Cairo University
Faculty of Engineering
Computer Engineering Department



# Big Data and Cloud Computing
## Clustering

## Requirement (1):

| | |
|---|---|
| 1. | First of all, start by cleaning the workspace and setting the working directory. |
| 2. | Import the dataset **clustering_data.csv** into a data frame and plot the points. |
| 3. | Perform a *k-means clustering* on the data with 10 clusters and 15 iterations. |
| 4. | Print the cluster centroids. |
| 5. | Plot data such that each point is colored according to its cluster. |
| 6. | Overlay the cluster centroids on the above plot. Plot them as solid filled triangles. (***Hint***: Check the documentation of points() ) |
| 7. | What's the difference between *plot()* and *points()*? |
| 8. | Now, determine the best number of clusters by two different ways. |
| 9. | Tabulate the voting results. |
| 10. | Perform *a k-means clustering* on the data with the best number of clusters chosen in step 3. Choose an appropriate number of iterations. |
| 11. | Repeat (4, 5, 6) for the new number of clusters. |

## Requirement (2):

In this requirement, you will apply K-means to image compression. In a straightforward 24-bit color representation of an image, each pixel is represented as three 8-bit unsigned integers (ranging from 0 to 255) that specify the red, green and blue intensity values. This encoding is often referred to as the **RGB** encoding.

Our image contains thousands of colors, and in this part of the exercise, you will reduce the number of colors to 16 colors. By making this reduction, it is possible to represent (compress) the photo in an efficient way. Specifically, you only need to store the RGB values of the 16 selected colors, and for each pixel in the image you now need to only store the index of the color at that location (where only 4 bits are necessary to represent 16 possibilities).

In this exercise, you will use the K-means algorithm to select the 16 colors that will be used to represent the compressed image. Concretely, you will treat every pixel in the original image as a data example and use the K-means algorithm to find the 16 colors that best group (cluster) the pixels in the 3-dimensional RGB space. Once you have computed the cluster centroids on the image, you will then use the 16 colors to replace the pixels in the original image.

| | |
|---|---|
| 1. | First of all, start by cleaning the workspace and setting the working directory. |
| 2. | Read the image **bird_small.png.**<br>**Hint:** `image<-readPNG("bird_small.png")` |

| | |
|---|---|
| | **Note (1):** You may need to install **png** package with the command<br>`install.packages("png")`<br>**Note (2):** The resulting image is a 3-dimensional vector. |
| 3. | For the 3-dimensional image, You have one dimension for R, one dimension for G, and one dimension for B.<br>Create a data frame with three columns:<br>- The first column contains all R values for all pixels.<br>- The second column contains all G values for all pixels.<br>- The third column contains all B values for all pixels.<br>Now every row in the data frame represents a pixel. |
| 4. | Perform a k-means clustering on the data with 16 clusters and 15 iterations. |
| 5. | Show the cluster each point belongs to. |
| 6. | Display the 16 centroids of the clusters. What do these centroids represent? |
| 7. | Assign each pixel to the centroid of its cluster. |
| 8. | Reshape the data frame again to a 3-dimensional form.<br>**Hint: You can reshape any data frame with the function dim()**<br>Each dimension represents one of the RGB values. |
| 9. | Write the compressed image in a file named **compressed.png**. Now check its size. |