Chapter 20: KMeans

Exercise 1: Random data

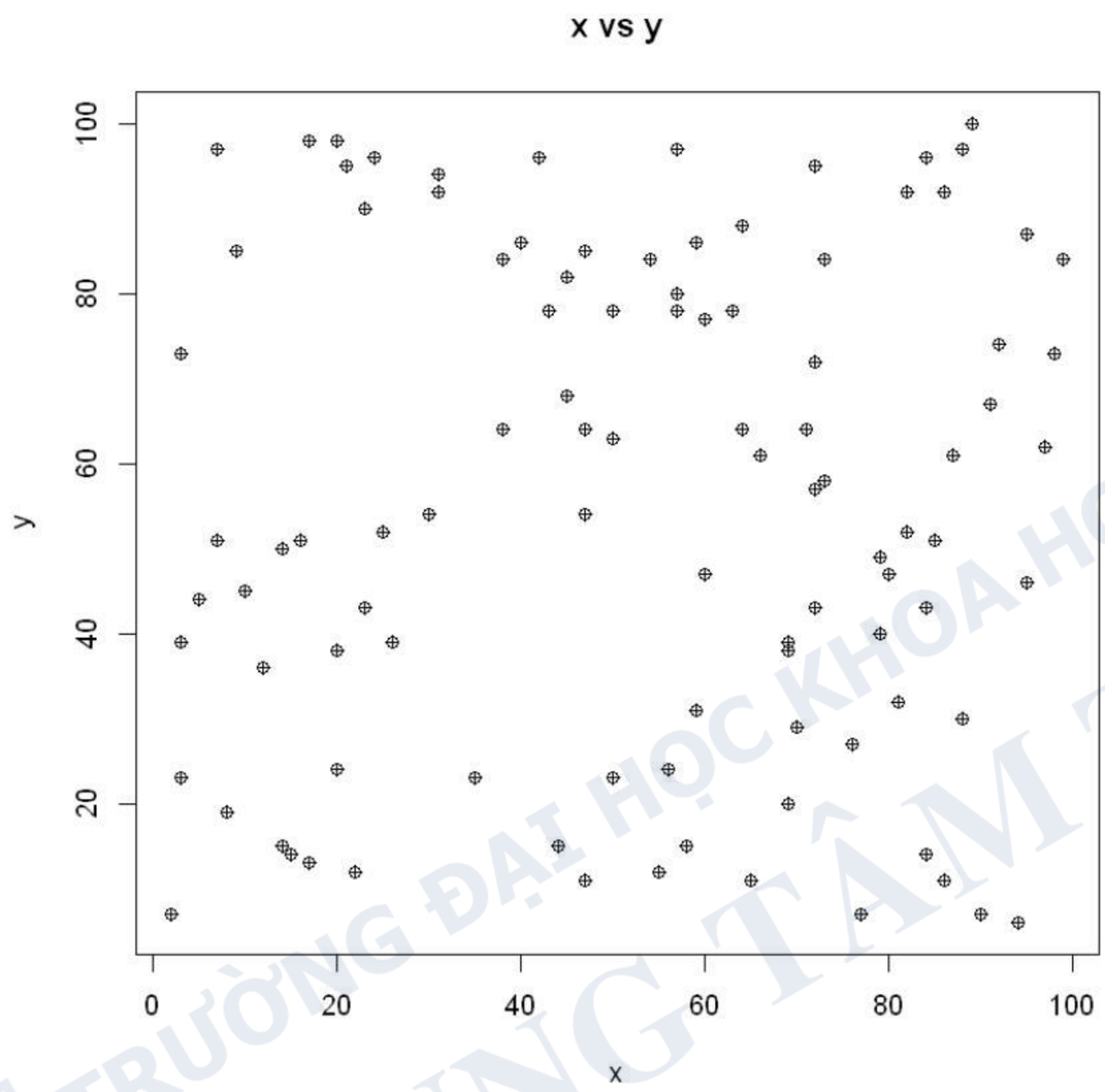
Yêu cầu: Thực hiện Kmeans để phân cụm dữ liệu theo yêu cầu sau:

- 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
)
```

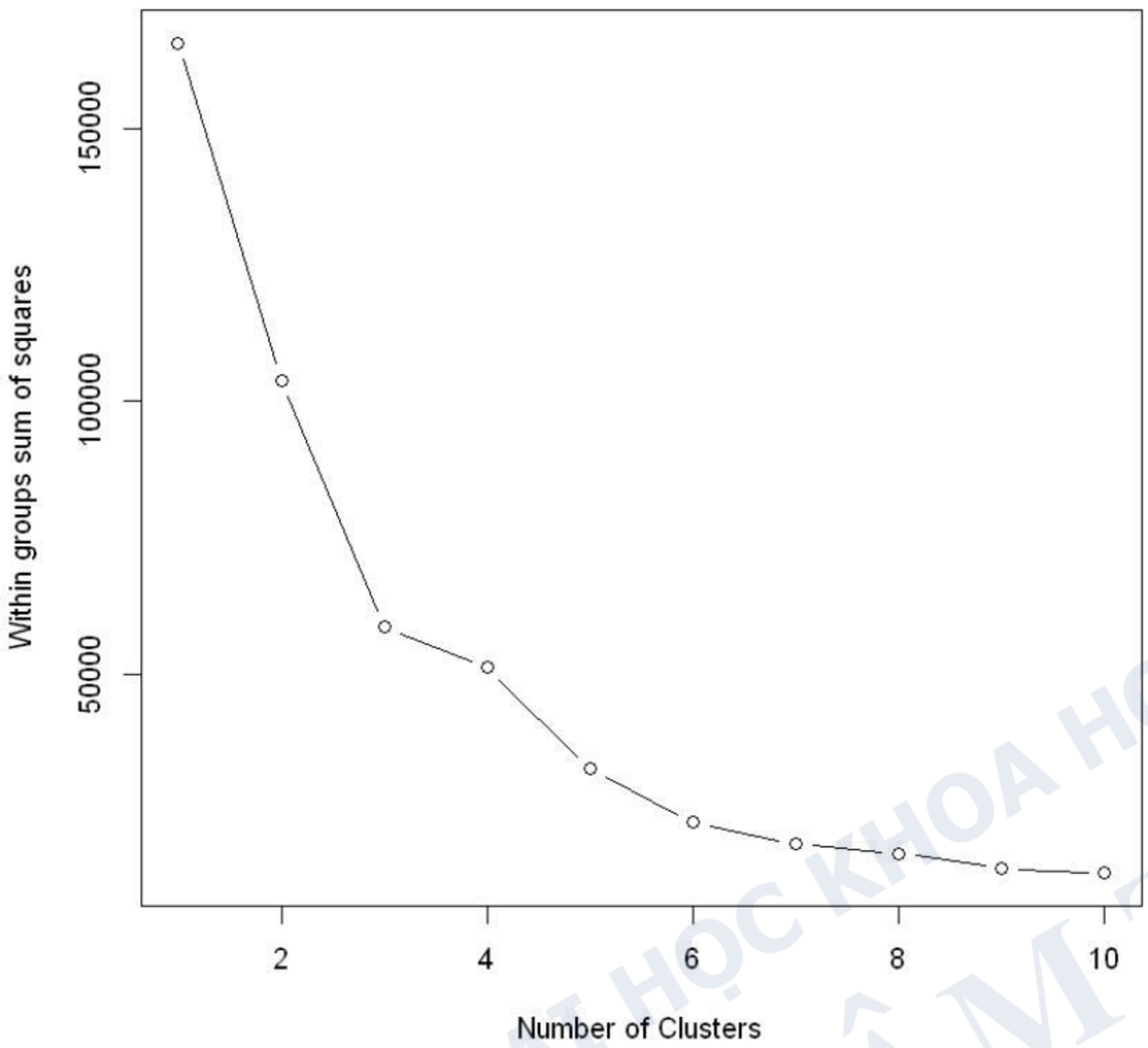


In [3]:
```
# Determine number of clusters
mydata<-data.frame(x = x, y = y)

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)
```

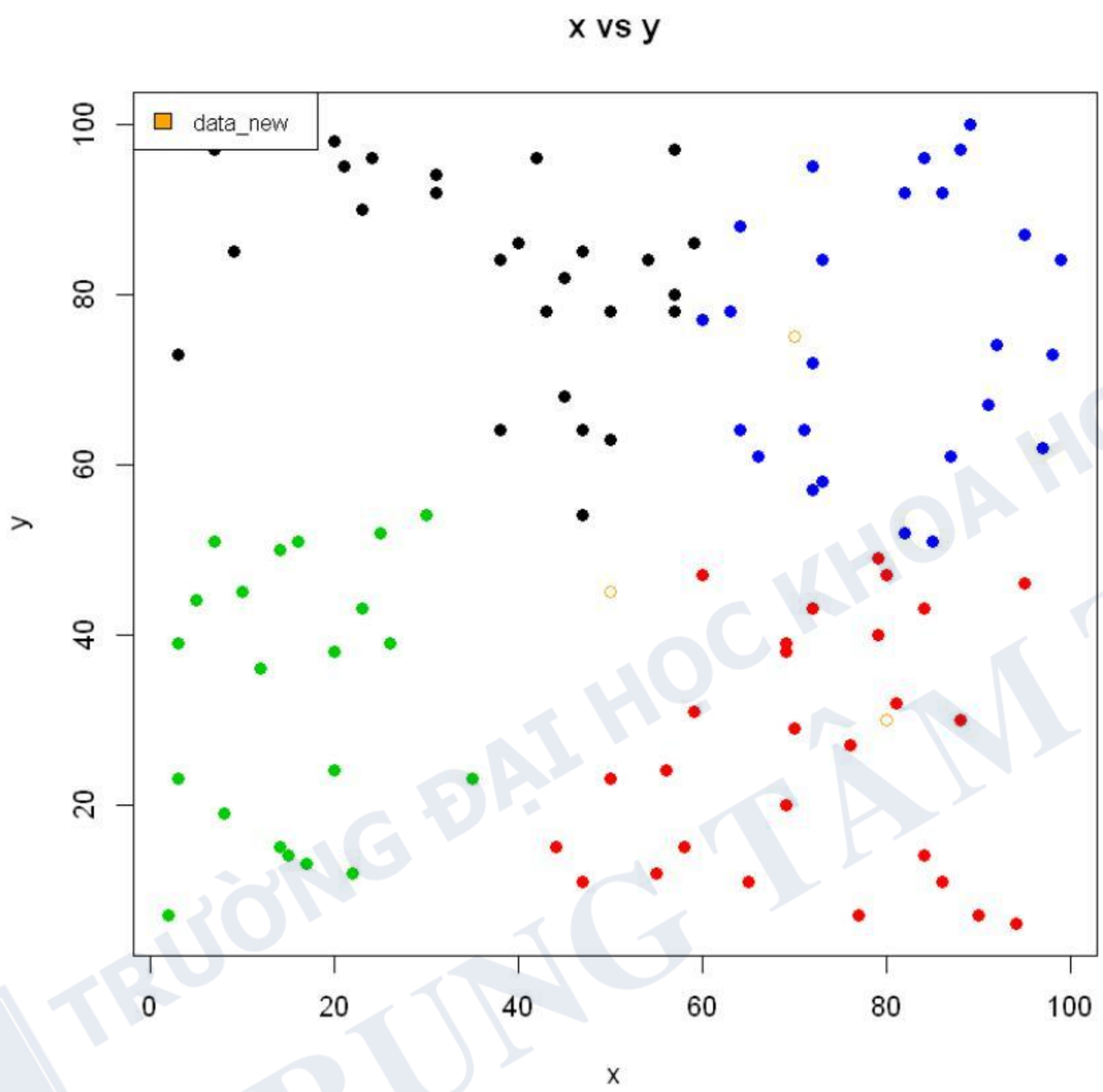
```
[1] "Centroid points:"
          x        y
1 37.11111 83.14815
2 71.70370 26.55556
3 15.57143 32.95238
4 80.20000 75.44000
```

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
```

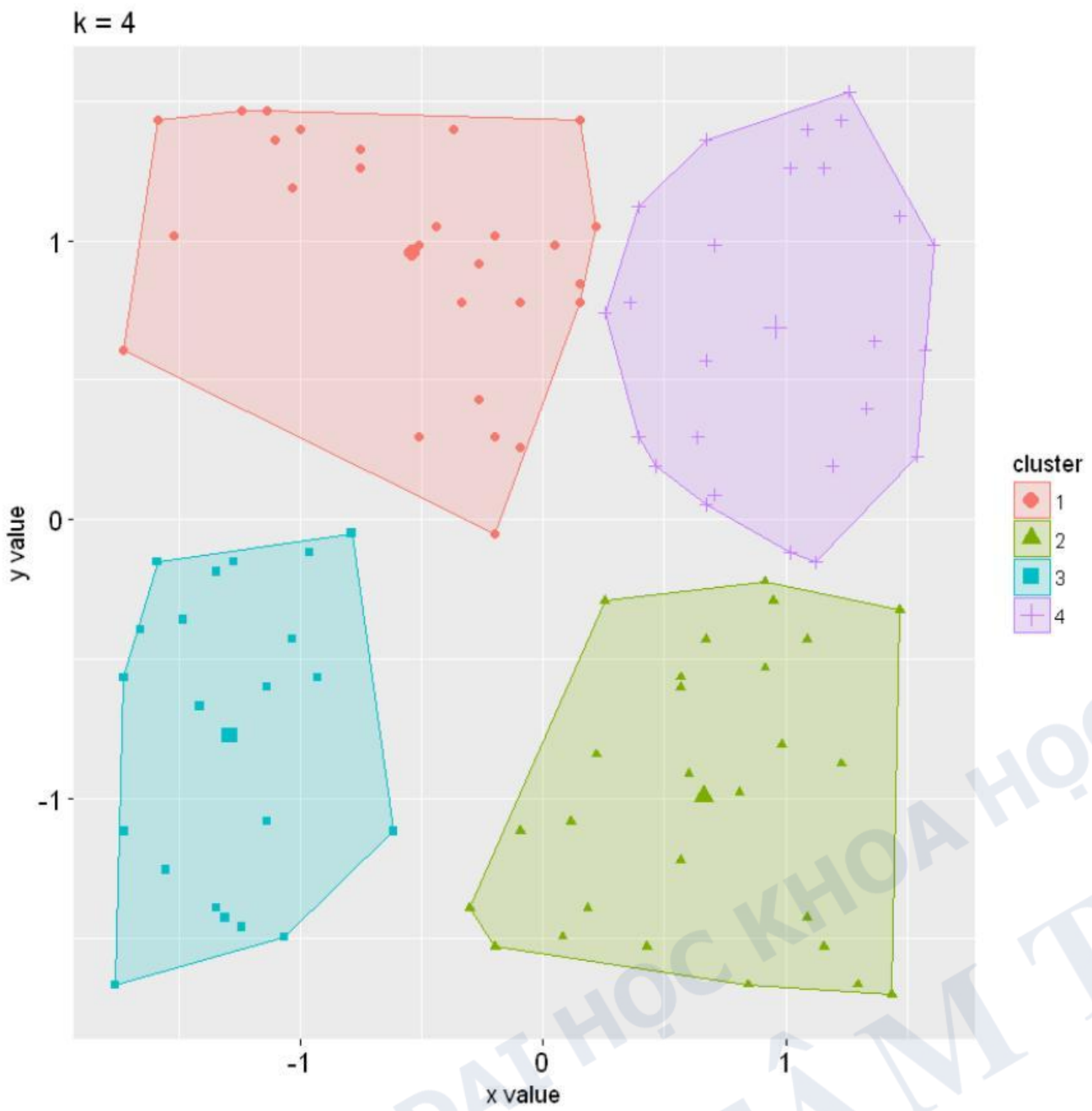
In [8]:
```
dataCluster$cluster <- as.factor(dataCluster$cluster)
plot(x = mydata$x, y = mydata$y,
     xlab = "x",
     ylab = "y",
     main = "x vs y", col = dataCluster$cluster,
     pch = 19
)
lines(x, y, col='orange', type='p')
legend("topleft", c("data_new"), cex=0.8, fill = c("orange"))
```



In [9]:
```
library(factoextra) # clustering algorithms & visualization
```

```
Loading required package: ggplot2
```

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



In [ ]:
```

```