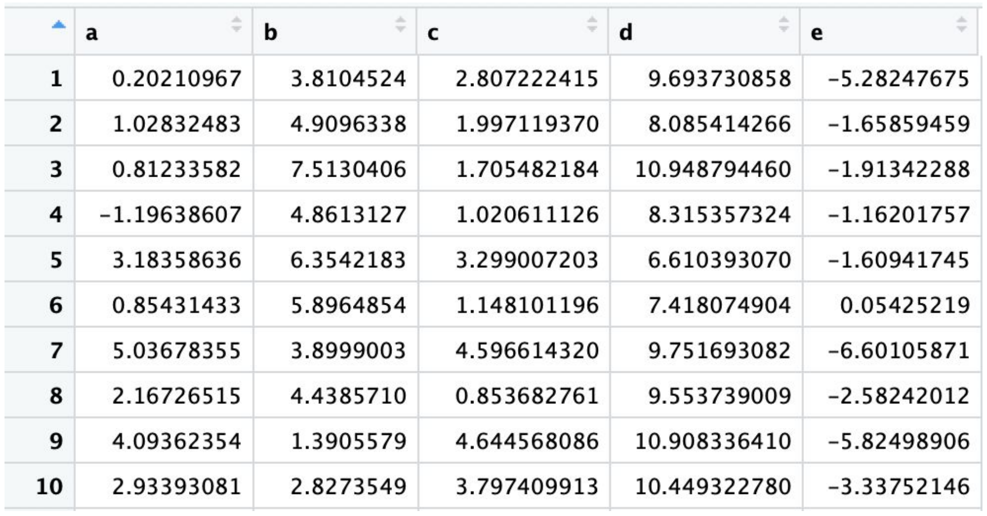
Stat-581: Probability and Statistical Inference for Data Science
Assignment - 5 Clustering

Keya Desai (kd706)
Twisha Naik (tn268)
Prakruti Joshi (phj15)

Dataset



	a	b	c	d	e
1	0.20210967	3.8104524	2.807222415	9.693730858	-5.28247675
2	1.02832483	4.9096338	1.997119370	8.085414266	-1.65859459
3	0.81233582	7.5130406	1.705482184	10.948794460	-1.91342288
4	-1.19638607	4.8613127	1.020611126	8.315357324	-1.16201757
5	3.18358636	6.3542183	3.299007203	6.610393070	-1.60941745
6	0.85431433	5.8964854	1.148101196	7.418074904	0.05425219
7	5.03678355	3.8999003	4.596614320	9.751693082	-6.60105871
8	2.16726515	4.4385710	0.853682761	9.553739009	-2.58242012
9	4.09362354	1.3905579	4.644568086	10.908336410	-5.82498906
10	2.93393081	2.8273549	3.797409913	10.449322780	-3.33752146

The dataset has 5 feature vectors (no. of columns) and 446 data points (no. of rows). The aim is to cluster these data points into groups of ‘k = 3’. We assume that the distribution of the data point is multivariate.

K-means

Pseudo Code

1. K = Number of clusters is given
2. Randomly assign each of the data points to a cluster
3. Compute K centroids by taking the average of the points assigned to that particular cluster
4. Compute this distance of each of the point from the K clusters and assign it to the closest cluster
5. Repeat step 4 and 5 until a convergence condition is satisfied

Stopping Criteria for K-Means Clustering

There can be different stopping criteria that can be adopted to stop the K-means algorithm:

1. Centroids of newly formed clusters do not change
2. Points remain in the same cluster
3. Maximum number of iterations are reached

Measure of Goodness of Clustering

Dunn index is the ratio of the minimum of inter-cluster distances and the maximum of intra-cluster distances.

- Min. Inter-cluster distance = Min (Distance between each pair of cluster centroids)
- Max. Intra-cluster distance = Max (Distance of each point from respective cluster centroid)

$$\boxed{Dunn\ Index\ =\ \frac{min(Inter\ cluster\ distance)}{max(Intra\ cluster\ distance)}}$$

The values of the Dunn index have to be maximized because we want to maximize minimum the distance between two clusters and minimize the maximum distance (or spread) within the cluster. Once the algorithm converges, the cluster assignment remains constant and hence the Dunn value gets stable.

With each iteration, the value of the Dunn index should increase as we are making our clusters better and better. This can be observed from the Dunn index calculated at each iteration for the k-means algorithm.

```
"-- 1 ---"
0.01946066
"-- 2 ---"
0.2814332
"-- 3 ---"
0.418629
"-- 4 ---"
0.5137268
"-- 5 ---"
0.5160091
"-- 6 ---"
0.5165099
"-- 7 ---"
0.5165099
```

<u>Results</u>

1. **Centroid values**

```
           a         b         c         d         e
1 -9.4204480  1.672011 0.5089758  1.919512 -1.232519
2  0.3144581 -7.201651 2.5441233 -6.065219  2.163828
3  2.1741308  4.229824 3.4080843  8.514269 -2.839595
```

2. **The final covariance matrix values for each cluster**

Cluster - 1

```
           a         b          c          d          e
a  6.0258522 -2.103989 -1.5924905 -0.6766731 -1.2684408
b -2.1039894  7.354028 -1.4223185 -1.9793705 -1.6959598
c -1.5924905 -1.422319  5.8006265 -1.5018690 -0.9890357
d -0.6766731 -1.979370 -1.5018690  5.1324572 -0.8965703
e -1.2684408 -1.695960 -0.9890357 -0.8965703  5.0772760
```

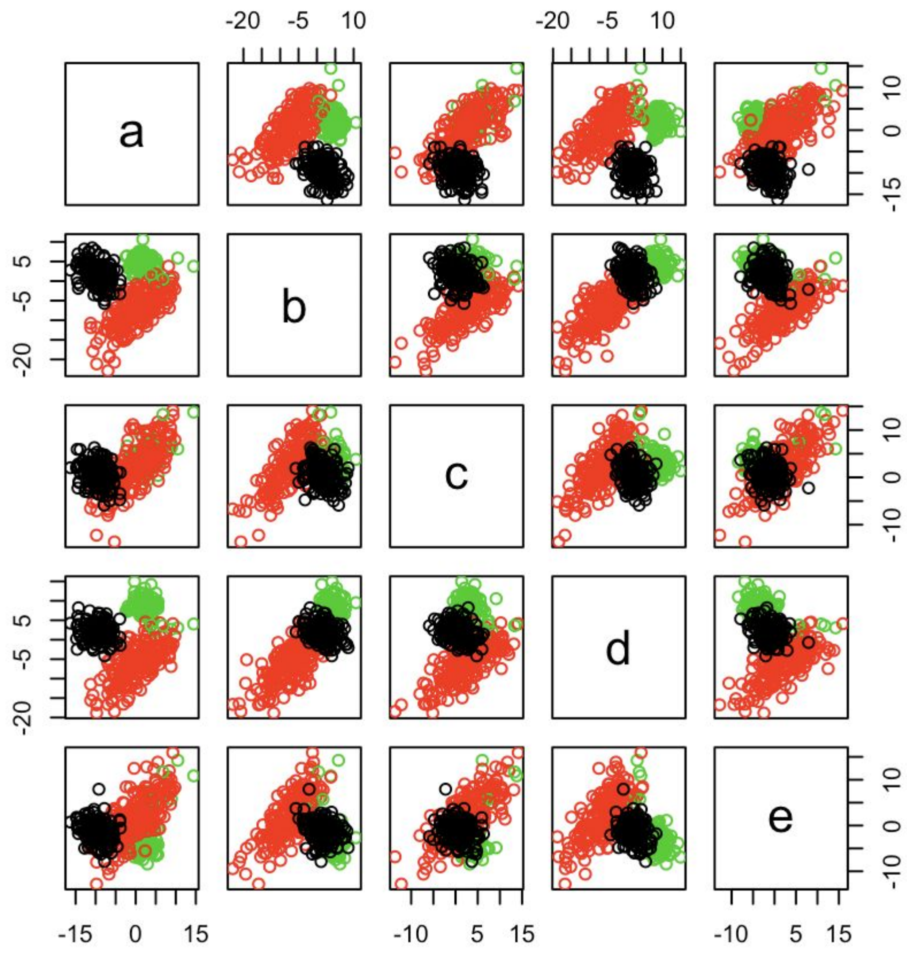
Cluster - 2

```
         a        b        c        d        e
a 22.23066 15.78934 17.03207 14.71740 16.34372
b 15.78934 22.68029 17.36112 15.27510 17.02358
c 17.03207 17.36112 23.71020 17.10150 19.16054
d 14.71740 15.27510 17.10150 22.88242 16.59680
e 16.34372 17.02358 19.16054 16.59680 24.47796
```

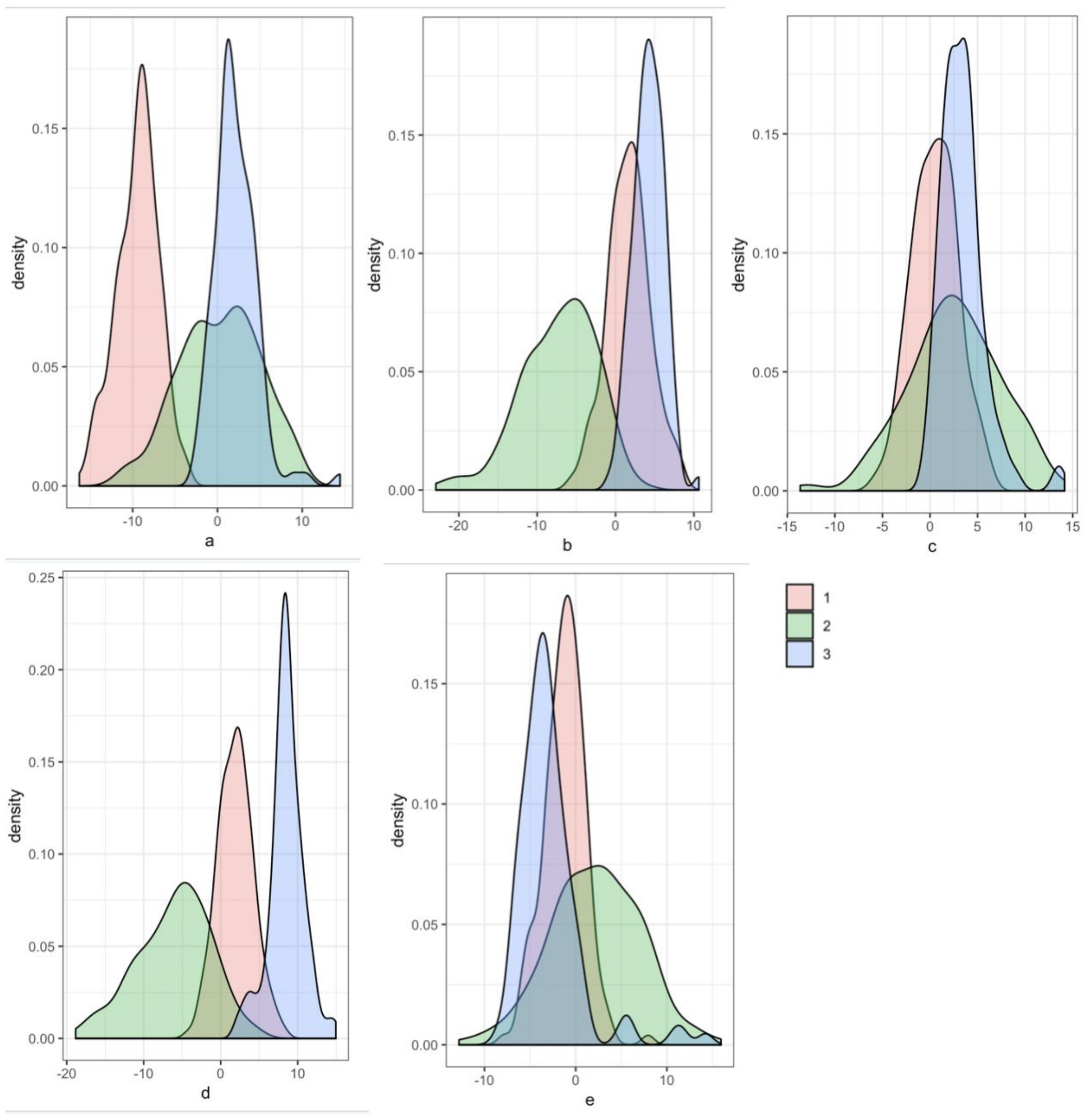
Cluster - 3

```
           a          b         c          d         e
a  6.4036049 -0.9770626  2.328004 -1.9622532  4.608219
b -0.9770626  3.7213929 -1.109333  0.3019885 -1.535218
c  2.3280044 -1.1093327  5.541228 -2.2649431  3.555287
d -1.9622532  0.3019885 -2.264943  4.9181470 -4.158362
e  4.6082188 -1.5352180  3.555287 -4.1583615 13.003833
```

3. **Pair-Pair Plots**



Below are the density plots for each of the clusters after the convergence of a clustering algorithm for each of the features - a, b, c, d, e - in the data.

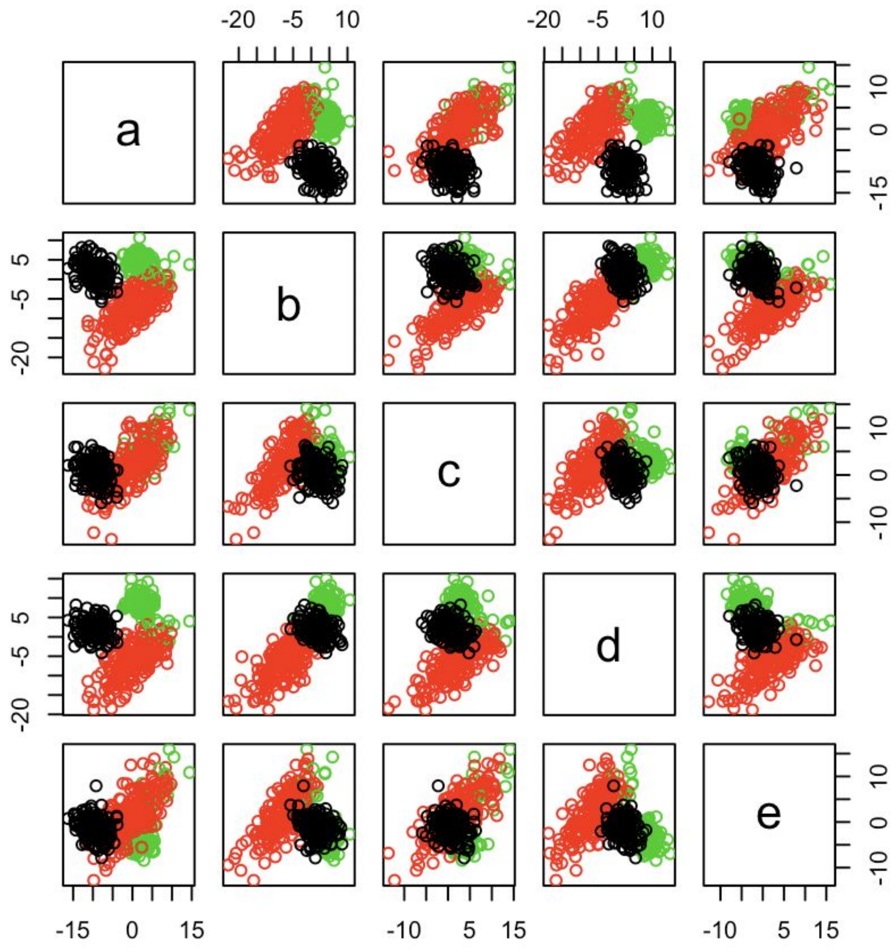


To verify the results, we used the built-in k-means function to obtain the clusters for the data and compare the results.

1. Centroid values

```
           a         b         c         d         e
1 -9.4204480  1.672011 0.5089758  1.919512 -1.232519
2  0.0870932 -7.460431 2.2863226 -6.361974  1.918583
3  2.4229265  3.941031 3.7666107  8.092804 -2.140835
```

2. Pair-Pair Plots



Observation:
The values obtained from this function are almost the same as the one calculated by the self-implemented function.

Gaussian Mixture Models -- (Additional Approach)

Assuming that the data points come from a Gaussian mixture distribution, its probability distribution will simply be a linear superposition of Gaussians:

$$\boxed{p(x) = \sum_{k=1}^{K} \pi_k N(x \mid \mu_k, \Sigma_k)}$$

Here, we will assume that our data is coming from a mixture of k different Gaussians. The value of each Gaussian at a certain point will describe the confidence that the point is coming from that Gaussian.

Algorithm:

- Firstly, we will randomly initialize K centers (mean of Gaussians) and a random covariance matrix.
- For each data point, we will calculate the confidence of it coming from one of the K Gaussians.
- We will assign each point to the centroid having maximum confidence.
- Using the new cluster assignment, we will recompute the covariance matrix for each of the clusters and repeat the process.

The block below explains the formal algorithm with the necessary equations.

Expectation-Maximization (EM) for Gaussian Mixtures

Given a Gaussian mixture model, the goal is to maximize the likelihood function with respect to the parameters - mean and covariances of the components and the mixing coefficients.

1. Initialize the means μ_k, covariances Σ_k and the mixing coefficients π_k, and evaluate the initial value of the log likelihood.
2. E Step - Evaluate the responsibilities using the parameter values

$$\gamma(z_{nk}) = \frac{\pi_k N(x \mid \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j N(x_n \mid \mu_j, \Sigma_j)}$$

3. M step - Re-estimate the parameters using the current responsibilities

$$\mu_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) x_n$$

$$\Sigma_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(x_n - \mu_k^{new})(x_n - \mu_k^{new})^T$$

$$\pi_k^{new} = \frac{N_k}{N}, \text{ where } N_k = \sum_{n=1}^{N} \gamma(z_{nk})$$

4. Evaluate the log likelihood

$$lnp(X \mid \mu, \Sigma, \pi) = \sum_{n=1}^{N} ln \{ \sum_{k=1}^{K} \pi_k N(x \mid \mu_k, \Sigma_k) \}$$

5. Check for convergence of either the parameters or the log likelihood. If the convergence criterion is not satisfied return to step 2.

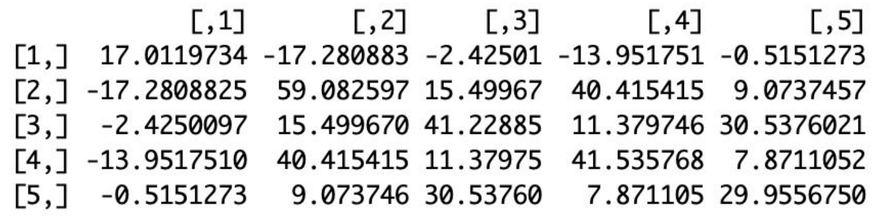
Results

1. **Centroid values**

Cluster	**a**	**b**	**c**	**d**	**e**
1	-6.432179	-5.282288	-0.5185152	-5.3914112	-1.474691
2	2.155437	3.971810	3.3992332	7.9862942	-2.527122
3	-2.579879	-2.029299	2.9644535	-0.6776968	1.895210

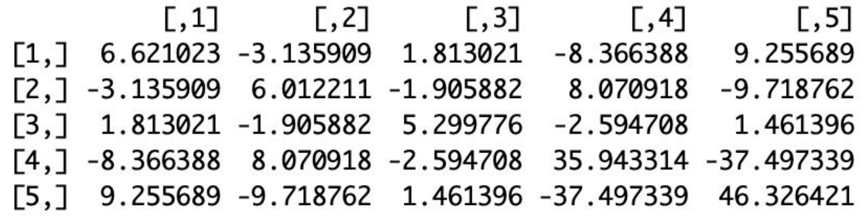
2. **The final covariance matrix values for each cluster**

Cluster - 1



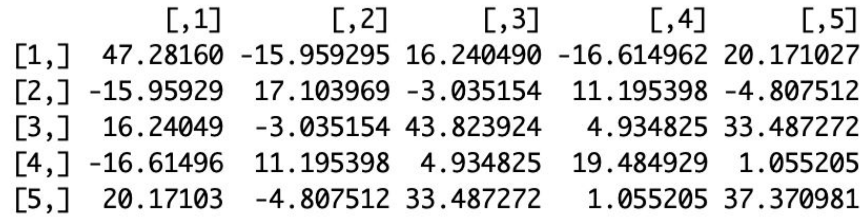
```
            [,1]       [,2]     [,3]       [,4]       [,5]
[1,]  17.0119734 -17.280883 -2.42501 -13.951751 -0.5151273
[2,] -17.2808825  59.082597 15.49967  40.415415  9.0737457
[3,]  -2.4250097  15.499670 41.22885  11.379746 30.5376021
[4,] -13.9517510  40.415415 11.37975  41.535768  7.8711052
[5,]  -0.5151273   9.073746 30.53760   7.871105 29.9556750
```

Cluster - 2



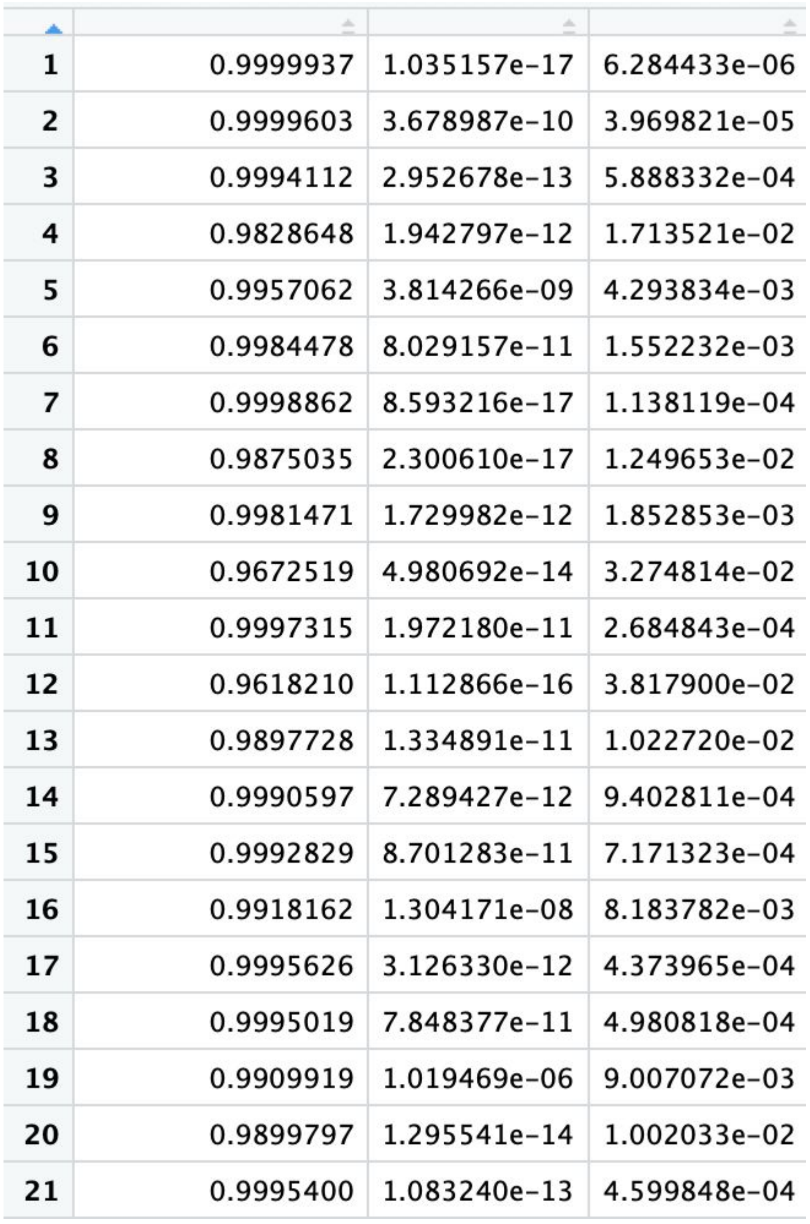
```
          [,1]      [,2]      [,3]       [,4]       [,5]
[1,]  6.621023 -3.135909  1.813021  -8.366388   9.255689
[2,] -3.135909  6.012211 -1.905882   8.070918  -9.718762
[3,]  1.813021 -1.905882  5.299776  -2.594708   1.461396
[4,] -8.366388  8.070918 -2.594708  35.943314 -37.497339
[5,]  9.255689 -9.718762  1.461396 -37.497339  46.326421
```

Cluster - 3



```
          [,1]       [,2]      [,3]       [,4]      [,5]
[1,]  47.28160 -15.959295 16.240490 -16.614962 20.171027
[2,] -15.95929  17.103969 -3.035154  11.195398 -4.807512
[3,]  16.24049  -3.035154 43.823924   4.934825 33.487272
[4,] -16.61496  11.195398  4.934825  19.484929  1.055205
[5,]  20.17103  -4.807512 33.487272   1.055205 37.370981
```

3. Responsibility Matrix

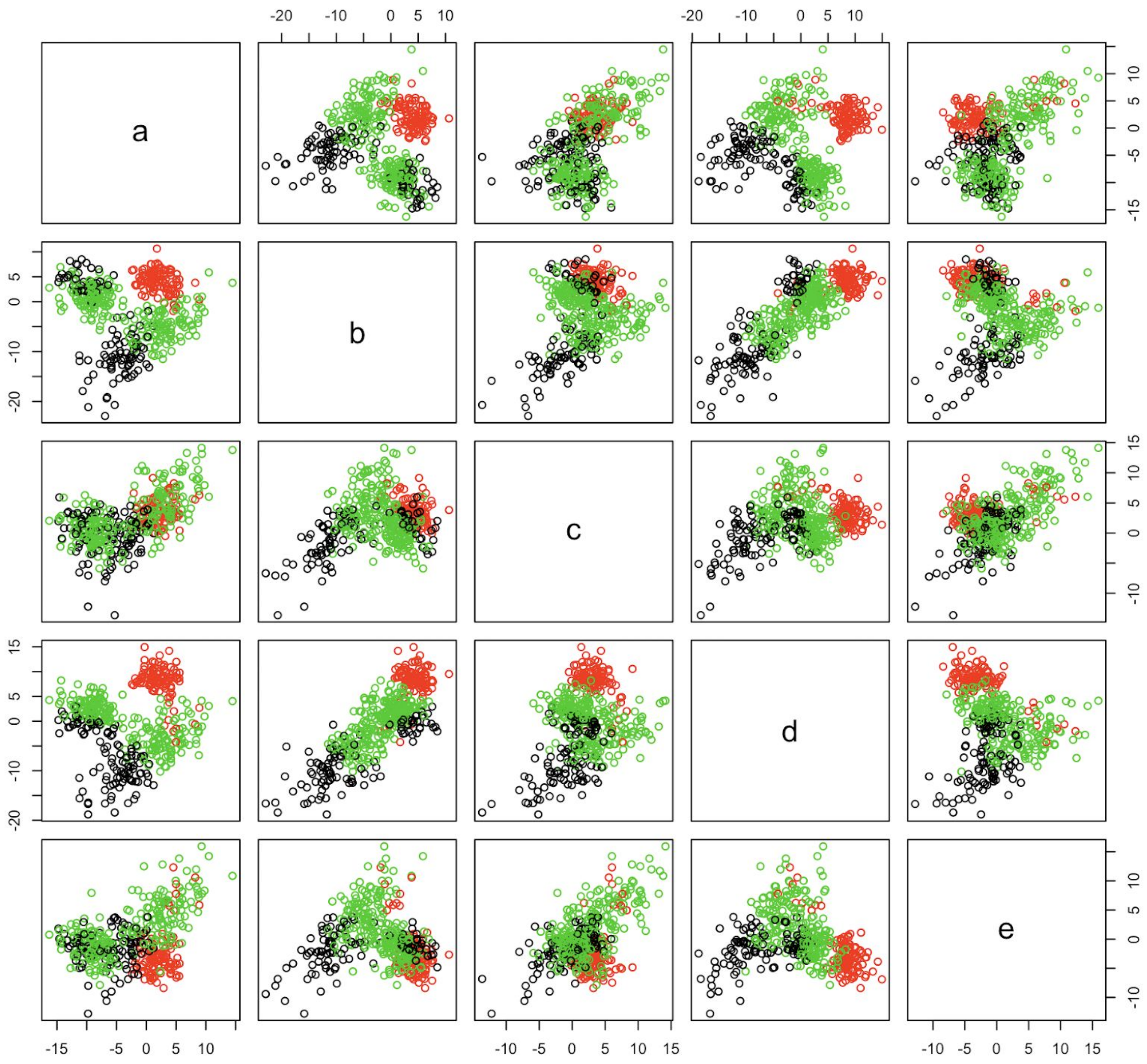


1	0.9999937	1.035157e-17	6.284433e-06
2	0.9999603	3.678987e-10	3.969821e-05
3	0.9994112	2.952678e-13	5.888332e-04
4	0.9828648	1.942797e-12	1.713521e-02
5	0.9957062	3.814266e-09	4.293834e-03
6	0.9984478	8.029157e-11	1.552232e-03
7	0.9998862	8.593216e-17	1.138119e-04
8	0.9875035	2.300610e-17	1.249653e-02
9	0.9981471	1.729982e-12	1.852853e-03
10	0.9672519	4.980692e-14	3.274814e-02
11	0.9997315	1.972180e-11	2.684843e-04
12	0.9618210	1.112866e-16	3.817900e-02
13	0.9897728	1.334891e-11	1.022720e-02
14	0.9990597	7.289427e-12	9.402811e-04
15	0.9992829	8.701283e-11	7.171323e-04
16	0.9918162	1.304171e-08	8.183782e-03
17	0.9995626	3.126330e-12	4.373965e-04
18	0.9995019	7.848377e-11	4.980818e-04
19	0.9909919	1.019469e-06	9.007072e-03
20	0.9899797	1.295541e-14	1.002033e-02
21	0.9995400	1.083240e-13	4.599848e-04

Each of the row in the matrix corresponds to a data point. Each of the column is the responsibility that the corresponding component k takes to explain the particular observation.

The observation is assigned to the cluster that has the highest responsibility for that particular observation.

4. Pair-Pair Plots



5. Convergence in delta (the difference between log-likelihood values)

We have used the difference in the log-likelihood function in consecutive iterations for convergence. We stop when the improvement in log-likelihood is less than some threshold value.

The EM for algorithm converges in 25 steps with the stopping condition being Delta < 0.1.

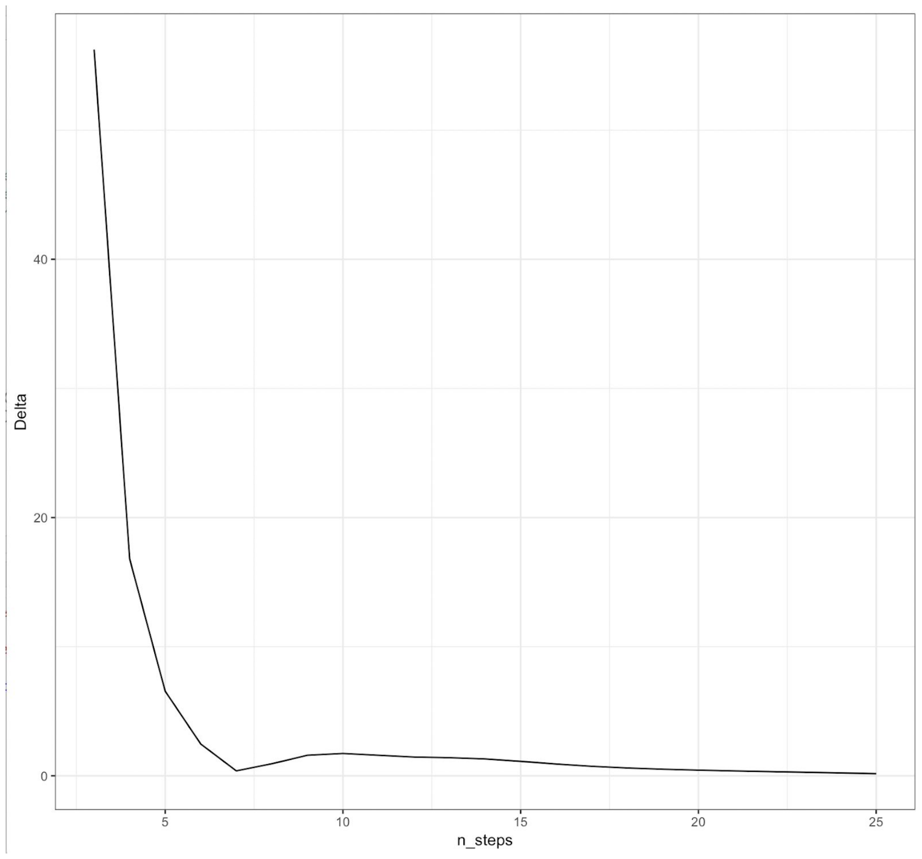


Fig. Convergence of Delta

References

1. https://stephens999.github.io/fiveMinuteStats/intro_to_mixture_models.html
2. https://stephens999.github.io/fiveMinuteStats/intro_to_em.html
3. http://www.cs.cmu.edu/~guestrin/Class/10701-S07/Slides/clustering.pdf
4. https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/

Appendix - Code Snippets

1. **Cluster initialization**

```
# assigning each data point to a cluster randomly at first
clusters.init <- function(data, k){

  n = nrow(data)
  ## assign each point to random clusters
  clusters = rep(1:k, length.out = n, replace = TRUE)
  data = cbind(data, clusters)

  return(data)
}
```

2. **Compute centroid means and variance based on cluster assignment**

```
# recalculating the centroid/means and the covariance of all the
clusters, based on the points assigned to them
clusters.parameters <- function(data, k){

  num.features = ncol(data)-1
  #taking mean feature wise
  mean = aggregate(data, by = list(data$clusters), mean, na.rm =
TRUE)
  cov.list <- lapply( sort(unique(data$clusters)), function(x)
cov(data[data$clusters==x,-(num.features+1)],use="na.or.complete"))

  return(list(mean = mean[-1], cov.list = cov.list, cluster_number =
sort(unique(data$clusters)) ))
}
```

3. **Compute Euclidean distance and re-assign clusters**

```
euc.dist <- function(x1, x2){
  return(sqrt(sum((x1 - x2) ^ 2)))
}
```

```
#re-assign cluster to each data point based on the points euclidean
distance from the cluster centroids
assign.cluster.euclean <- function(data, clusters.list){

  n = nrow(data)
  num.features = ncol(data)-1

  for(i in c(1:n)){
    min_distance = +Inf
    for( j in  unique(data$clusters) ){
        mean = clusters.list$mean[j,-(num.features+1)]
        distance = euc.dist(data[i,-ncol(data)], mean )
        if(distance < min_distance){
          min_distance = distance
          #assign the cluster with min distance from the data point
          data[i,]$clusters = j
         }
      }
    }
  return(data)
}
```

4. **Main K-means function**

```
k.means <- function(data, k){
  data = clusters.init(data, k)
  data.init.clusters = data
  plot.clusters(data)

  ##convergence of dunn index
  prev_dunn_index = 0
  delta = 1

  # stop when dunn index becomes constant
  while(delta!=0){
    print("-------------")
    #clusters.list contains the cluster parameters of mean and
covariance for each cluster
```

```
    clusters.list = clusters.parameters(data, k)
    # new column of 'clusters' gets appended
    data = assign.cluster.eucliean(data,clusters.list )
    plot.clusters(data)

    new_dunn_index = dunn_index(data, clusters.list$mean[-6])
    print(new_dunn_index)
    delta = abs(new_dunn_index - prev_dunn_index)
    prev_dunn_index = new_dunn_index
  }

  return(data)
}
```

5. **Dunn Index**

A. **Intra-cluster Distance for a given cluster**

```
intracluster_dist <- function(cluster_mean, cluster_data){
  n = nrow(cluster_data)
  max = 0
  for (i in c(1:n)){
      distance = euc.dist(cluster_mean, cluster_data[i,])
      if(distance>max){
        max = distance
      }
  }
  return(max)
}
```

B. **Maximum Intra-cluster Distance for a given cluster**

```
get_max_intracluster_distance <- function(data, mean){
  num.features = ncol(data)-1
  for (i in c(1:k)){
    t = lapply(sort(unique(data$clusters)), function(x)
intracluster_dist(mean[x,], data[data$clusters==x,
-(num.features+1)]))
  }
  return(max(unlist(t)))
}
```

C. Minimum Intra-cluster Distance between each pair of clusters

```
get_min_intercluster_distance <- function(mean){

  k=3
  min = +Inf
  for (i in c(1:(k-1))){
    for (j in c((i+1):k)){
      distance = euc.dist(mean[i,], mean[j,])
      #print(distance)
      if(distance<min){
        min = distance
      }
    }
  }
  return(min)
}
```

D. Main function for computing Dunn index

```
dunn_index <- function(data, mean){
  min_intercluster_distance = get_min_intercluster_distance(mean[-6])
  #print(min_intercluster_distance)
  max_intracluster_distance = get_max_intracluster_distance(data,
mean[-6])
  #print(max_intracluster_distance)
  return(min_intercluster_distance/ max_intracluster_distance)
}
```

NOTE - Codes for GMM in the submission zip file.