



R for Data Science

Bài 20: *KMeans*

Phòng LT & Mạng

https://csc.edu.vn/lap-trinh-va-cSDL/R-Programming-Language-for-Data-Science_190

2020



Nội dung



1. Cluster Analysis

2. KMeans



Cluster Analysis

❑ Bài toán phân nhóm toàn bộ dữ

liệu X thành các nhóm nhỏ dựa trên
sự liên quan giữa các dữ liệu trong
mỗi nhóm.

❑ Mục tiêu

- Tổ chức các item vào các nhóm



Cluster Analysis

❑ Ví dụ

- Phân nhóm khách hàng vào các nhóm
- Phân loại các mẫu thời tiết khác nhau cho một vùng
- Nhóm các bài viết/tin tức vào các chủ đề
- Khám phá các điểm nóng tội phạm



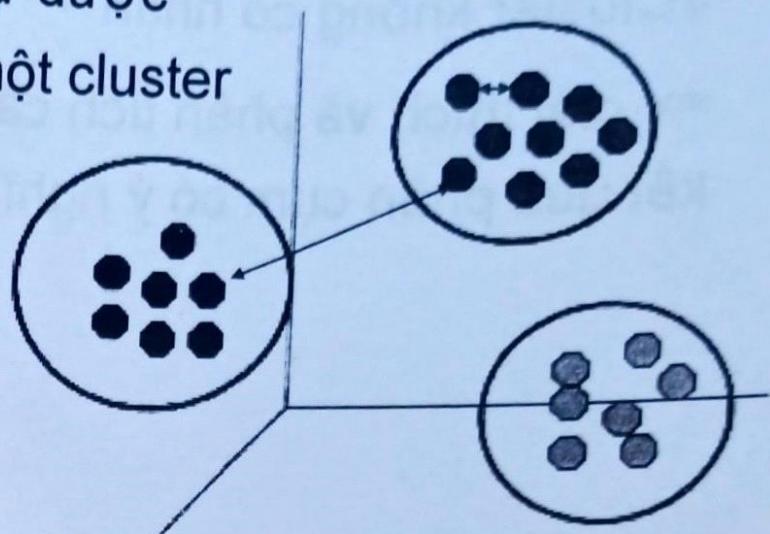
Cluster Analysis

□ Ý tưởng

- Chia dữ liệu vào các cluster
- Các item tương tự được nhóm vào cùng một cluster

Sự khác biệt của intra-cluster là tối thiểu

Sự khác biệt của inter-cluster là tối đa

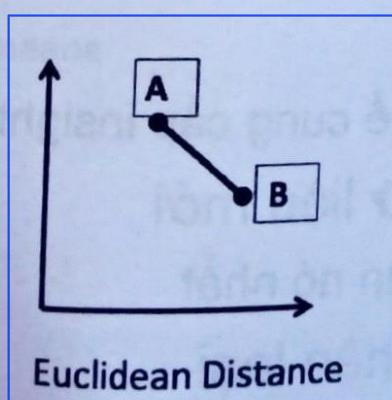


R programming language for Data Science

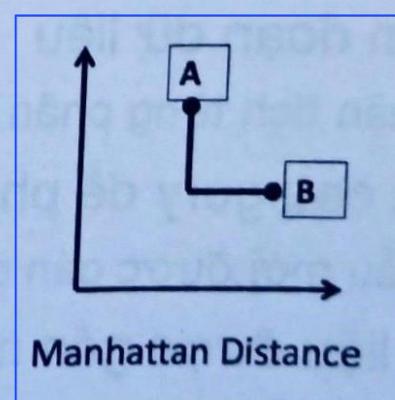
5

Cluster Analysis

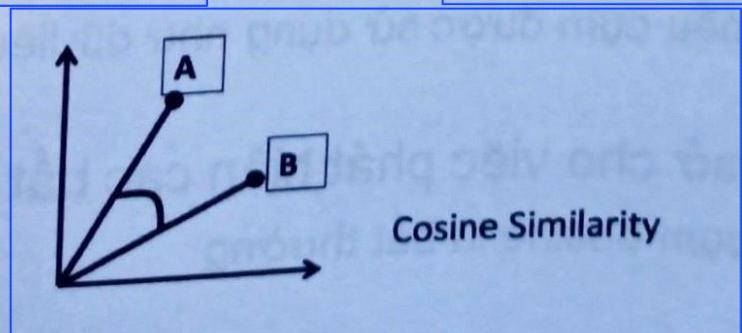
□ Cách tính khoảng cách



Euclidean Distance



Manhattan Distance



Cosine Similarity

R programming language for Data Science

6

Cluster Analysis

□ Ghi chú: trong Cluster Analysis

- Không có khái niệm cluster “đúng”
 - Cluster không có nhãn
- => giải thích và phân tích cần thiết để tạo các kết quả phân cụm có ý nghĩa.



Cluster Analysis

□ Sử dụng kết quả phân cụm

- Phân đoạn dữ liệu
 - Phân tích từng phân đoạn có thể cung cấp insight
- Các category để phân loại dữ liệu mới
 - Mẫu mới được gán cho cụm gần nó nhất
- Dữ liệu được gán nhãn để phân loại
 - Các mẫu cụm được sử dụng như dữ liệu có gán nhãn
- **Là cơ sở cho việc phát hiện các bất thường**
 - Các cụm outline là bất thường



Một số thuật toán cluster

- K-Means clustering
- Hierarchical clustering
- ...

Nội dung

1. Cluster Analysis

2. Kmeans

K-Means

□ Giới thiệu

- K-Means là một thuật toán thuộc nhóm Unsupervised Learning.
- Được sử dụng khi ta có dữ liệu không có nhãn



K-Means

□ Ý tưởng

- Tìm các nhóm trong dữ liệu với số lượng nhóm được đại diện bởi biến K
- Thuật toán được lặp lại để gán mỗi điểm dữ liệu vào một trong các nhóm K dựa trên các tính năng được cung cấp
- Các điểm dữ liệu được phân cụm dựa trên sự tương tự về tính năng



K-Means

□ Ý tưởng

- Kết quả của thuật toán K-Means:

- Các trung tâm centroid của các cụm K có thể được sử dụng để gán nhãn dữ liệu mới
- Các nhãn cho training data: mỗi điểm dữ liệu được gán cho một cụm duy nhất



K-Means

□ K-Means Algorithm

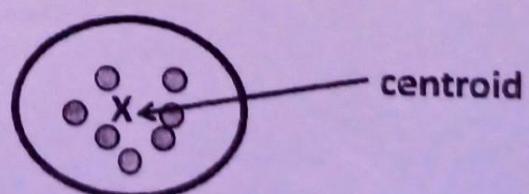
- Lựa chọn k centroid ban đầu (cluster center)

- Lặp:

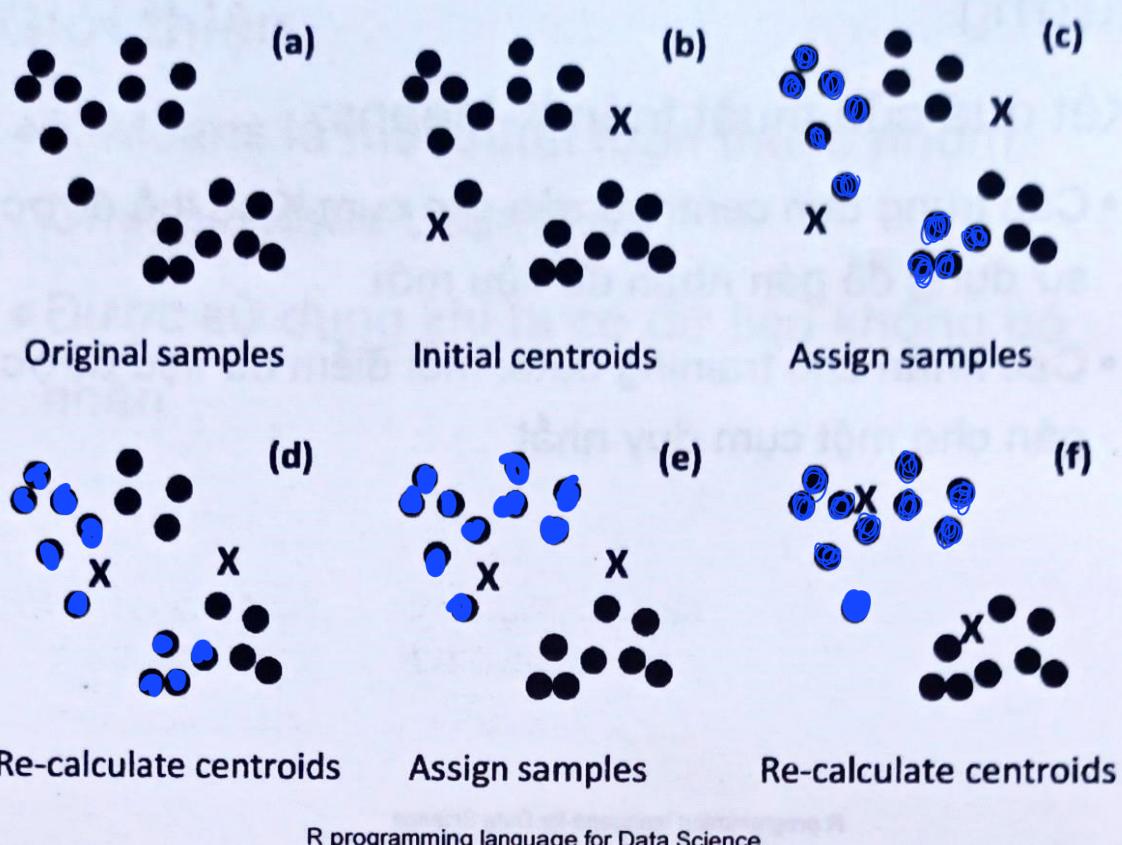
- Gán từng mẫu với centroid gần nhất
- Tính toán trung bình của các cluster để xác định centroid mới

- Cho đến khi:

- Đạt được tiêu chí dừng



K-Means



R programming language for Data Science

15

K-Means

- Chọn các centroid ban đầu
 - Vấn đề: các cluster cuối cùng thường ‘nhạy cảm’ với các centroid ban đầu
 - Giải pháp: chạy k-means nhiều lần với các centroid ban đầu ngẫu nhiên khác nhau => chọn kết quả tốt nhất



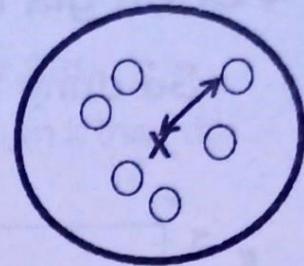
R programming language for Data Science

16

K-Means

- **Tính toán kết quả cluster**

- error = khoảng cách giữa sample và centroid



- squared error = error^2
- Tính tổng các squared error giữa tất cả các sample và centroid

=> Tính tổng trên tất cả các cluster => WSSE

Within-Cluster Sum of Squared Error



K-Means

- **Sử dụng WSSE**

- WSSE1 < WSSE2 => WSSE1 là số tốt hơn

- Chú ý:

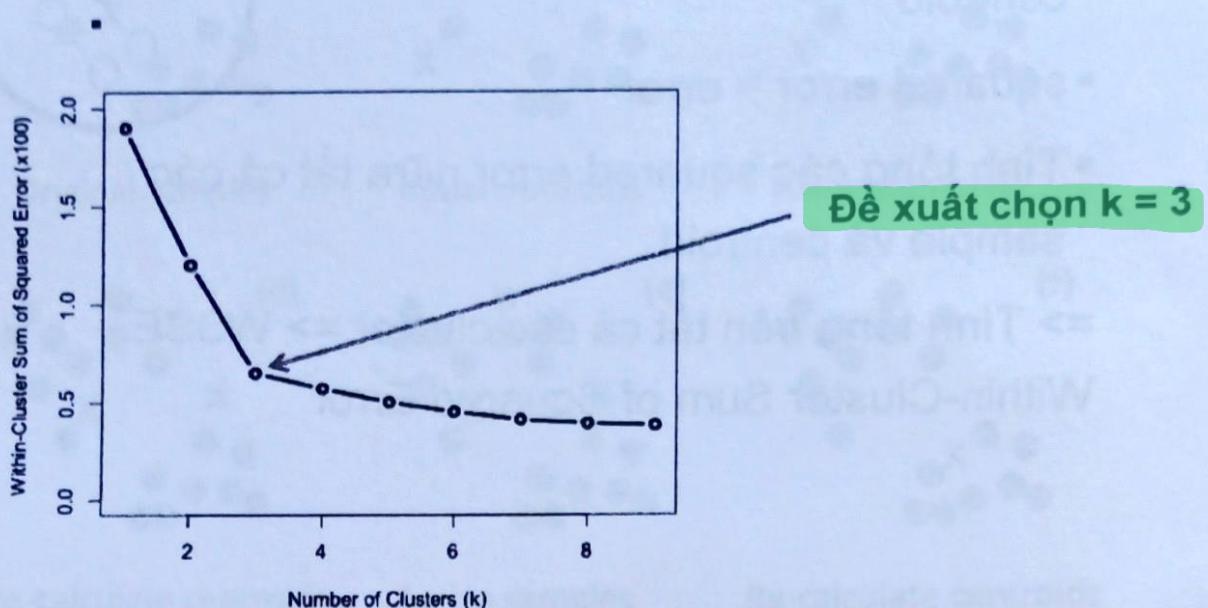
- Điều này không có nghĩa là bộ cluster 1 “đúng” hơn bộ cluster 2
- Giá trị lớn hơn cho k sẽ luôn giảm WSSE



K-Means

- Chọn giá trị cho k: k=?

- Sử dụng Elbow Method để chọn k



R programming language for Data Science

19

K-Means

- Elbow Method

Tư tưởng chính của phương pháp K-Means là định nghĩa 1 cụm sao cho tổng biến thiên bình phương khoảng cách trong cụm là nhỏ nhất, tham số này là WSS (Within-cluster Sum of Square)

Elbow method chọn số cụm k sao cho khi thêm vào một cụm khác thì không làm cho WSS thay đổi nhiều.

K-Means

• Elbow Method

- Qui trình triển khai Elbow method như sau:

- Triển khai thuật toán k-means với các số cụm k thay đổi (ví dụ từ 1 đến 10)
- Với mỗi giá trị k, tính giá trị WSS
- Vẽ Elbow curve theo các giá trị k.
- Dựa vào Elbow curve chọn số k thích hợp, là vị trí ở khúc cua (bend|knee)



K-Means

• Điều kiện dừng

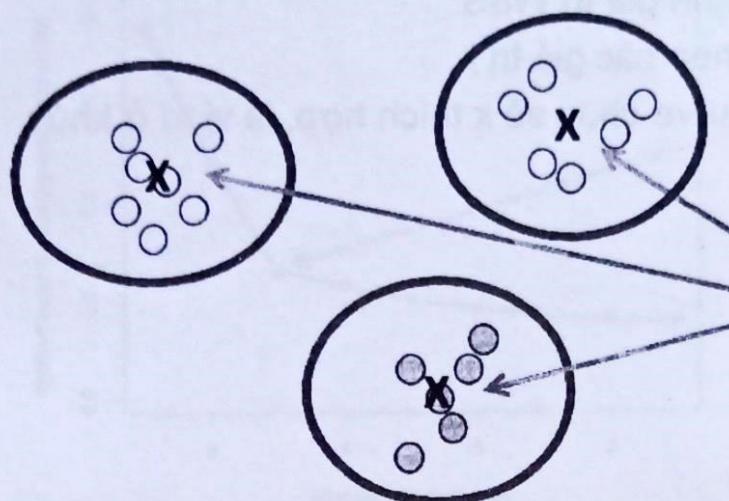
- Việc lặp sẽ dừng khi không có sự thay đổi các centroid
- Số lượng các mẫu thay đổi cluster dưới ngưỡng (threshold)



K-Means

- **Diễn giải kết quả**

- Kiểm tra các cluster centroid
 - Các cụm khác nhau như thế nào?



**So sánh các centroid
để thấy sự khác biệt
của các cluster**



K-Means

- **Tổng kết**

- Thuật toán K-Means để phân tích cluster
- Đơn giản, dễ hiểu và thực hiện có hiệu quả
- Giá trị của k cần phải được xác định
- Các cluster cuối cùng thường ‘nhạy cảm’
với các centroid ban đầu



K-Means

□ Ưu điểm

- Dễ thực hiện
- K-Means có thể tạo ra các cụm chặt chẽ
- Một thực thể có thể làm thay đổi cluster (di chuyển qua cluster khác) khi centroid được tính lại.



K-Means

□ Hạn chế

- Khó khăn khi dự đoán số lượng cluster (giá trị K)
- Các phân vùng ban đầu khác nhau có thể dẫn đến các cụm cuối cùng khác nhau
- Thứ tự của dữ liệu có tác động đến kết quả cuối cùng
- 'Nhạy cảm' với việc thay đổi kích cỡ: thay đổi bộ dữ liệu (chuẩn hóa) sẽ làm thay đổi kết quả hoàn toàn.
- Hoạt động không tốt với các cụm (trong dữ liệu gốc) có kích thước và mật độ khác nhau



K-Means

❑ Các bước thực hiện

- Tìm hiểu về dataset (Thu thập các số liệu quan sát, tiền xử lý, trực quan hóa dữ liệu)
- Tìm k
- Xây dựng mô hình (model) sử dụng kmeans(data, centers = k, nstart = 20)
- In kết quả của model
- Trực quan hóa kết quả



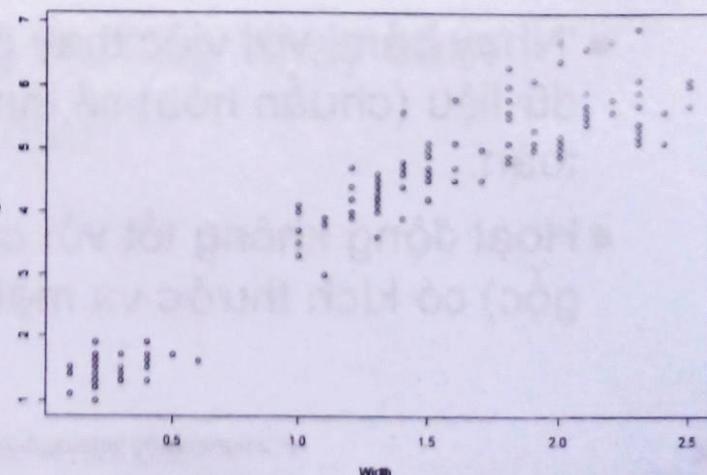
K-Means

- Tìm hiểu về dataset (Thu thập các số liệu quan sát, tiền xử lý, trực quan hóa dữ liệu)

```
library(datasets)
head(iris)
tail(iris)

# Plot the chart
plot(x = iris$Petal.Width, y = iris$Petal.Length,
      xlab = "width",
      ylab = "Length",
      main = "Iris Petal.Length vs Petal.Width", col = iris$Species)
```

```
Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5        1.4       0.2   setosa
2          4.9         3.0        1.4       0.2   setosa
3          4.7         3.2        1.3       0.2   setosa
4          4.6         3.1        1.5       0.2   setosa
5          5.0         3.6        1.4       0.2   setosa
6          5.4         3.9        1.7       0.4   setosa
> tail(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
145         6.7         3.3        5.7       2.5 virginica
146         6.7         3.0        5.2       2.3 virginica
147         6.3         2.5        5.0       1.9 virginica
148         6.5         3.0        5.2       2.0 virginica
149         6.2         3.4        5.4       2.3 virginica
150         5.9         3.0        5.1       1.8 virginica
```

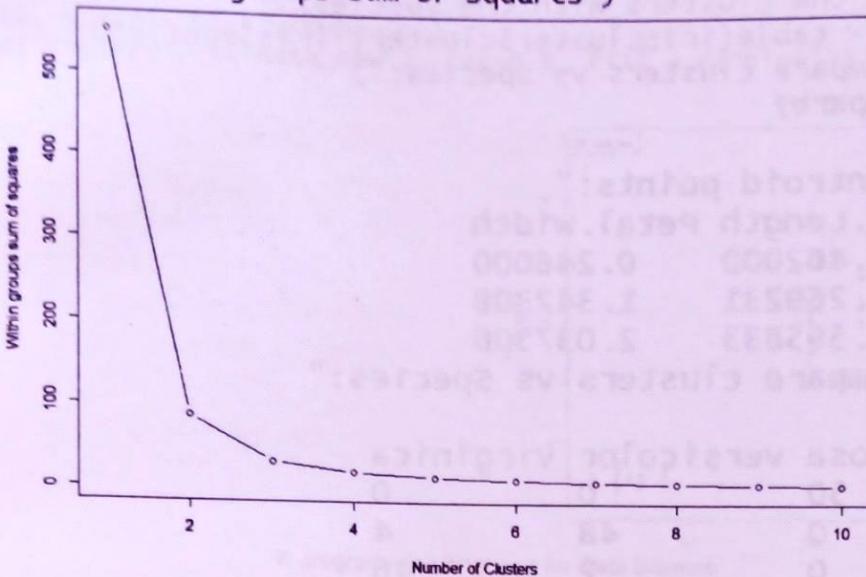


K-Means



- ### ▪ Tìm k

```
* Determine number of clusters
mydata<-iris
wss <- (nrow(mydata)-1)*sum(apply(mydata,2,var))
for (i in 1:10) wss[i] <- sum(kmeans(mydata[, 3:4],
                                         centers=i, nstart = 20)$withinss)
plot(1:10, wss, type="b", xlab="Number of Clusters",
      ylab="within groups sum of squares")
```



K-Means



- Xây dựng mô hình (model) sử dụng kmeans(data, centers = k, nstart = 20)

- #### ▪ In kết quả của model

K-Means

- Trung tâm cụm
- So sánh kết quả (nếu có)

```

print("Centroid points:")
print(iriscluster$centers)

# compare the clusters with the species.
compare <- table(iriscluster$cluster, iris$species)
print("Compare clusters vs Species:")
print(compare)

[1] "Centroid points:"
  Petal.Length Petal.Width
1      1.462000    0.246000
2      4.269231    1.342308
3      5.595833    2.037500
[1] "Compare clusters vs Species:"
```

	setosa	versicolor	virginica
1	50	0	0
2	0	48	4
3	0	2	46

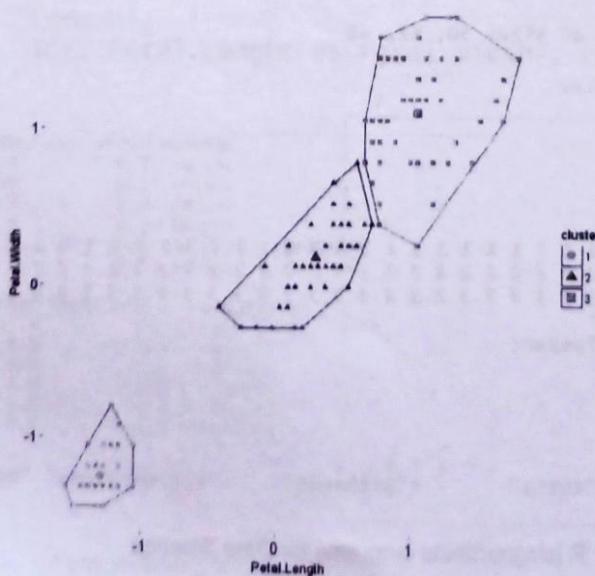


K-Means

- Trực quan hóa kết quả
- Cài install.packages("factoextra")

```

library(factoextra) # clustering algorithms & visualization
fviz_cluster(irisCluster, geom = "point", data = iris[, 3:4]) + ggtitle("k = 2")
  k = 3
```



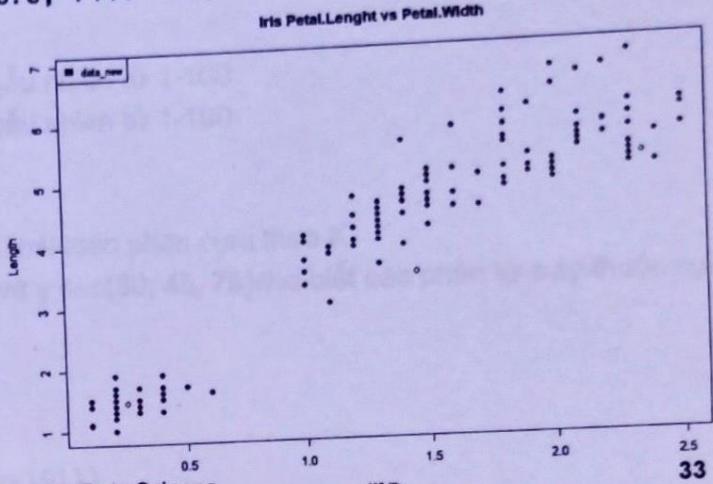
K-Means

- Trực quan hóa kết quả

```

x <- c(0.25, 1.45, 2.35)
y <- c(1.45, 3.45, 5.25)
iriscluster$cluster <- as.factor(irisCluster$cluster)
plot(x = iris$Petal.Width, y = iris$Petal.Length,
     xlab = "width",
     ylab = "Length",
     main = "Iris Petal.Length vs Petal.Width", col = irisCluster$cluster,
     pch = 19
)
lines(x, y, col='blue', type='p')
legend("topleft", c("data_new"), cex=0.8, fill = c("blue"))

```



R programming language for Data Science



Chapter 20: KMeans

Exercise 1: Random data

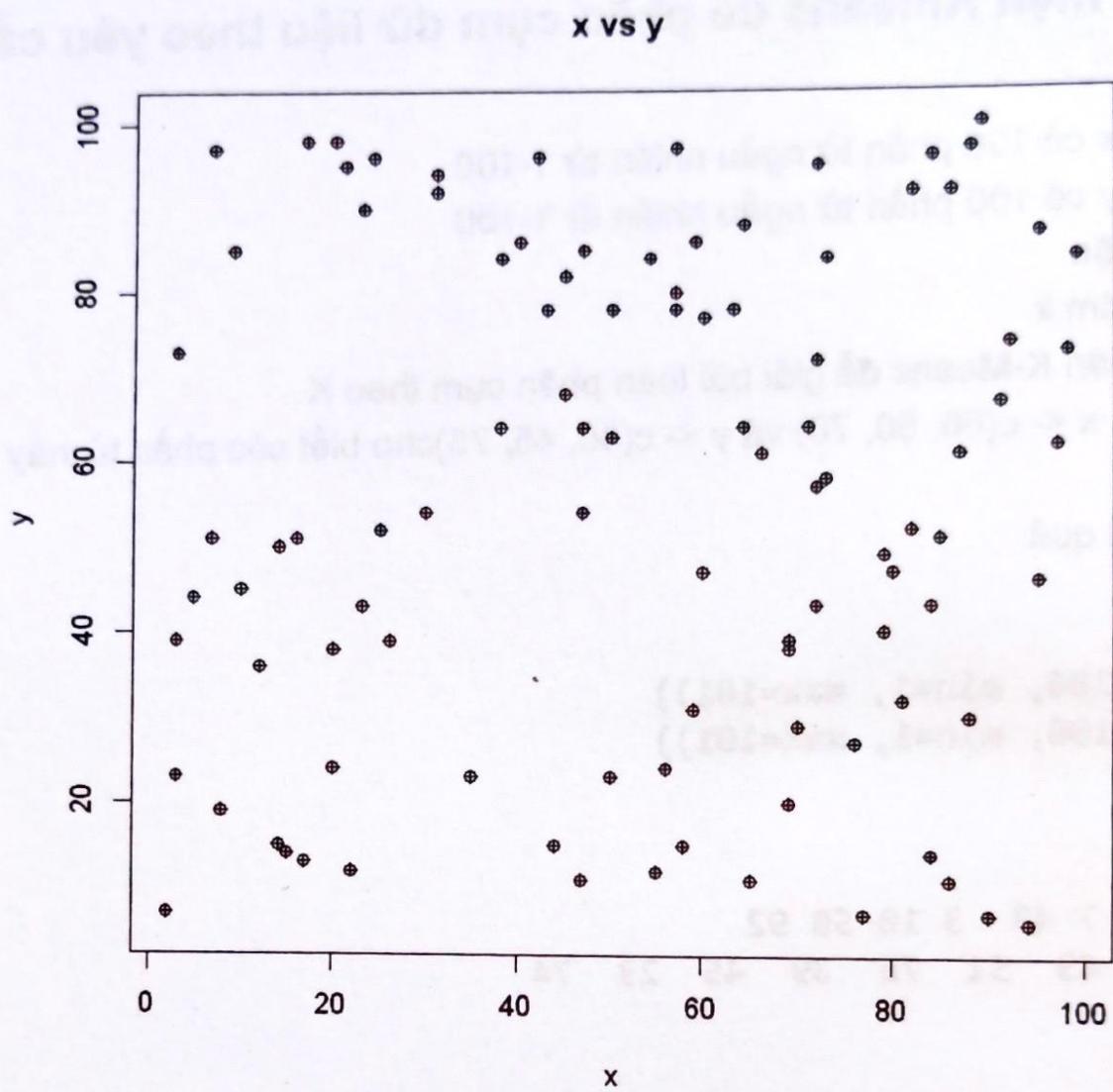
Yêu cầu: Thực hiện Kmeans để phân cụm dữ liệu theo yêu cầu sau:

- Tạo ra 1 vector x có 100 phần tử ngẫu nhiên từ 1-100
- Tạo ra 1 vector y có 100 phần tử ngẫu nhiên từ 1-100
- Chuẩn hóa dữ liệu
- Áp dụng Elbow tìm k
- Áp dụng thuật toán K-Means để giải bài toán phân cụm theo K
- Cho dữ liệu test: $x <- c(80, 50, 70)$ và $y <- c(30, 45, 75)$ cho biết các phần tử này thuộc cụm nào?
- Vẽ hình, xem kết quả

```
In [1]: x <- floor(runif(100, min=1, max=101))
y <- floor(runif(100, min=1, max=101))
print(x[1:10])
print(y[1:10])
```

```
[1] 77 89 88 23 7 43 3 10 50 92
[1] 7 100 30 43 51 78 39 45 23 74
```

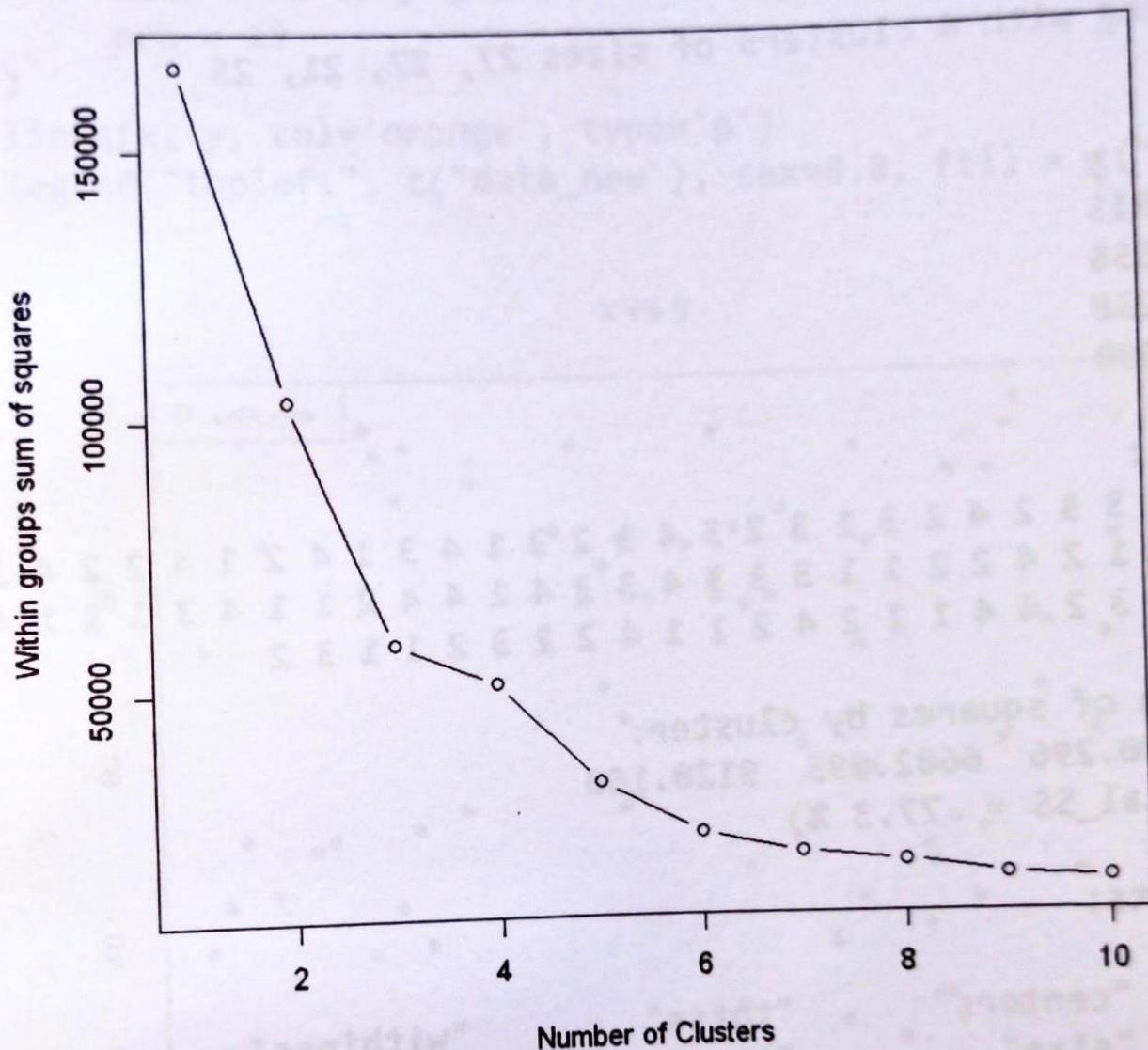
```
In [2]: # Plot the chart  
plot(x = x, y = y,  
      xlab = "x",  
      ylab = "y",  
      main = "x vs y",  
      pch = 10  
)
```



```
In [3]: # Determine number of clusters  
mydata<-data.frame(x = x, y = y)  
  
print(head(mydata))  
wss <- (nrow(mydata)-1)*sum(apply(mydata,2,var))  
for (i in 1:10) wss[i] <- sum(kmeans(mydata,  
                                      centers=i)$withinss)
```

	x	y
1	77	7
2	89	100
3	88	30
4	23	43
5	7	51
6	43	78

```
In [4]: plot(1:10, wss, type="b", xlab="Number of Clusters",  
         ylab="Within groups sum of squares")
```



```
In [5]: # clustering
set.seed(20)
dataCluster <- kmeans(mydata, centers = 4, nstart = 20)
print(dataCluster)

K-means clustering with 4 clusters of sizes 27, 27, 21, 25

Cluster means:
      x       y
1 37.11111 83.14815
2 71.70370 26.55556
3 15.57143 32.95238
4 80.20000 75.44000

Clustering vector:
 [1] 2 4 2 3 3 1 3 3 2 4 2 3 1 3 2 3 4 1 2 3 1 4 3 1 4 2 1 4 2 2 4 3 4 1 2 1 2
[38] 4 1 1 2 2 1 1 2 4 2 2 1 1 3 3 3 4 3 3 4 1 4 4 2 1 1 4 3 1 1 1 4 4 2 4 4 3
[75] 3 1 4 4 2 2 3 2 4 4 1 1 2 4 2 1 1 4 2 2 3 2 1 1 3 2

Within cluster sum of squares by cluster:
[1] 11008.074 10790.296 6602.095 9120.160
(between_SS / total_SS = 77.3 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"          "ifault"
```

```
In [6]: print("Centroid points:")
print(dataCluster$centers)

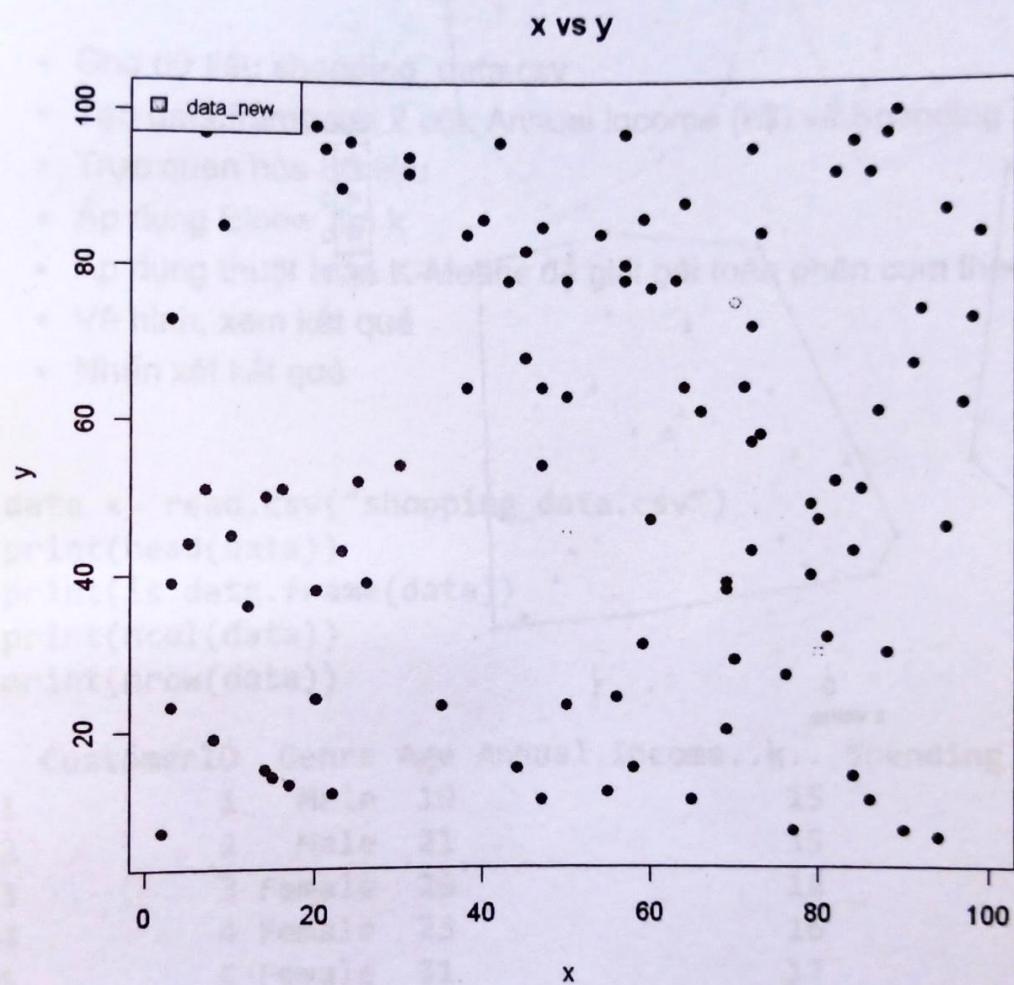
[1] "Centroid points:"
      x       y
1 37.11111 83.14815
2 71.70370 26.55556
3 15.57143 32.95238
4 80.20000 75.44000

In [7]: # Plot the chart
x <- c(80, 50, 70)
y <- c(30, 45, 75)
data_new <- data.frame(x = x, y = y)

clusters <- function(x, centers) {
  # compute squared euclidean distance from each sample to each cluster center
  tmp <- sapply(seq_len(nrow(x)),
               function(i) apply(centers, 1,
                                 function(v) sum((x[i, ]-v)^2)))
  max.col(-t(tmp)) # find index of min distance
}
new <- clusters(data_new, dataCluster[["centers"]])
new

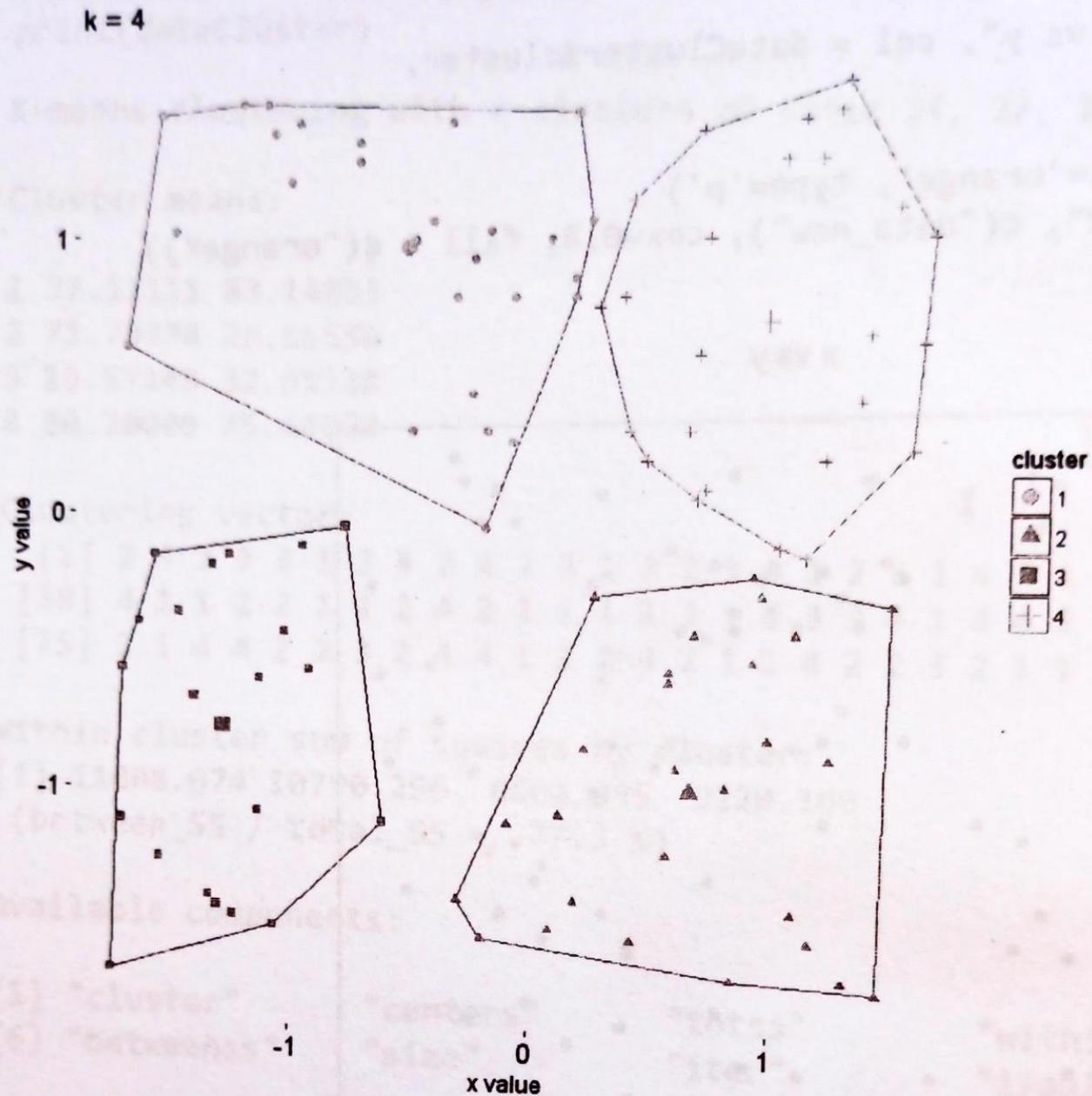
2 2 4
```

```
In [8]: dataCluster$cluster <- as.factor(dataCluster$cluster)
plot(x = mydata$x, y = mydata$y,
      xlab = "x",
      ylab = "y",
      main = "x vs y", col = dataCluster$cluster,
      pch = 19
)
lines(x, y, col='orange', type='p')
legend("topleft", c("data_new"), cex=0.8, fill = c("orange"))
```



```
In [9]: library(factoextra) # clustering algorithms & visualization
Loading required package: ggplot2
```

```
In [10]: fviz_cluster(dataCluster, geom = "point", data = mydata) +  
ggttitle("k = 4")
```



Chapter 20: KMeans

Exercise 2: Shopping

Yêu cầu: Thực hiện Kmeans để phân cụm dữ liệu theo yêu cầu sau:

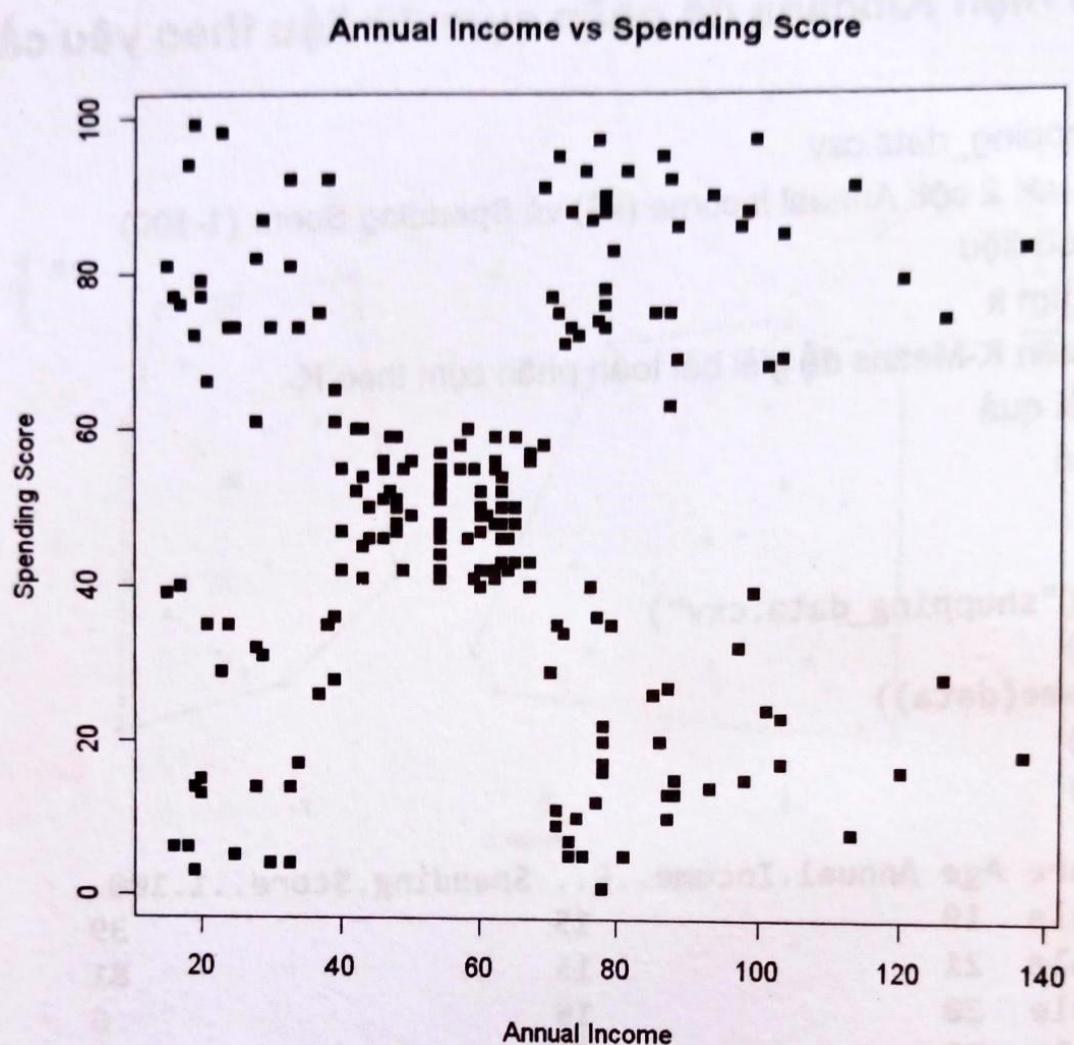
- Cho dữ liệu shopping_data.csv
- Tạo data.frame với 2 cột: Annual Income (k\$) và Spending Score (1-100)
- Trực quan hóa dữ liệu
- Áp dụng Elbow tìm k
- Áp dụng thuật toán K-Means để giải bài toán phân cụm theo K
- Vẽ hình, xem kết quả
- Nhận xét kết quả

```
In [1]: data <- read.csv("shopping_data.csv")
print(head(data))
print(is.data.frame(data))
print(ncol(data))
print(nrow(data))
```

	CustomerID	Genre	Age	Annual.Income..k..	Spending.Score..1.100.
1		1 Male	19	15	39
2		2 Male	21	15	81
3		3 Female	20	16	6
4		4 Female	23	16	77
5		5 Female	31	17	40
6		6 Female	22	17	76

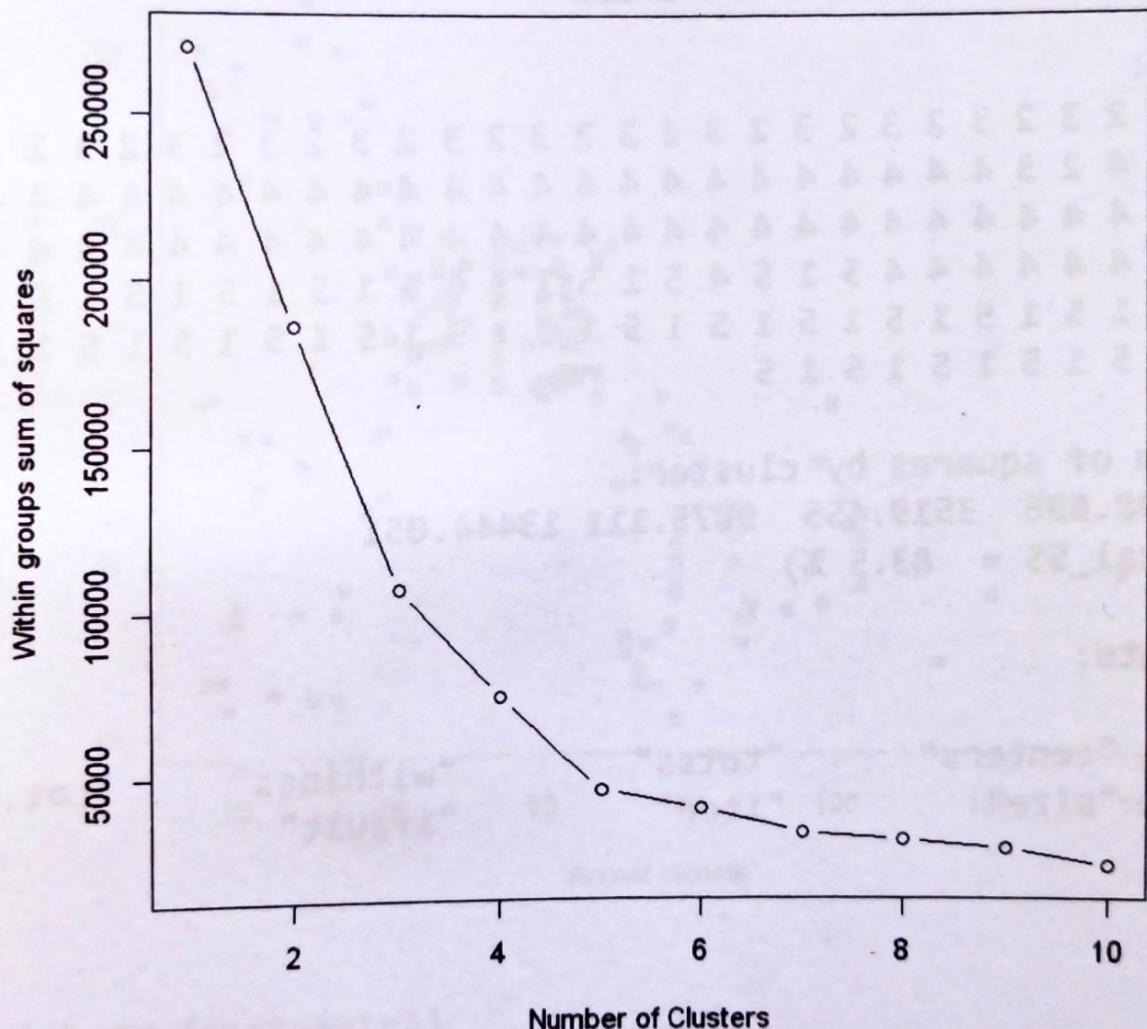
```
[1] TRUE
[1] 5
[1] 200
```

```
In [2]: # Plot the chart
plot(x = data$Annual.Income..k..,y = data$Spending.Score..1.100.,
      xlab = "Annual Income",
      ylab = "Spending Score",
      main = "Annual Income vs Spending Score",
      pch = 15, col = "blue")
)
```



```
In [3]: # finding k  
# Determine number of clusters  
mydata<-data  
wss <- (nrow(mydata)-1)*sum(apply(mydata,2,var))  
for (i in 1:10) wss[i] <- sum(kmeans(mydata[, 4:5],  
                                centers=i)$withinss)  
plot(1:10, wss, type="b", xlab="Number of Clusters",  
     ylab="Within groups sum of squares")
```

Warning message in FUN(newX[, i], ...):
"NAs introduced by coercion"





```
In [4]: # clustering
    set.seed(20)
    dataCluster <- kmeans(mydata[, 4:5], 5, nstart = 20)
    dataCluster
```

K-means clustering with 5 clusters of sizes 35, 23, 22, 81, 39

Cluster means:

	Annual.Income..k..	Spending.Score..1.100.
1	88.20000	17.11429
2	26.30435	20.91304
3	25.72727	79.36364
4	55.29630	49.51852
5	86.53846	82.12821

Clustering vector:

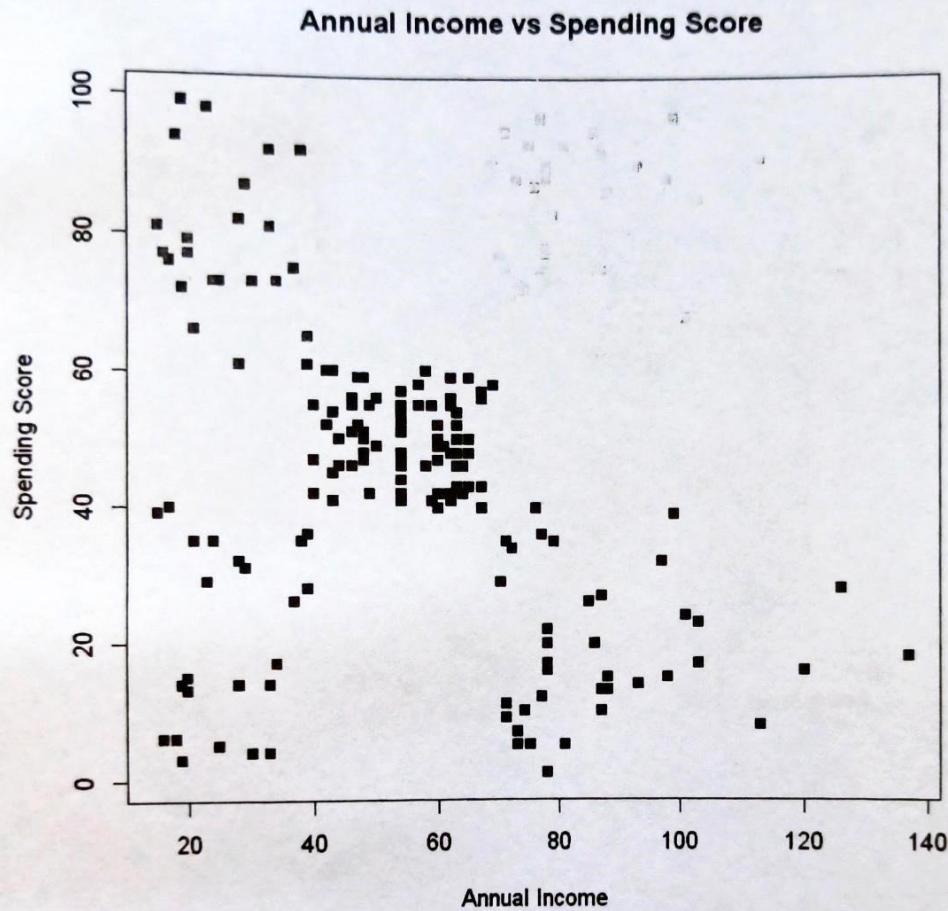
Within cluster sum of squares by cluster:

[1] 12511.143 5098.696 3519.455 9875.111 13444.051
(between SS / total SS = 83.5 %)

Available components:

```
[1] "cluster"      "centers"       "totss"        "withinss"      "tot.withinss"  
[6] "betweenss"    "size"          "iter"         "ifault"
```

In [5]: # Plot the chart
dataCluster\$cluster <- as.factor(dataCluster\$cluster)
plot(x = mydata\$Annual.Income..k..,y = mydata\$Spending.Score..1.100.,
 xlab = "Annual Income",
 ylab = "Spending Score",
 main = "Annual Income vs Spending Score", col = dataCluster\$cluster,
 pch = 15
)



In [6]: library(factoextra)

Loading required package: ggplot2

```
In [7]: fviz_cluster(dataCluster, geom = "point", data = mydata[, 4:5]) +  
ggttitle("k = 5")
```

