Statistical Computing HW3

周聖諺

4/23/2021

Problem 1:

Two random variables are defined X_1, X_2 and combined as a set $X = \{X_1, X_2\}$

$$X_1 = \sigma_{X_1} Z_1 + \mu_{X_1}$$

$$X_2 = \sigma_{X_2}\left(\rho Z_1 + \sqrt{1-\rho^2} Z_2\right) + \mu_{X_2}$$

where $Z_1, Z_2 \sim \mathcal{N}(0,1)$ and Z_1, Z_2 are independent

The Expectation

$$\mathbb{E}[X_1] = \mathbb{E}[\sigma_{X_1} Z_1 + \mu_{X_1}] = \sigma_{X_1}\mathbb{E}[Z_1] + \mu_{X_1} = \mu_{X_1}$$

$$\mathbb{E}[X_2] = \mathbb{E}[\sigma_{X_2}(\rho Z_1 + \sqrt{1-\rho^2} Z_2 + \mu_{X_2}]$$

$$= \sigma_{X_2}\mathbb{E}[\rho Z_1 + \sqrt{1-\rho^2} Z_2] + \mu_{X_2}$$

$$= \sigma_{X_2}(\rho\mathbb{E}[Z_1] + \sqrt{1-\rho^2}\mathbb{E}[Z_2]) + \mu_{X_2} = \mu_{X_2}$$

$$\mathbb{E}[X] = \{\mu_{X_1}, \mu_{X_2}\}$$

The Covariance

$$\sigma_{X_1,X_2} = \sigma_{X_2,X_1} = \mathbb{E}[(X_1 - \mu_{X_1})(X_2 - \mu_{X_2})]$$

$$= \mathbb{E}[(\sigma_{X_1} Z_1 + \mu_{X_1} - \mu_{X_1})(\sigma_{X_2}(\rho Z_1 + \sqrt{1-\rho^2} Z_2 + \mu_{X_2} - \mu_{X_2})]$$

$$= \mathbb{E}[(\sigma_{X_1} Z_1)(\sigma_{X_2}(\rho Z_1 + \sqrt{1-\rho^2} Z_2)]$$

$$= \mathbb{E}[\sigma_{X_1}\sigma_{X_2}(\rho Z_1^2 + \sqrt{1-\rho^2} Z_1 Z_2)]$$

$$= \sigma_{X_1}\sigma_{X_2}\left(\rho\mathbb{E}[Z_1^2] + \sqrt{1-\rho^2}\mathbb{E}[Z_1 Z_2]\right)$$

With the definition of variance, we can derive $\mathbb{E}[Z_1^2] = Var[Z_1] + E[Z_1]^2 = 1$. Since Z_1, Z_2 are independent, $\mathbb{E}[Z_1 Z_2] = \mathbb{E}[Z_1]\mathbb{E}[Z_2] = 0$

$$= \sigma_{X_1}\sigma_{X_2}\rho$$

Then, conduct a simulation as pseudo code

Pseudo Code

Z_1 = (0, 1)

Z_2 = (0, 1)

$X_1 = \sigma_{X_1} Z_1 + \mu_{X_1}$

$X_2 = \sigma_{X_2}(\rho Z_1 + \sqrt{1-\rho^2} Z_2 + \mu_{X_2}$

return X_1, X_2

```
## [1] "Parameters"
```

```
## [1] "mu_X_1:  1"
```

```
## [1] "mu_X_2:  2"
```

```
## [1] "sigma_X_1:  1"
```

```
## [1] "sigma_X_2:  2"
```

```
## [1] "rho:  0.4"
```

```
## [1] "------------------"
```

```
## Mean of X_1:  0.9856213
## NULL
```

```
## Mean of X_2:  1.998983
## NULL
```

```
## Covariance of (X_1, X_2):  0.8037399
## NULL
```

Problem 2:

The conditional probability of bivariate normal distribution

$$\begin{pmatrix} X \\ Y \end{pmatrix} \sim N\left[\begin{pmatrix} \mu_X \\ \mu_Y \end{pmatrix}, \begin{pmatrix} \sigma_X^2 & \rho\sigma_X\sigma_Y \\ \rho\sigma_X\sigma_Y & \sigma_Y^2 \end{pmatrix}\right]$$

$$Y|X = N\left(\mu_Y + \rho\frac{\sigma_Y}{\sigma_X}(X-\mu_X),\ \sigma_Y^2(1-\rho^2)\right)$$

$$= \mu_Y + \rho\frac{\sigma_Y}{\sigma_X}(X-\mu_X) + \sigma_Y\sqrt{(1-\rho^2)}\mathcal{N}(0,1)$$

Gibbs Sampling Pseudo Code

For each k-th smapling

- $X_1^k = \mu_{X_2} + \rho\frac{\sigma_{X_2}}{\sigma_{X_1}}(X_2^{k-1} - \mu_{X_1}) + \sigma_{X_2}\sqrt{(1-\rho^2)}\mathcal{N}(0,1)$
- $X_2^k = \mu_{X_1} + \rho\frac{\sigma_{X_1}}{\sigma_{X_2}}(X_1^k - \mu_{X_2}) + \sigma_{X_1}\sqrt{(1-\rho^2)}\mathcal{N}(0,1)$

Return $\{\{X_1^1, \ldots, X_1^N\}, \{X_2^1, \ldots, X_2^N\}\}$

```
## [1] "The Prameters of The GMM"
```

```
##             mu_X_1 mu_X_2
## component_1      1      2
## component_2      2      1
```

```
## [1] "---"
```

```
##             sigma_X_1 sigma_X_2
## component_1         1         2
## component_2         2         1
```

```
## [1] "---"
```

```
##             rho
## component_1 0.4
## component_2 0.6
```

```
## [1] "---"
```

```
##             coefficient
## component_1         0.4
## component_2         0.6
```

```
## [1] "--------------------------------"
```

```
## [1] "Population Mean & Covariance"
```

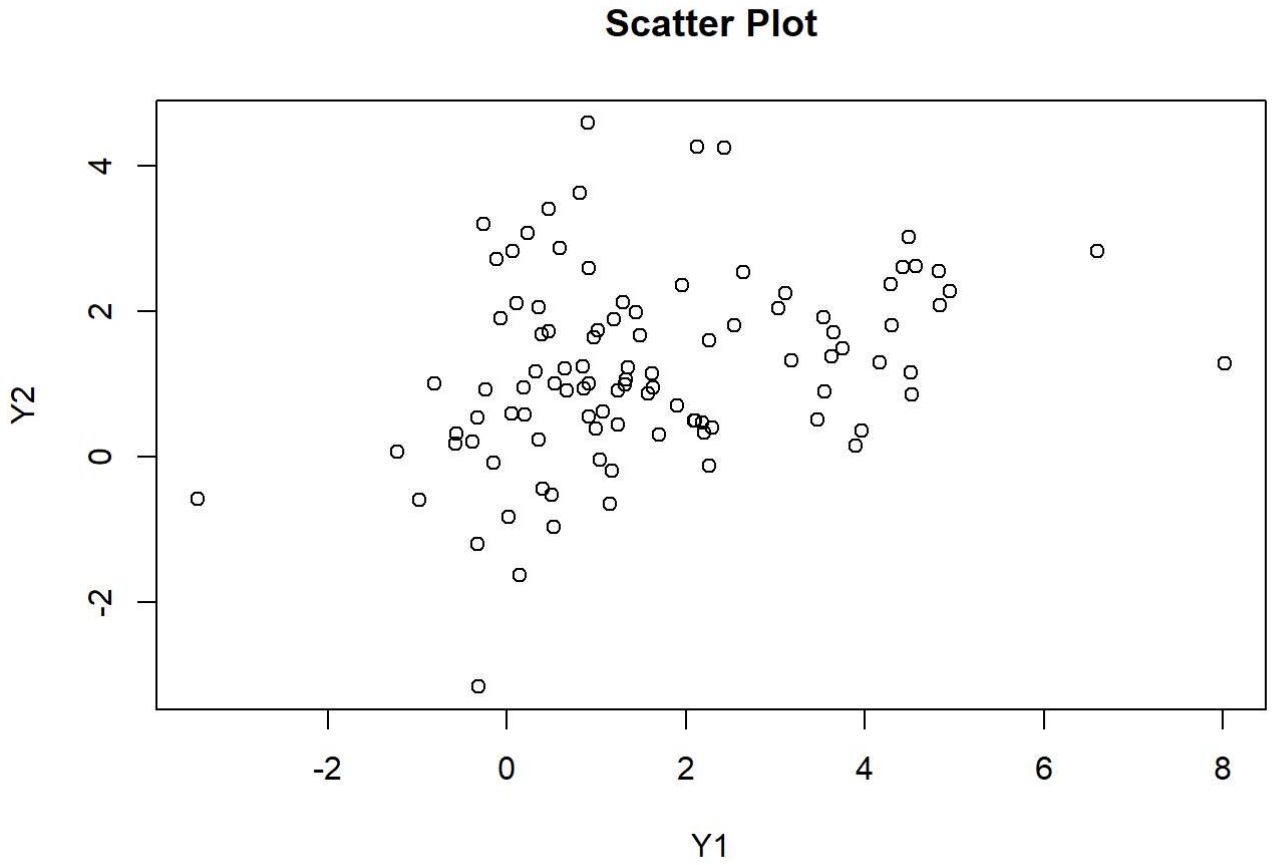
```
## Mean of Y1:  1.618999NULL
```

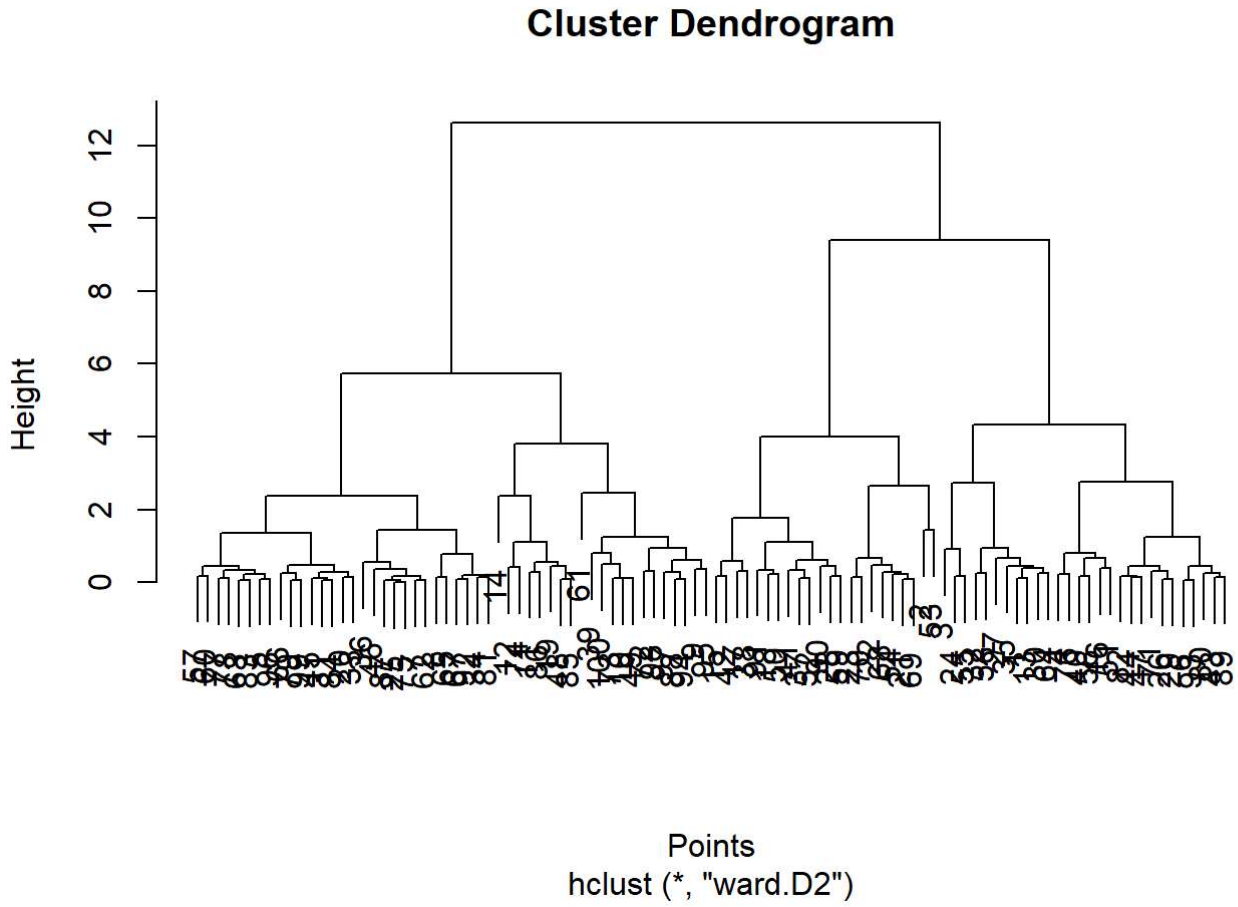
```
## Mean of Y2:  1.211232NULL
```

```
## [1] "Covariance Matrix"
```

```
##           [,1]      [,2]
## [1,] 3.3886618 0.7628359
## [2,] 0.7628359 1.6634070
```

```
## Number of data point:  100NULL
```


Scatter Plot
Y2
Y1
-2
0
2
4
6
8


Cluster Dendrogram
Height
0
2
4
6
8
10
12
Points
hclust (*, "ward.D2")

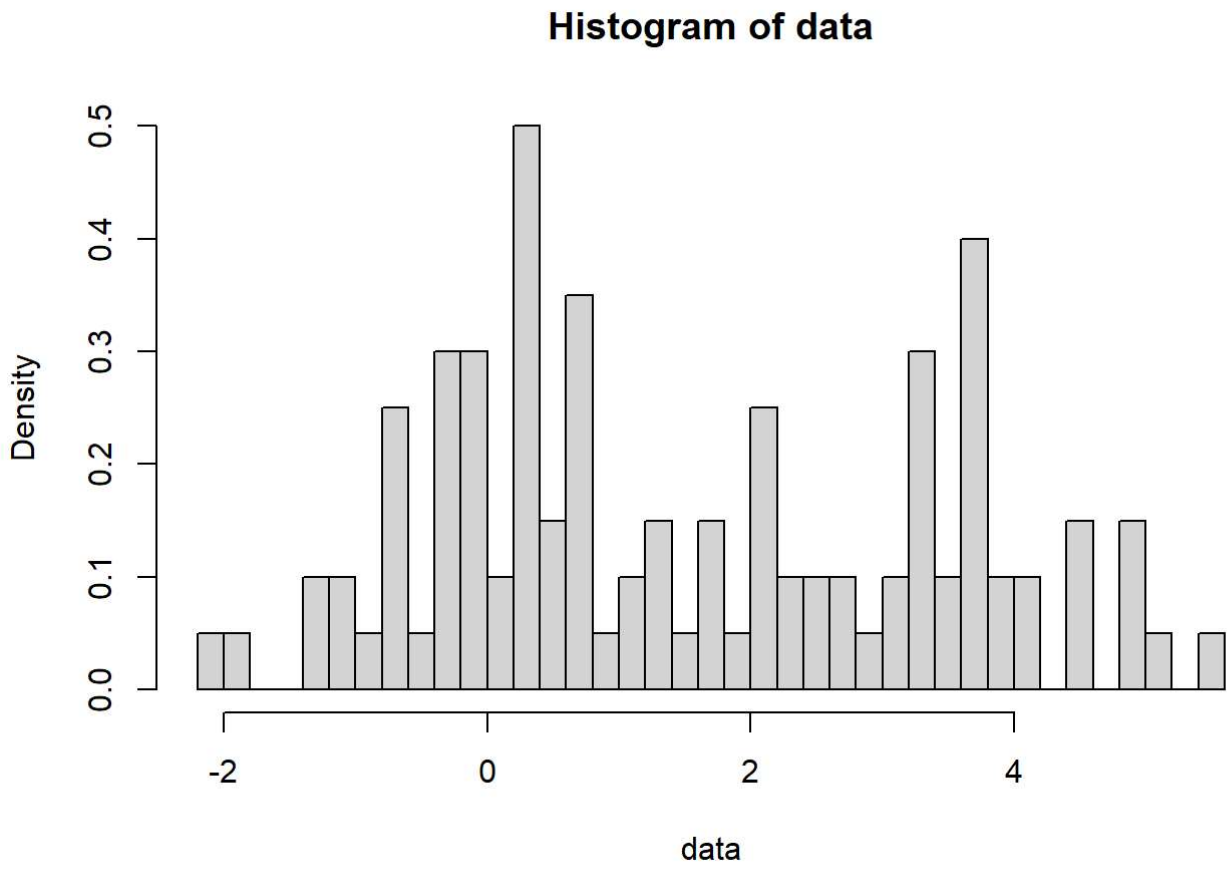
Problem 3:

Generate Data

```
## [1] 200
```

```
## [1] 100
```

```
## [1] 1.235513
```



(1) K-Means

Pseudo Code

Repeat until converge

- For each data point, Compute the distance between the data point and the nearest cluster center
- Assign the data point to the nearest cluster
- Compute the distance between each data point and each cluster center and check whether it converge or not

---

```
## Warning in cbind(dt_1, dt_2): number of rows of result is not a multiple of
## vector length (arg 1)
```

K-Means seems work well. Two clusters are divided clearly.

(2) EM-GMM

Pseudo Code

Support the GMM has K components, the j-th component follows the normal distribution $\mathcal{N}(\mu_j, \sigma_j), j \leq K$ and weighted by w_j. We also denote the i-th data point as x_i

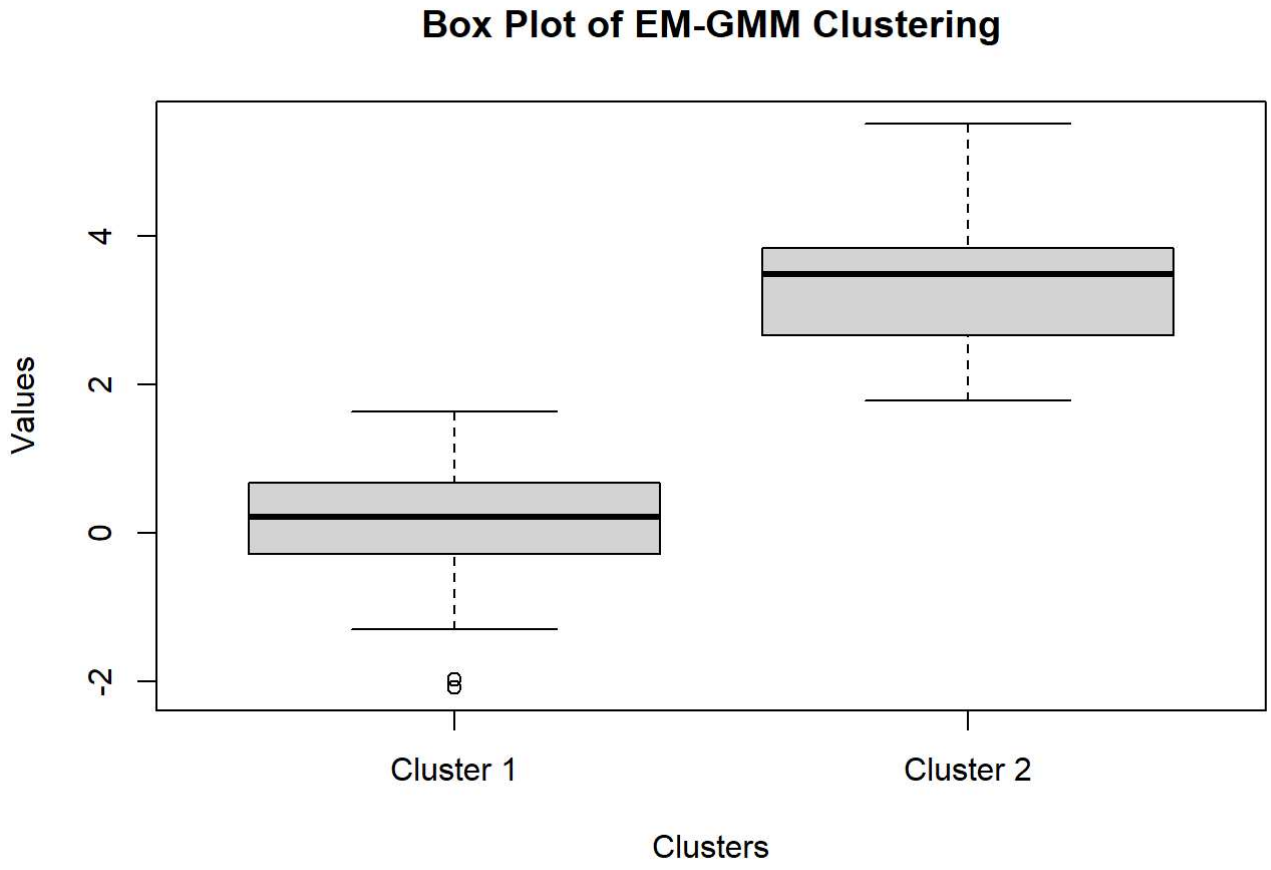
Repeat until converge

- E Step
  - Compute the likelihood $\mathcal{L}_{ij}$ of i-th data point and j-th component
  - Compute $\gamma_{ij} = \frac{w_j \mathcal{L}_{ij}}{\sum_{j=1}^{K} w_j \mathcal{L}_{ij}}$
- M Step
  - Estimate $\mu_j = \frac{\sum_{i=1}^{K} \gamma_{ij} x_i}{\sum_{i=1}^{K} \gamma_{ij}}$
  - Estimate $\sigma_j = \frac{\sqrt{\sum_{i=1}^{K} \gamma_{ij} (x_i - \mu_j)^2}}{\sum_{i=1}^{K} \gamma_{ij}}$
  - Estimate $w_j = \frac{1}{K} \sum_{i=1}^{K} \gamma_{ij}$

```
## [1] "------------------"
```

```
##        Component  1 Component  2
## ws        0.5545328    0.4454672
## mus       0.1240485    3.3407989
## sigmas    0.8716445    1.0158904
```

```
## Warning in cbind(dt_1, dt_2): number of rows of result is not a multiple of
## vector length (arg 2)
```



EM-GMM seems work well. The estimated values are close to the truth values.

Code

Problem 1

```
gen_binorm <- function(n, mu_1, mu_2, sigma_1, sigma_2, rho){
  z_1 <- rnorm(n, 0, 1)
  z_2 <- rnorm(n, 0, 1)
  z_bind <- rbind(z_1, z_2)

  x_1 <- sapply(z_1, function(z){return(sigma_1 * z + mu_1)})
  x_2 <- apply(z_bind, 2, function(z){return(sigma_2 * (rho * z[1] + sqrt(1 - rho * rho) * z[
2]) + mu_2)})

  x_bind <- rbind(x_1, x_2)
  return(x_bind)
}

mu_X_1 <- 1
mu_X_2 <- 2
sigma_X_1 <- 1
sigma_X_2 <- 2
rho <- 0.4

x <- gen_binorm(5000, mu_X_1, mu_X_2, sigma_X_1, sigma_X_2, rho)

print("Parameters")
print(paste("mu_X_1: ", toString(mu_X_1)))
print(paste("mu_X_2: ", toString(mu_X_2)))
print(paste("sigma_X_1: ", toString(sigma_X_1)))
print(paste("sigma_X_2: ", toString(sigma_X_2)))
print(paste("rho: ", toString(rho)))
print("------------------")

print(cat("Mean of X_1: ", format(mean(x[1, ])), " \n"))
print(cat("Mean of X_2: ", format(mean(x[2, ])), " \n"))
print(cat("Covariance of (X_1, X_2): ", format(cov(x[1, ], x[2, ])), " \n"))
```

Problem 2

```
library(Rlab)
```

```
binorm <- function(n, mus, sigmas, rho, warmup=10000){
  # Dimension
  d <- length(mus)
  total_n <- n+warmup
  rvs <- matrix(0, nrow = total_n, ncol = d)

  for(i in 2:total_n){
    # Generate X_1
    rvs[i, 2] <- mus[2] + (rho * sigmas[2] / sigmas[1] * (rvs[i-1, 1] - mus[1])) + sigmas[2]
 * sqrt(1-rho^2) * rnorm(1, 0, 1)

    # Generate X_2
    rvs[i, 1] <- mus[1] + (rho * sigmas[1] / sigmas[2] * (rvs[i, 2] - mus[2])) + sigmas[1] *
 sqrt(1-rho^2) * rnorm(1, 0, 1)
  }

  return(rvs[(warmup+1):total_n, ])
}

mixture_binorm <- function(n, mus, sigmas, rhos, ws){
  modals <- length(mus[, 1])
  d <- length(mus[1, ])
  total_rvs <- array(0, c(n, d, modals))
  inds <- rbern(n, ws[2])

  #print(modals)
  #print(d)
  #print(inds)
  #print(total_rvs)

  for(i in 1:modals){
    total_rvs[,,i] <- binorm(n, mus[i, ], sigmas[i, ], rhos[i])
  }
  i <- 1
  #print(total_rvs)

  rvs <- matrix(0, ncol=d, nrow=n)
  #rvs[, 1] = rvs[, 2] <- c(1:n)
  for(i in 1:n){
    rvs[i, ] <- c(total_rvs[i,,inds[i]+1])
  }

  return(rvs)
}

mus <- matrix(c(1, 2,
                2, 1), ncol=2)
colnames(mus) <- c("mu_X_1", "mu_X_2")
rownames(mus) <- c("component_1", "component_2")

sigmas <- matrix(c(1, 2,
                   2, 1), ncol=2)
colnames(sigmas) <- c("sigma_X_1", "sigma_X_2")
rownames(sigmas) <- c("component_1", "component_2")

rhos <- c(0.4, 0.6)
rhos_m <- matrix(rhos, ncol=1)
```

```
colnames(rhos_m) <- c("rho")
rownames(rhos_m) <- c("component_1", "component_2")

ws <- c(0.4, 0.6)
ws_m <- matrix(ws, ncol=1)
colnames(ws_m) <- c("coefficient")
rownames(ws_m) <- c("component_1", "component_2")

print("The Prameters of The GMM")
print(mus)
print("---")
print(sigmas)
print("---")
print(rhos_m)
print("---")
print(ws_m)
print("--------------------------------")

#res <- binorm(10000, mus[1,], sigmas[1,], rhos[1])
#res <- binorm(10000, mus[2,], sigmas[2,], rhos[2])

res <- mixture_binorm(100, mus, sigmas, rhos, ws)
#res <- mixture_binorm(10000, matrix(c(1, 2, 2, 1), ncol=2), matrix(c(1, 2, 2, 1), ncol=2), c
(0.4, 0.6), c(0.4, 0.6))

print("Population Mean & Covariance")
print(cat("Mean of Y1: ", mean(res[, 1])))
print(cat("Mean of Y2: ", mean(res[, 2])))
print("Covariance Matrix")
print(cov(res))
print(cat("Number of data point: ", length(res[, 1])))

# Scatter
plot(res[,1], res[,2], main="Scatter Plot", xlab="Y1", ylab="Y2")

# Dendrogram
dg <- hclust(dist(scale(res), method = "euclidean"), method = "ward.D2")
plot(dg, hang=0.1, main="Cluster Dendrogram", sub=NULL, xlab="Points", ylab="Height")
```

Problem 3

Generate Data

```
gen_datas <- function(n){
  p_1 <- 0.6
  mu_1 <- 0
  sigma_1 <- 1
  mu_2 <- 3
  sigma_2 <- 1

  p <- runif(n, 0, 1)

  switch_func <- function(p){
    if(p < p_1){
      # Cluster 1
      return(rnorm(1, mu_1, sigma_1))
    }else{
      # Cluster 2
      return(rnorm(1, mu_2, sigma_2))
    }
  }
  x <- sapply(p, switch_func)

  return(x)
}

n <- 100
M <- 200
datas <- matrix(n, ncol=n, nrow=M)
for(i in 1:M){
 datas[i, ] <- gen_datas(n)
}

print(length(datas[, 1]))
print(length(datas[1, ]))
print(mean(datas[1, ]))

data <- gen_datas(n)

hist(data, breaks=50, freq = FALSE)
```

(a)

```
dis <- function(a, b){
  return(sum((a - b)^2))
}

kmean <- function(X, K, threshold=1e-6, max_iter=10){
  n <- length(X)
  d <- 1
  cent_idxs <- sample(1:n, K)
  cent_cord <- matrix(0, nrow=K, ncol=d)
  dis_table <- matrix(0, nrow=n, ncol=K)
  clustered <- array(0, c(n))
  tot_dis <- Inf

  for(k in 1:K){
    cent_cord[k] <- X[cent_idxs[k]]
  }

  for(iter in 1:max_iter){
    # Calculate distance between every pairs of cluster center and data point
    for(k in 1:K){
      dis_table[, k] <- sapply(X, dis, cent_cord[k])
    }
    # Assign the nearest cluster
    clustered <- apply(dis_table, 1, which.min)
    # Update the center
    for(k in 1:K){
      idxs <- which(clustered == k)
      cent_cord[k] <- mean(X[idxs])

      #print(idxs)
      #print(X[idxs])
    }
    # Calculate total distance
    temp_tot_dis <- sum(dis_table)
    if(abs(temp_tot_dis - tot_dis) < threshold){break}
    tot_dis <- temp_tot_dis

    #print(dis_table)
    #print(clustered)
    #print(tot_dis)
  }

  #print(clustered)
  #print(tot_dis)

  res <- structure(list(data=X, cluster=clustered, total_distance=tot_dis, cluster_centers=ce
nt_cord), class= "KMEAN_res")

  return(res)
}

kmean_sim200 <- function(data, K){return(kmean(data, K))}

kmean_model <- kmean_sim200(data, 2)

dt_1 <- data[which(kmean_model$cluster == 1)]
dt_2 <- data[which(kmean_model$cluster == 2)]
```

```
dt <- cbind(dt_1, dt_2)
colnames(dt) <- c("Cluster 1", "Cluster 2")

boxplot(dt, n=2, xlab="Clusters", ylab="Values", main="Box Plot of K-Means Clustering")
```

(b)

```
e_step <- function(ys, ws, mus, sigmas){
  k <- length(ws)
  n <- length(ys)
  likelihoods <- matrix(rep(0, k*n), nrow = n)
  weighted_likelihoods <- matrix(rep(0, k*n), nrow = n)

  # Evaluate the hidden variables
  for(j in 1:k){
    likelihoods[, j] <- sapply(ys, dnorm, mus[j], sigmas[j])
    weighted_likelihoods[, j] <- ws[j] * likelihoods[, j]
  }
  # gamma_i
  weighted_likelihoods <- weighted_likelihoods / rowSums(weighted_likelihoods)

  return(weighted_likelihoods)
}

m_step <- function(ys, ws, mus, sigmas, gammas){
  k <- length(ws)
  n <- length(ys)

  #Maximize the estimate
  for(j in 1:k){
    sum_gammas <- sum(gammas[, j])
    mus[j] <- sum(ys * gammas[, j]) / sum_gammas
    sigmas[j] <- sqrt(sum(gammas[, j] * (ys - mus[j])^2) / sum_gammas)
    ws[j] <- mean(gammas[, j])
  }

  return(rbind(ws, mus, sigmas))
}

em <- function(ys, k, threshold=1e-9, max_iter=201){
  mus <- runif(k)
  sigmas <- runif(k)
  ws <- rep(1/k, k)

  old_params <- rbind(ws, mus, sigmas)

  for(i in 1:max_iter){
    gammas <- e_step(data, ws, mus, sigmas)
    params <- m_step(data, ws, mus, sigmas, gammas)
    #print(gammas)

    # Update parameters
    ws <- params[1, ]
    mus <- params[2, ]
    sigmas <- params[3, ]

    # Until converge
    if(abs(mean(params - old_params)) < threshold){
      break
    }

    # Record old values
    old_params <- params
```

```
    #if(i %% 10 == 1){
    #  print(cat("Iter ", i))
    #  print(params)
    #}
  }

  dens_table <- matrix(0, nrow=length(ys), ncol=k)
  for(j in 1:k){
    dens_table[, j] <- dnorm(ys, mus[j], sigmas[j])
  }
  #print(dens_table)

  col_names <- c(1:k)
  col_names <- sapply(col_names, function(j){return(paste("Component ", toString(j)))})

  colnames(params) <- col_names

  clustered <- apply(dens_table, 1, which.max)
  res <- structure(list(data=ys, cluster=clustered, params=params, dens_table=dens_table), cl
ass= "EM-GMM_res")
  return(res)
}

em_sim200 <- function(data, K){return(em(data, K))}

K <- 2

em_gmm_model <- em_sim200(data, K)

print("------------------")

print(em_gmm_model$params)

dt_1 <- data[which(em_gmm_model$cluster == 1)]
dt_2 <- data[which(em_gmm_model$cluster == 2)]
dt <- cbind(dt_1, dt_2)
colnames(dt) <- c("Cluster 1", "Cluster 2")

boxplot(dt, n=2, xlab="Clusters", ylab="Values", main="Box Plot of EM-GMM Clustering")
```

Reference

- Gibbs Sampling from a Bivariate Normal Distribution (https://www.aptech.com/resources/tutorials/bayesian-fundamentals/gibbs-sampling-from-a-bivariate-normal-distribution/)
- 21.1 - Conditional Distribution of Y Given X (https://online.stat.psu.edu/stat414/lesson/21/21.1)
- Cross Validated - Deriving the conditional distributions of a multivariate normal distribution (https://stats.stackexchange.com/questions/30588/deriving-the-conditional-distributions-of-a-multivariate-normal-distribution)