Mohamed Rahouti

Unsupervised Machine Learning Lab

K-Means Clustering

A. Dataset overview:

- In this lab we are going to use an existing dataset from the 'Pokemon' game. More detailed information about Pokemon and dataset features can be found at the link: https://pokemondb.net/pokedex

```
> head(pokemon)
     HitPoints Attack Defense SpecialAttack SpecialDefense Speed
[1,]        45     49      49            65            65    45
[2,]        60     62      63            80            80    60
[3,]        80     82      83           100           100    80
[4,]        80    100     123           122           120    80
[5,]        39     52      43            60            50    65
[6,]        58     64      58            80            65    80
```

B. Overview of K-means clustering:

- An algorithm used to find homogeneous subgroups in a population
- K-means comes in base R
    - Need the data
    - Number of centers or groups
    - Number of runs. its start by randomly assigning points to groups and you can find local minimums so running it multiple times helps you find the global min.
- You can run k-means many times to estimate the number of subgroups when it is not known a priori

C. Let's practice: First hands-on exercise

1. Read the provided data file. You need to specify the correct path to the location of your dataset file.
   > data_set<-read.csv("C:/Pokemon.csv")

2. Create and initialize a variable x that will contain column data from the Pokemon dataset, Let's assign the observations for Hit Point (HP) to x.
   > x <- data_set$HP
   > x

3. Create the K-means model, called km.out
   > km.out <- kmeans(x, centers = 3, nstart = 20)

4. Inspect the output
   > summary(km.out)

```
> summary(km.out)
             Length Class  Mode
cluster      800    -none- numeric
centers        3    -none- numeric
totss          1    -none- numeric
withinss       3    -none- numeric
tot.withinss   1    -none- numeric
betweenss      1    -none- numeric
size           3    -none- numeric
iter           1    -none- numeric
ifault         1    -none- numeric
> 
```

5. Print the cluster membership component of the model
   > km.out$cluster

```
> km.out$cluster
  [1] 2 3 3 3 2 2 3 3 3 2 3 3 3 2 2 3 2 2 3 3 2 3 3 3 2 2 2 3 2 3 2 3 2 3 3 2 3 3 3 1 2 3 1 1 2 3 2 3 3 2 3 3 3 2 2 2
 [59] 3 2 3 2 3 2 3 2 3 3 2 2 2 2 3 3 3 2 3 3 2 3 2 2 3 2 3 3 1 1 2 2 2 2 3 3 3 3 1 2 2 2 2 3 3 2 3 3 2 2 2 3 3 1 2 3 2 2
[117] 3 2 3 3 1 1 3 1 1 2 2 2 3 2 3 2 3 3 3 3 3 3 3 3 2 1 1 1 2 2 1 3 3 3 2 3 2 3 3 3 1 3 3 3 2 3 3 1 1 1 1 2 3 3 2 2 3 2 3
[175] 3 2 3 3 1 2 2 2 3 3 3 1 2 2 3 2 2 2 3 2 3 3 3 3 3 1 3 3 2 2 3 2 2 3 3 2 1 3 1 3 1 3 2 1 3 2 3 1 3 3 3 3 3 3 3 3 2 3
[233] 3 2 3 3 2 2 2 1 2 2 3 2 3 3 2 3 3 3 3 3 3 3 3 2 2 2 2 2 2 1 1 3 1 1 2 3 1 1 1 1 1 2 2 3 3 2 3 3 3 2 3 1 1 2 3 2 3 2 2
[291] 3 2 3 2 3 3 2 3 3 2 3 2 3 3 2 2 3 3 2 3 3 3 3 3 3 1 2 3 2 3 3 1 3 1 2 2 2 3 2 2 2 2 2 3 3 3 2 3 3 2 3 3 3 3 3 3 2 3 1 2
[349] 3 3 1 1 3 3 3 3 3 3 3 2 2 3 2 3 2 3 3 3 3 3 3 3 2 1 2 3 2 3 3 3 2 3 2 1 3 3 2 3 3 2 2 1 3 3 3 1 2 3 3 3 3 3 1 2 2 2 1 2
[407] 2 3 1 1 2 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 2 2 2 2 2 2 3 1 2 3 3 2 3 3 2 2 3 3 3 2 3 2 3 3 2 3 3 3 1 2 3 2 3 3 3 3 2 3
[465] 3 2 3 2 3 3 1 3 3 1 2 3 3 3 1 2 3 2 3 1 2 3 2 2 1 3 2 2 3 1 1 1 2 3 3 3 1 2 3 2 3 3 2 3 2 3 3 3 3 3 3 1 1 1 3 3 3 3 3
[523] 3 3 1 3 3 3 3 3 2 3 2 2 2 2 2 2 3 3 3 1 3 3 1 1 1 1 3 1 3 1 1 1 1 2 3 3 3 3 1 2 3 1 2 3 2 3 3 3 2 3 2 3 2 3 2 3 3 1 2 3
[581] 3 2 3 2 3 3 2 3 3 1 1 1 3 3 1 2 3 1 1 3 2 2 3 2 2 3 2 3 2 3 3 2 3 1 3 1 1 3 2 3 2 3 3 2 2 2 3 2 3 2 3 2 3 2 3 2 3 3
[639] 2 3 1 3 3 3 2 2 3 3 3 2 2 3 3 1 2 1 1 2 3 2 3 2 3 3 2 3 3 2 3 2 3 3 2 3 3 3 2 1 3 2 3 1 2 3 3 3 3 2 3 1 3 1 3 1 3 2 2 3
[697] 3 2 3 3 3 3 3 3 3 3 3 1 1 1 3 3 1 1 1 3 2 3 3 2 2 3 2 3 3 2 2 3 3 2 3 3 3 2 2 3 3 3 1 3 1 3 3 3 3 2 3 3 3 3 1
[755] 3 3 2 3 2 3 2 3 2 3 2 3 2 3 3 1 1 3 3 2 2 3 3 2 2 3 2 2 2 3 3 2 3 3 2 1 2 3 1 1 1 2 2 3 3 3
>
```

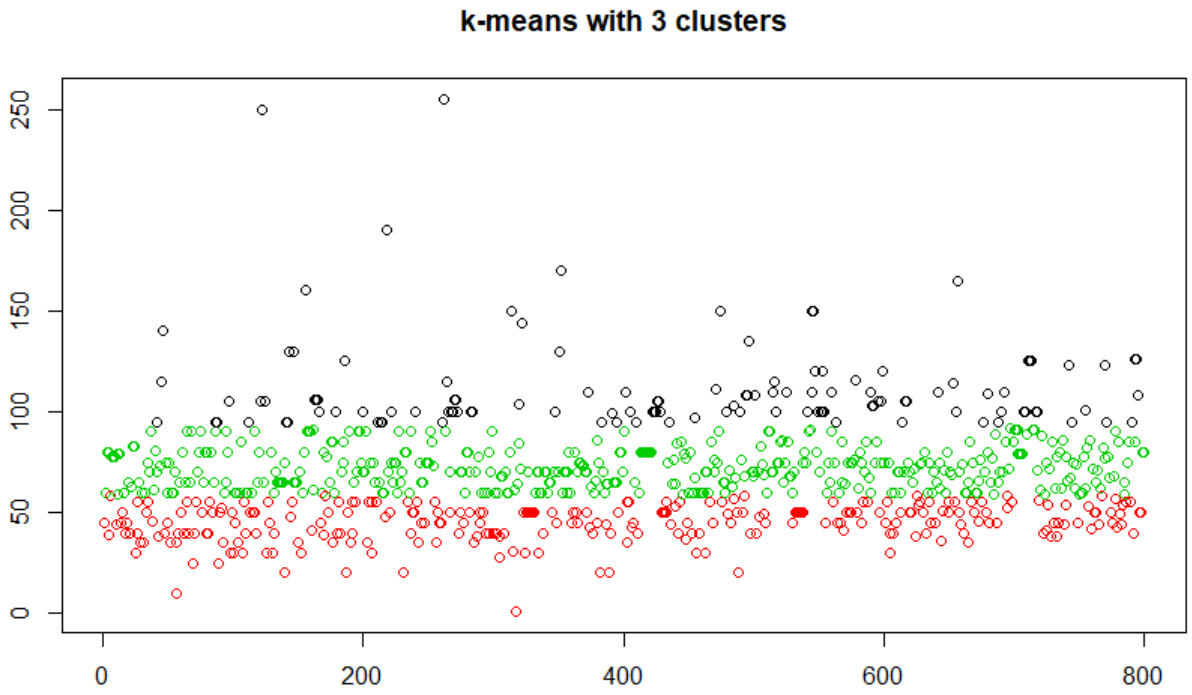
6. Print the km.out object

```
> km.out
K-means clustering with 3 clusters of sizes 123, 266, 411

Cluster means:
       [,1]
1 111.82114
2  44.84586
3  72.32117

Clustering vector:
  [1] 2 3 3 3 2 2 3 3 3 2 3 3 3 2 2 3 2 2 3 3 2 3 3 3 2 2 2 3 2 3 2 3 2 3 3 2 3 3 3 1 2 3 1 1 2 3 2 3 3 2 3 3 3 2 2 2
 [59] 3 2 3 2 3 2 3 2 3 3 2 2 2 2 3 3 3 2 3 3 2 3 2 2 3 2 3 3 1 1 2 2 2 2 3 3 3 3 1 2 2 2 2 3 3 2 3 3 2 2 2 3 3 1 2 3 2 2
[117] 3 2 3 3 1 1 3 1 1 2 2 2 3 2 3 2 3 3 3 3 3 3 3 3 2 1 1 1 2 2 1 3 3 3 2 3 2 3 3 3 1 3 3 3 2 3 3 1 1 1 1 2 3 3 2 2 3 2 3
[175] 3 2 3 3 1 2 2 2 3 3 3 1 2 2 3 2 2 2 3 2 3 3 3 3 3 1 3 3 2 2 3 2 2 3 3 2 1 3 1 3 1 3 2 1 3 2 3 1 3 3 3 3 3 3 3 3 2 3
[233] 3 2 3 3 2 2 2 1 2 2 3 2 3 3 2 3 3 3 3 3 3 3 3 2 2 2 2 2 2 1 1 3 1 1 2 3 1 1 1 1 1 2 2 3 3 2 3 3 3 2 3 1 1 2 3 2 3 2 2
[291] 3 2 3 2 3 3 2 3 3 2 3 2 3 3 2 2 3 3 2 3 3 3 3 3 3 1 2 3 2 3 3 1 3 1 2 2 2 3 2 2 2 2 2 3 3 3 2 3 3 2 3 3 3 3 3 3 2 3 1 2
[349] 3 3 1 1 3 3 3 3 3 3 3 2 2 3 2 3 2 3 3 3 3 3 3 3 2 1 2 3 2 3 3 3 2 3 2 1 3 3 2 3 3 2 2 1 3 3 3 1 2 3 3 3 3 3 1 2 2 2 1 2
[407] 2 3 1 1 2 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 2 2 2 2 2 2 3 1 2 3 3 2 3 3 2 2 3 3 3 2 3 2 3 3 2 3 3 3 1 2 3 2 3 3 3 3 2 3
[465] 3 2 3 2 3 3 1 3 3 1 2 3 3 3 1 2 3 2 3 1 2 3 2 2 1 3 2 2 3 1 1 1 2 3 3 3 1 2 3 2 3 3 2 3 2 3 3 3 3 3 3 1 1 1 3 3 3 3 3
[523] 3 3 1 3 3 3 3 3 2 3 2 2 2 2 2 2 3 3 3 1 3 3 1 1 1 1 3 1 3 1 1 1 1 2 3 3 3 3 1 2 3 1 2 3 2 3 3 3 2 3 2 3 2 3 2 3 3 1 2 3
[581] 3 2 3 2 3 3 2 3 3 1 1 1 3 3 1 2 3 1 1 3 2 2 3 2 2 3 2 3 2 3 3 2 3 1 3 1 1 3 2 3 2 3 3 2 2 2 3 2 3 2 3 2 3 2 3 2 3 3
[639] 2 3 1 3 3 3 2 2 3 3 3 2 2 3 3 1 2 1 1 2 3 2 3 2 3 3 2 3 3 2 3 2 3 3 2 3 3 3 2 1 3 2 3 1 2 3 3 3 3 2 3 1 3 1 3 1 3 2 2 3
[697] 3 2 3 3 3 3 3 3 3 3 3 1 1 1 3 3 1 1 1 3 3 1 1 3 2 3 3 2 3 3 2 2 3 2 3 2 3 3 2 2 3 3 3 2 2 3 3 1 3 1 3 3 3 3 2 3 3 3 3 1
[755] 3 3 2 3 2 3 2 3 2 3 2 3 2 3 3 1 1 3 3 2 2 3 3 2 2 3 2 2 2 3 3 2 3 3 2 1 2 3 1 1 1 2 2 3 3 3

Within cluster sum of squares by cluster:
[1] 77252.07 21512.68 36989.61
 (between_SS / total_SS =  73.9 %)

Available components:

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

7. **Visualizing and interpreting results of K-means()**

#scatter plot of x

```
plot(x,
  col = km.out$cluster,
  main = "k-means with 3 clusters",
  xlab = "",
  ylab = "")
```



D. How kmeans() works and practical matters

Process of k-means:

- randomly assign all points to a cluster
- calculate center of each cluster
- convert points to cluster of nearest center
- if no points changed, done, otherwise repeat
- calculate new center based new points
- convert points to cluster of nearest center
- and so on..

model selection:

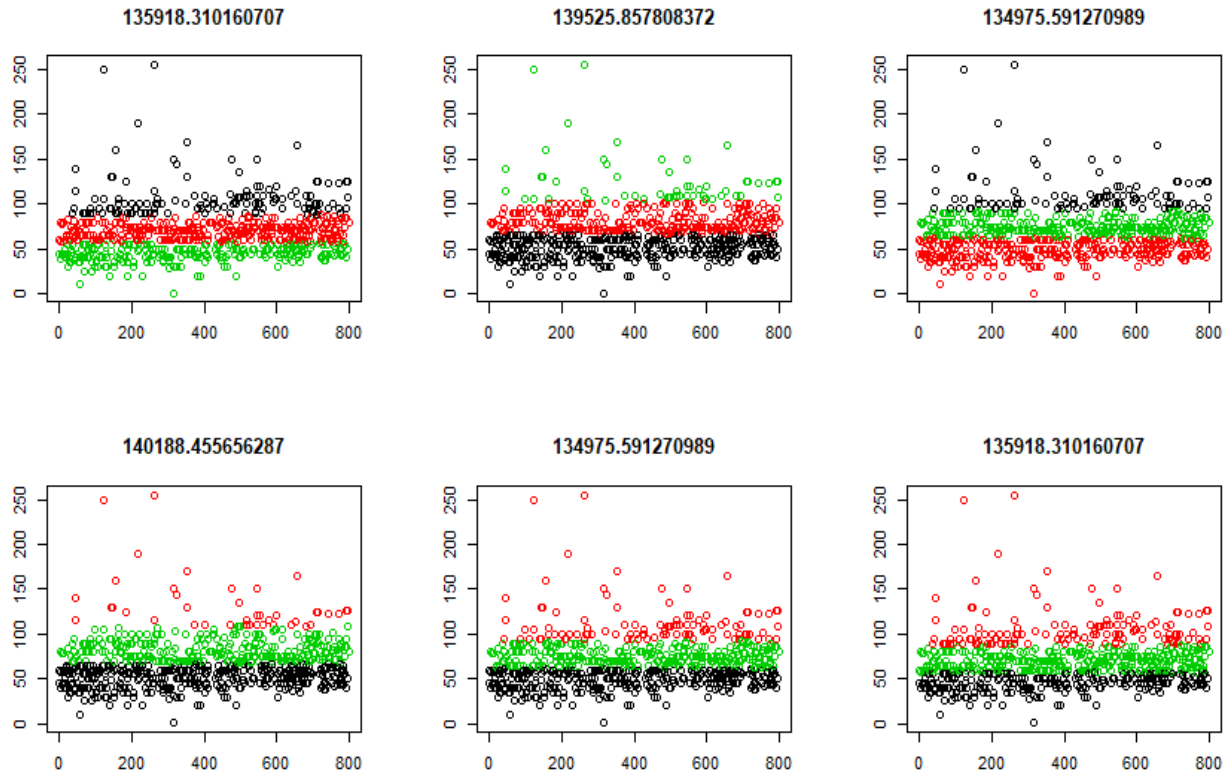
- best outcome is based on total within cluster sum of squares
- run many times to get global optimum
- R will automatically take the run with the lowest total withinss

determining number of clusters

- scree plot
- look for the elbow
- find where addition on new cluster does not change best withinss much
- there usually is no clear elbow in real world data

1. Handling random algorithms

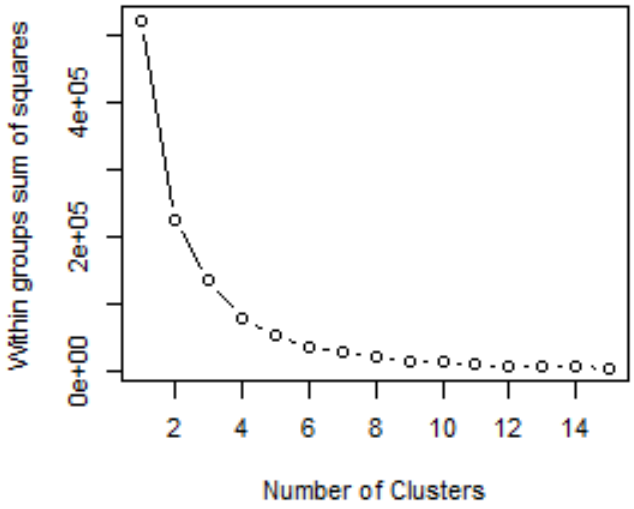
```
# Set up 2 x 3 plotting grid
par(mfrow = c(2, 3))

# Set seed
set.seed(1)

for(i in 1:6) {
  # Run kmeans() on x with three clusters and one start
  km.out <- kmeans(x, centers = 3, nstart = 1)

  # Plot clusters
  plot(x, col = km.out$cluster,
       main = km.out$tot.withinss,
       xlab = "", ylab = "")
}
```



2. Selecting number of clusters

```
# Initialize total within sum of squares error: wss
wss <- 0
```

```
# For 1 to 15 cluster centers
for (i in 1:15) {
  km.out <- kmeans(x, centers = i, nstart = 20)
  # Save total within sum of squares to wss variable
  wss[i] <- km.out$tot.withinss
}

# Plot total within sum of squares vs. number of clusters
plot(1:15, wss, type = "b",
     xlab = "Number of Clusters",
     ylab = "Within groups sum of squares")
```




```
# Set k equal to the number of clusters corresponding to the elbow location
k <- 2
```

Now, note the difference after regenerating the K-means and plotting the clusters with k = 2.