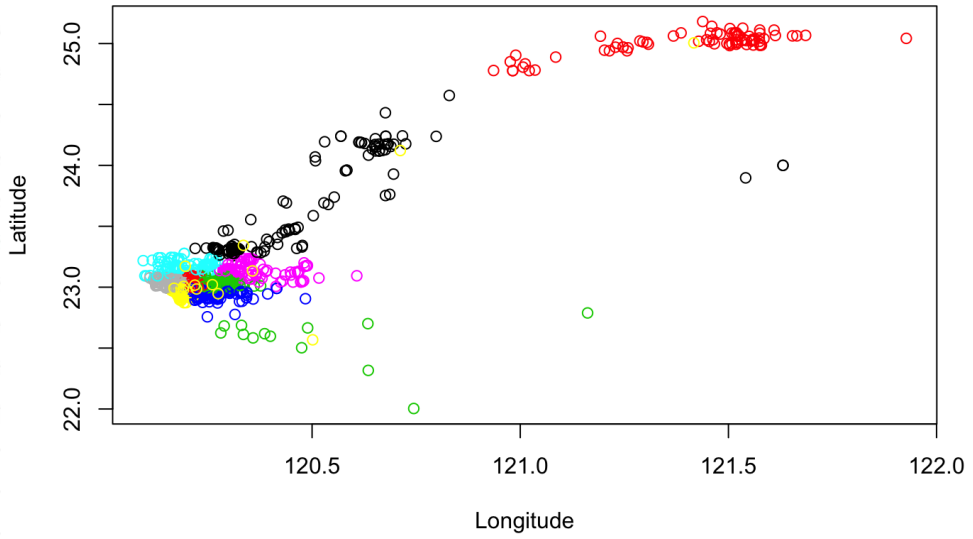
資料來源:https://data.gov.tw/dataset/21025

使用kmeans 分群

```
Dengue <- Dengue[,c('最小統計區中心點X', '最小統計區中心點Y')]
Dengue <- Dengue[! is.na(Dengue$最小統計區中心點X) & ! is.na(Dengue$最小統計區中心點Y), ]
set.seed(123)
kc <-  kmeans(Dengue, centers = 12)
kc
```

繪製分群結果

```
plot(最小統計區中心點Y~最小統計區中心點X, data = Dengue,
col=kc$cluster,  xlab= 'Longitude', ylab = 'Latitude')
points(kc$centers[,1], kc$centers[,2], col='yellow')
```



計算 Average Silhouette Width

```
library(cluster)
kcs <- silhouette(kc$cluster,dist(Dengue))
summary(kcs)
plot(kcs)
```

找出最好的K值

```
library(fpc)
nk <- 2:20
set.seed(123)
SW <- sapply(nk, function(k) {
  cluster.stats(dist(Dengue), kmeans(Dengue, centers=k)$cluster)$avg.silwidth
})
plot(nk, SW, type="l", xlab="number of clusers", ylab="average silhouette width")
```

到底找到哪一點比較合適?

DBSCAN

密度為基礎分群法（DBSCAN）

- **D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise
- 以密度為基礎的分群方式，用密度的概念剔除不屬於所有分群資料的雜訊點

DBSCAN示意圖

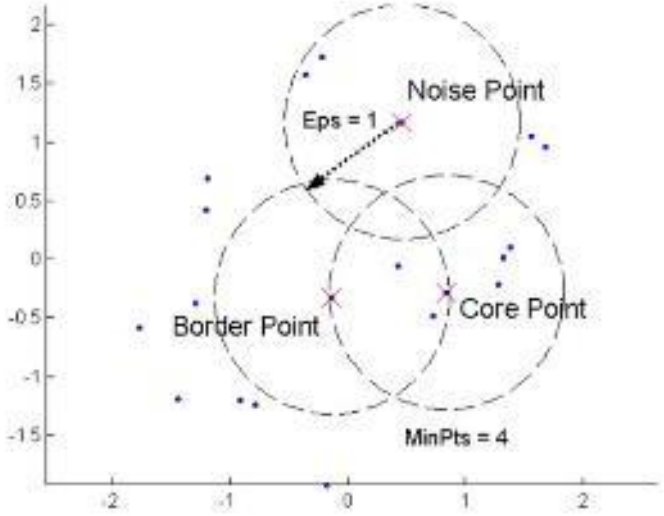
1. Eps = Density = 以資料點為圓心所設的半徑長度

2. Core Point（核心點）：以核心點為半徑所圍繞出來的範圍

3. Border Point（ 邊界點）：被某個核心點包含

4. Noise Point（雜訊點）：不屬於核心點，也不屬於邊界點，即為雜訊點

5. 密度相連：如果兩個核心點互為邊界點的話，則可把兩個核心點合併在同一個群組中



DBSCAN演算法

1. 將所有的點做過一次搜尋，找出**核心點、邊界點、雜訊點**
2. 移除所有雜訊點
3. SET「當前群集編號」＝0
4. FOR 1 到 最後一個核心點 do
5. IF 這個核心點並沒有被貼上群組編號 則
6. 「當前群集編號」的變數 + 1
7. 把「當前群集編號」給這個被抽出的核心點
8. END
9. FOR 這個核心點在密度相連後所有可以包含的點 do
10. IF 這個點還沒有被貼上任何群組編號 則
11. 把這個點貼上「當前變數的編號」
12. END
13. END FORLOOP
14. END FORLOOP

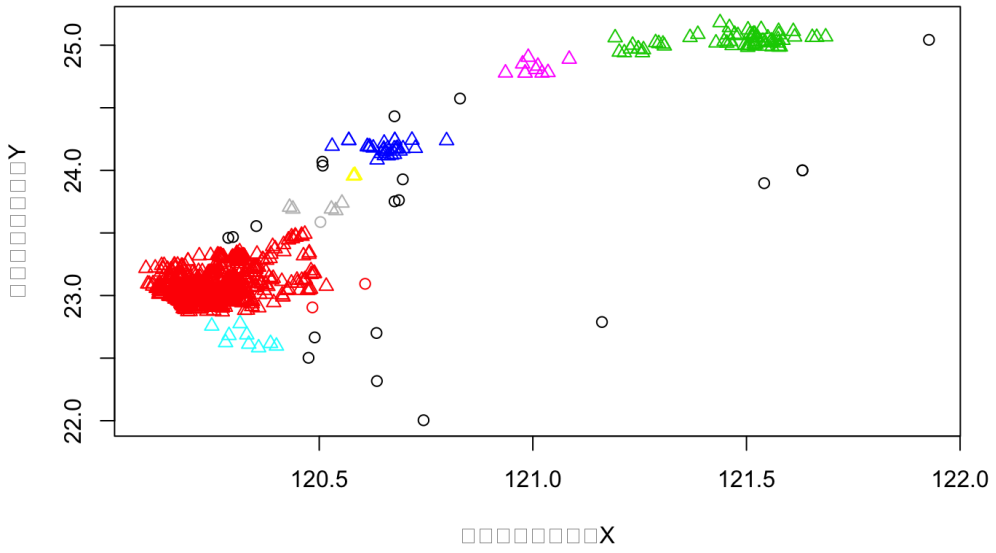
與K-means 比較

- 優點
  - 與K-means方法相比，DBSCAN不需要事先知道K
  - 與K-means方法相比，DBSCAN可以找到任意形狀
  - DBSCAN能夠識別出雜訊點
  - DBSCAN對於資料庫中樣本的順序不敏感

- 缺點
  - DBScan不適合反映高維度資料。
  - DBScan不適合反映已變化資料的密度

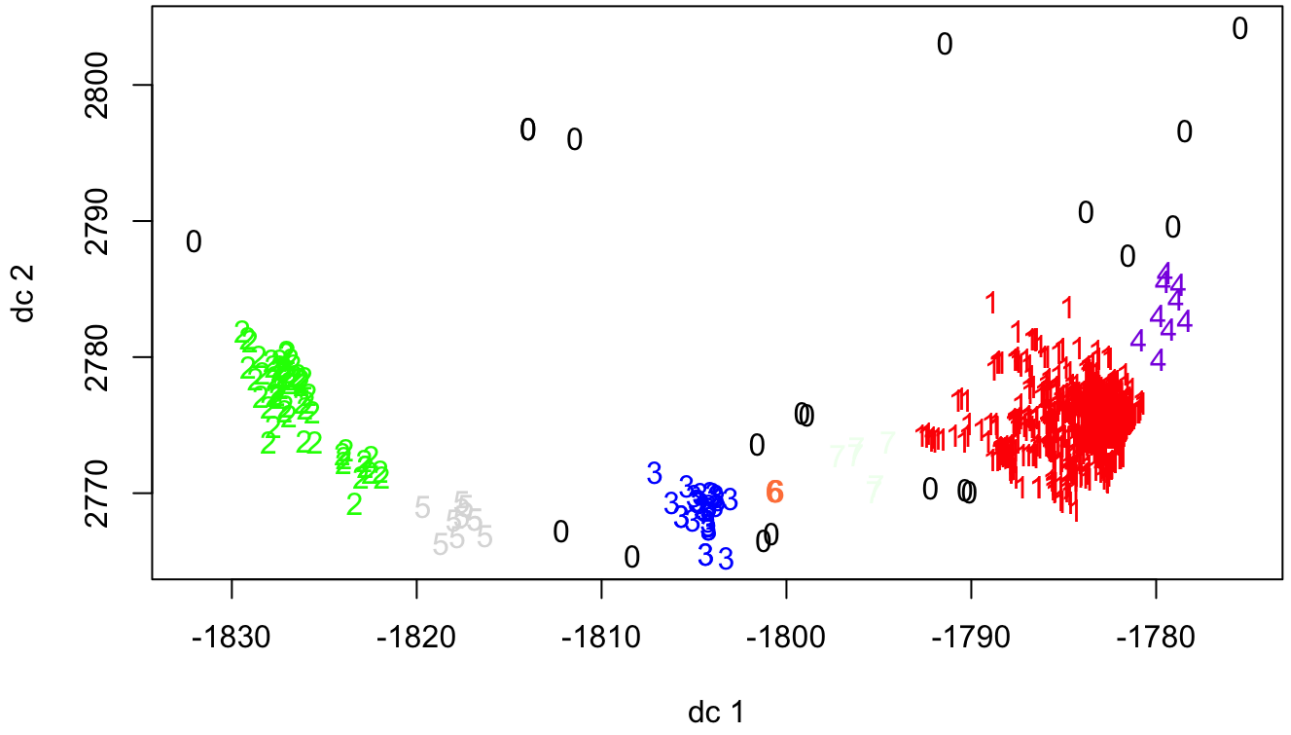
使用DBSCAN 分群

```
library(fpc)
res <- dbscan(Dengue,eps=0.1,MinPts=3)
table(res$cluster)
plot(res, Dengue)
```



繪製分群結果

```
plotcluster(Dengue,res$cluster)
```



THANK YOU