# 0504 統計計算作業

108071601 賴冠維

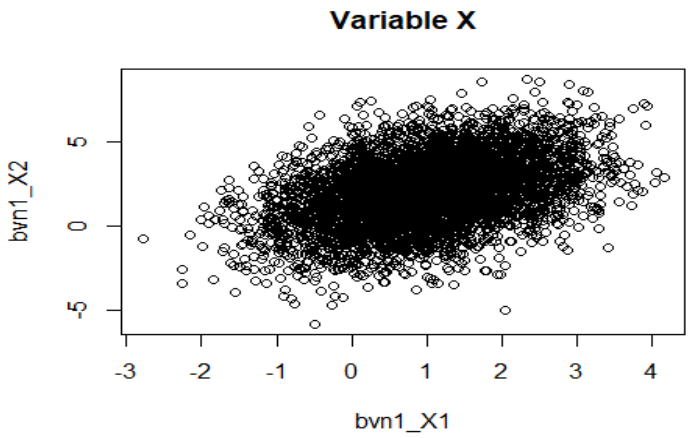
2021/5/4

## Problem 2

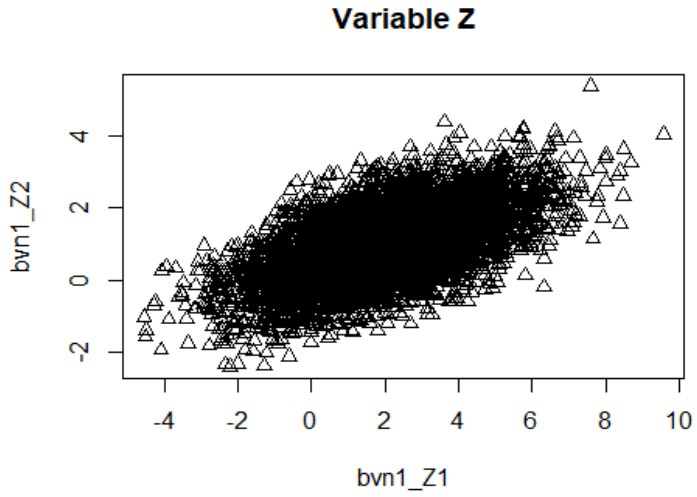
### 建立 X,Z 兩個二維常態分配

首先我們先建立 X、Z 變數，可見兩變數散佈圖如下：

變數 X




變數 Z




先模擬 Uniform(0,1)樣本，在從該樣本中抽樣，若大於 0.6 則從 Z 當中抽一個樣本，反之從 X 中抽一個，直到達到 100 個樣本，以此形成兩者的混合樣本。

可以看到混合樣本散佈圖如下：


(b)

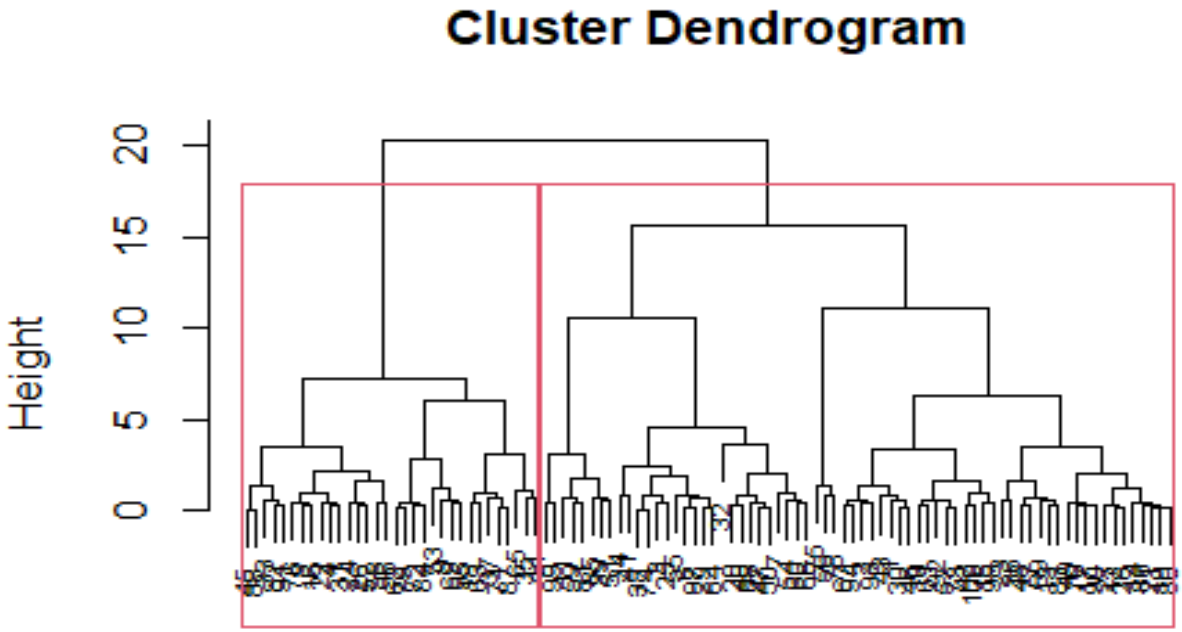
對混合樣本採用$HierarchicalClustering$,並選取$Ward.D2$作為合併標準，$Ward.D2$為最小組內變異法，其定義如下：

$$argmin(ESS) = ESS_1 + ESS_2 + \ldots + ESS_k$$

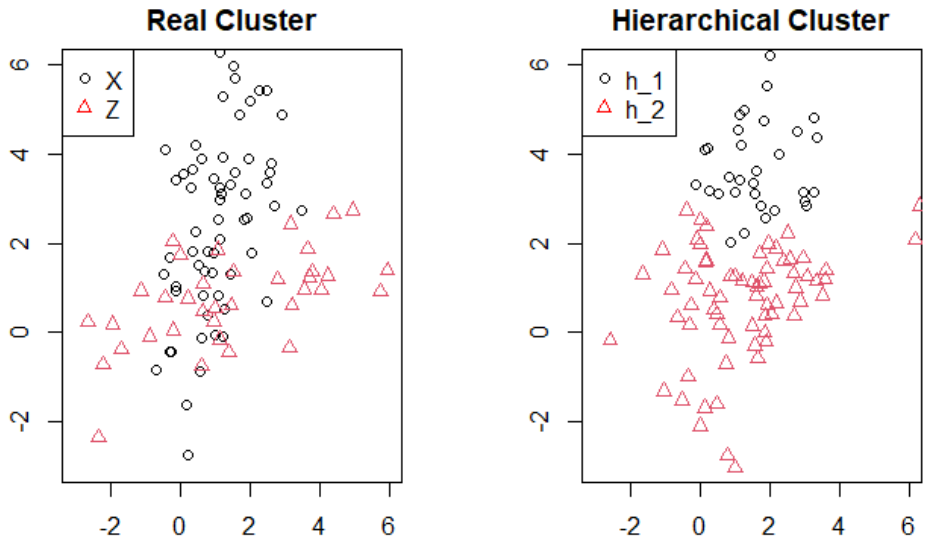
$$ESS_k = (x_{ij} - \overline{x_i})^T (x_{ij} - \overline{x_i})$$

˙ $x_{ij}$為第 i 群集裡面第 j 個觀測值

以最小組內變異增加量為原則，逐步加入其他樣本直至合併到最後一群結束。 我們預先知道樣本來自兩母體，因此我們選取群數為 2，結果如下：



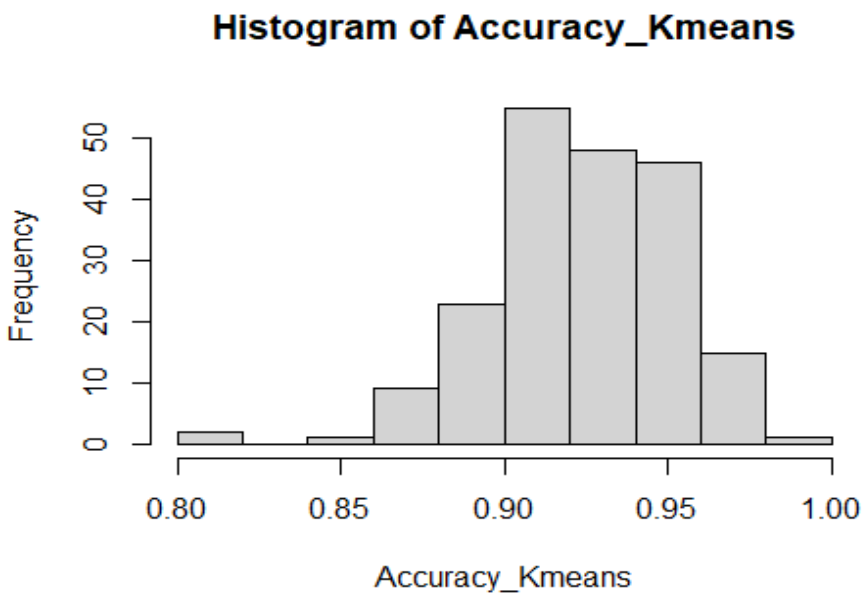

下圖左邊為真實兩變數在隨機樣本的分布，下圖右方為$Hierarchical\ Clustering$分群結果，可發現其分群的結果與真實的分群差異很大，可能來自於所選的合併標準是按照組內變異最小進行的，而真實樣本並不太符合兩組群組內變異較小的特性。




## Problem 3

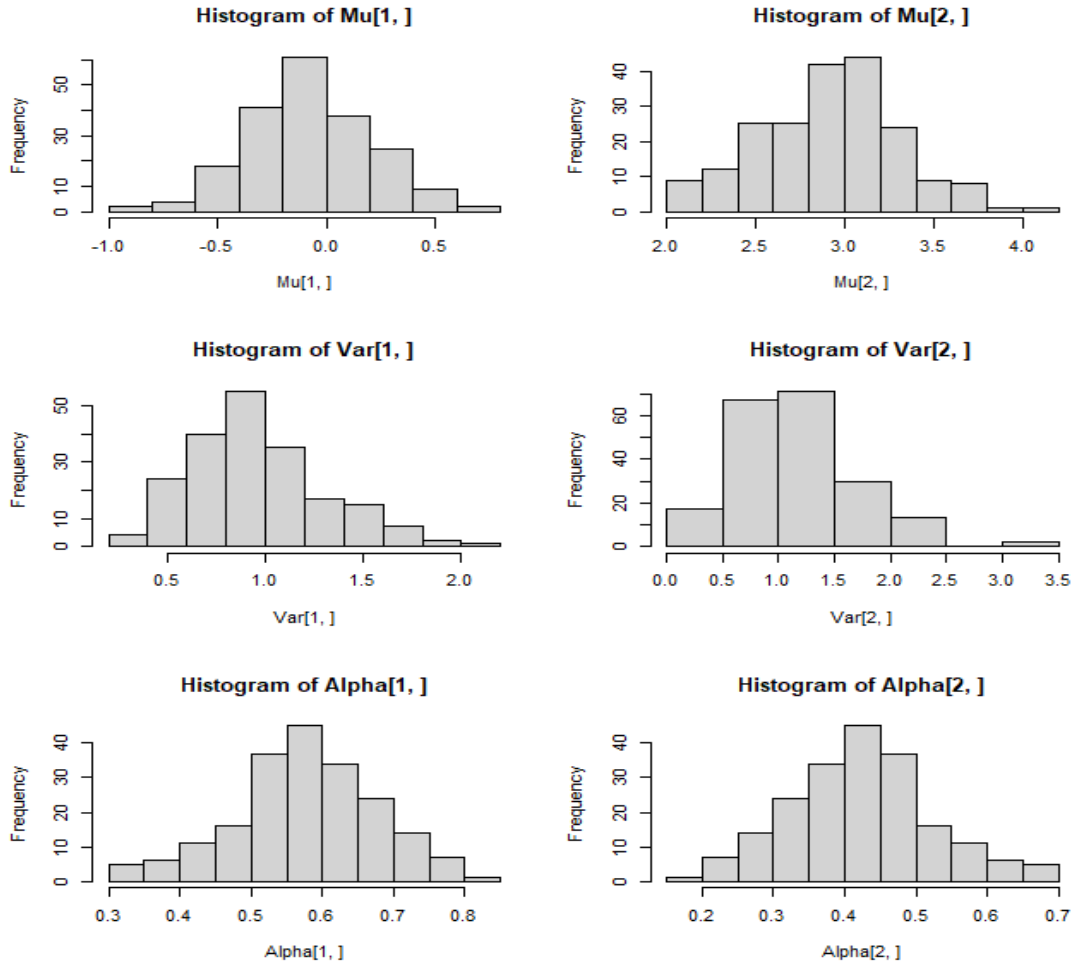
我們先重複$Problem\ 2$的作法，先從$N(0,1),N(3,1)$兩分配中，製造 100 筆隨機樣本，並重複 200 次，形成 200 組樣本數為 100，由$N(0,1),N(3,1)$組成的隨機樣本。再來對每組樣本進行$Kmeans$的分群，設定$center = 2$

我們將 200 組 Kmeans 分群結果，於其正確分群計算$Accuracy$，可得直方圖如下，可以發現大多數準確率都介在 85%至 95%之間，效果不錯。

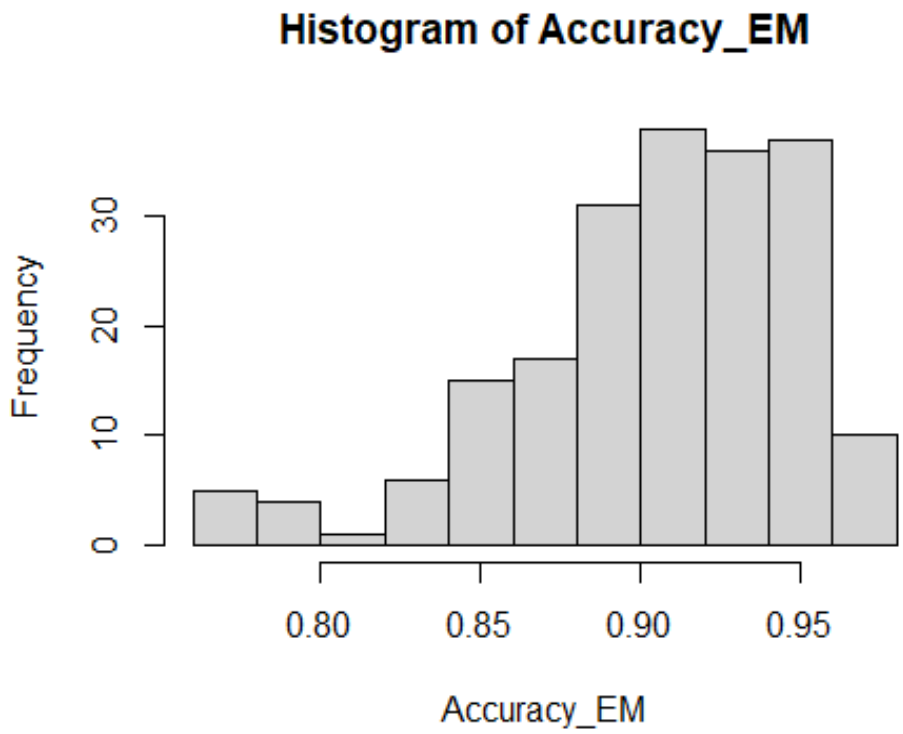



接著設定 EM 演算法的公式，我們將 Kmeans 分群結果，計算的平均數、期望值，作為起始值，在$E\ step$中計算$log\ Likelihood$以及兩群的機率，再來進入$M\ step$以 MLE 估計新的平均數、變異數，重複迭代此兩作法直到$log\ Likelihood$的改變量收斂至$10^{-4}$，結束演算法。

可看到 200 組收斂後所估計的$\mu$、$\sigma^2$、$\pi$所畫的直方圖，與我們真實分配$N(0,1)$、$N(3,1)$、$\pi = 0.6$進行比較，可發現非常接近真實分配的結果非常接近。




最後看到 EM 演算法做出來的直方圖正確率不錯，大部分同樣落在 85%至 95%之間。



## 附錄(Code)

```
library(MASS)
library(tidyverse)
mu <- c(1,2) # Mean
s1 = 1;s2=2;rho=0.4
sigma <- matrix(c(s1^2, s1*s2*rho, s1*s2*rho, s2^2),2) # Covariance matrix
bvn_X <- mvrnorm(5000, mu = mu, Sigma = sigma ) # from MASS package
colnames(bvn_X) <- c("bvn1_X1","bvn1_X2")
mu <- c(2,1) # Mean
s1 = 2;s2=1;rho=0.6
sigma <- matrix(c(s1^2, s1*s2*rho, s1*s2*rho, s2^2),2) # Covariance matrix
bvn_Z <- mvrnorm(5000, mu = mu, Sigma = sigma ) # from MASS package
colnames(bvn_Z) <- c("bvn1_Z1","bvn1_Z2")
plot(bvn_X)
points(bvn_Z,col=2)
set.seed(12345)
u = runif(n = 5000,min = 0,max = 1)
Y = matrix(NA, nrow = 100, ncol = 2)
set.seed(12345)
for (i in 1:100) {
  index = sample(1:5000,1)
  x = u[index]
  if (x<0.6) {
    Y[i,] = bvn_X[index,]
  }else{
    Y[i,] = bvn_Z[index,]
  }
}

index_x = which(Y[,1] %in% bvn_X[,1])
index_z = which(Y[,1] %in% bvn_Z[,1])
```

```
plot(bvn_X[index_x,],pch=1,main = "Real Cluster",
     xlab = "",ylab = "",xlim = c(-3,6),ylim = c(-3,6))
points(bvn_Z[index_z,],pch=2,col=2)
legend("topleft", legend=c("X", "Z"),
       col=c("black", "red"), pch=1:2, cex=1)
d <- dist(Y, method = "euclidean")
hc1 <- hclust(d, method = "ward.D2" )
plot(hc1, cex = 0.6)
rect.hclust(hc1, k = 2)
x = rect.hclust(hc1, k = 2)
h_1 = Y[x[[1]],]
h_2 = Y[x[[2]],]
plot(h_1,pch=1,main = "Hierarchical Cluster",
     xlab = "",ylab = "",xlim = c(-3,6),ylim = c(-3,6))
points(h_2,col=2,pch=2)
legend("topleft", legend=c("h_1", "h_2"),
       col=c("black", "red"), pch=1:2, cex=1)
X1 = rnorm(n = 5000,mean = 0,sd = 1)
X2 = rnorm(n = 5000,mean = 3,sd = 1)
Output = lapply(1:200, function(a){
  Y = matrix(NA, nrow = 100, ncol = 1)
  for (i in 1:100) {
    index = sample(1:5000,1)
    x = u[index]
    if (x<0.6) {
      Y[i] = X1[index]
    }else{
      Y[i] = X2[index]
    }
  }
  Y = Y %>% as.data.frame()
```

```
  Y$K = ifelse(Y[,1] %in% X1,"1","2") %>% as.matrix()
  kmeans <- kmeans(Y[,1], centers=2)
  kk = table(prob = kmeans$cluster,True = Y$K)
  if (kk[1,1]<kk[1,2]) {
    Y$Kmeans = ifelse(kmeans$cluster=="1","2","1")
  } else{
    Y$Kmeans = ifelse(kmeans$cluster=="1","1","2")
  }
  return(Y)
})
Accuracy_Kmeans = sapply(1:200, function(a){
  x = Output[[a]][,c("K","Kmeans")]
  xx = which(x$K==x$Kmeans) %>% length()
  xx/nrow(x)
})
hist(Accuracy_Kmeans)
e_step <- function(x, mu.vector, sd.vector, alpha.vector) {
  comp1.prod <- dnorm(x, mu.vector[1], sd.vector[1]) * alpha.vector[1]
  comp2.prod <- dnorm(x, mu.vector[2], sd.vector[2]) * alpha.vector[2]
  sum.of.comps <- comp1.prod + comp2.prod
  comp1.post <- comp1.prod / sum.of.comps
  comp2.post <- comp2.prod / sum.of.comps
  comp1.post.lik = alpha.vector[1]*comp1.post
  comp2.post.lik = alpha.vector[2]*comp2.post
  sum.of.post.lik <- comp1.post.lik + comp2.post.lik
  sum.of.comps.ln <- log(sum.of.comps, base = exp(1))
  sum.of.post.lik.ln <- log(sum.of.post.lik, base = exp(1))
  sum.of.comps.ln.sum <- sum(sum.of.comps.ln)+sum(sum.of.post.lik.ln)
  list("loglik" = sum.of.comps.ln.sum,
       "prob" = comp1.post,
       "posterior.df" = cbind(comp1.post, comp2.post))
```

```
}
m_step <- function(x, posterior.df) {
  comp1.n <- sum(posterior.df[, 1])
  comp2.n <- sum(posterior.df[, 2])
  comp1.mu <- 1/comp1.n * sum(posterior.df[, 1] * x)
  comp2.mu <- 1/comp2.n * sum(posterior.df[, 2] * x)
  comp1.var <- sum(posterior.df[, 1] * (x - comp1.mu)^2) * 1/comp1.n
  comp2.var <- sum(posterior.df[, 2] * (x - comp2.mu)^2) * 1/comp2.n
  comp1.alpha <- comp1.n / length(x)
  comp2.alpha <- comp2.n / length(x)
  list("mu" = c(comp1.mu, comp2.mu),
       "var" = c(comp1.var, comp2.var),
       "alpha" = c(comp1.alpha, comp2.alpha))
}
EM = lapply(1:200, function(a){
  t = Output[[a]]
  t.summary.df <- t %>%
    group_by(Kmeans) %>%
    summarize(mu = mean(V1), variance = var(V1), std = sd(V1), size = n())
  t.summary.df <- t.summary.df %>%
    mutate(alpha = size / sum(size))



  for (i in 1:50) {
    if (i == 1) {
      # Initialization
      e.step <- e_step(t$V1, t.summary.df[["mu"]], t.summary.df[["std"]],
                       t.summary.df[["alpha"]])
      m.step <- m_step(t$V1, e.step[["posterior.df"]])
      cur.loglik <- e.step[["loglik"]]
      loglik.vector <- e.step[["loglik"]]
```

```
    } else {
      # Repeat E and M steps till convergence
      e.step <- e_step(t$V1, m.step[["mu"]], sqrt(m.step[["var"]]),
                       m.step[["alpha"]])
      m.step <- m_step(t$V1, e.step[["posterior.df"]])

      # 紀錄 log-Likelihood
      loglik.vector <- c(loglik.vector, e.step[["loglik"]])

      loglik.diff <- abs((cur.loglik - e.step[["loglik"]]))
      if(loglik.diff < 1e-4) {
        break
      } else {
        cur.loglik <- e.step[["loglik"]]
      }
    }
  }
  return(list(e.step = e.step,
              m.step = m.step,
              loglik.vector = loglik.vector))
})
for (i in 1:200) {
  Output[[i]]$EM = ifelse(EM[[i]]$e.step$prob>0.5,"1","2")
}
Accuracy_EM = sapply(1:200, function(a){
  x = Output[[a]][,c("K","EM")]
  xx = which(x$K==x$EM) %>% length()
  xx/nrow(x)
})
hist(Accuracy_EM)
```