0504統計計算作業

108071601 賴冠維

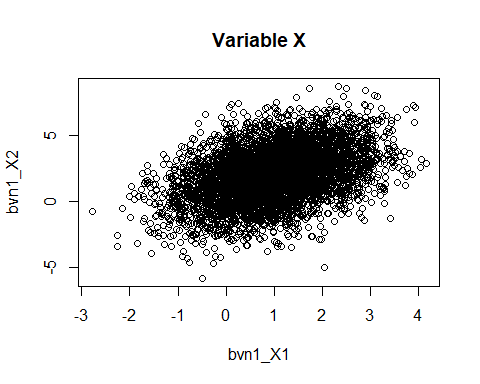
2021/5/4

## Problem 2

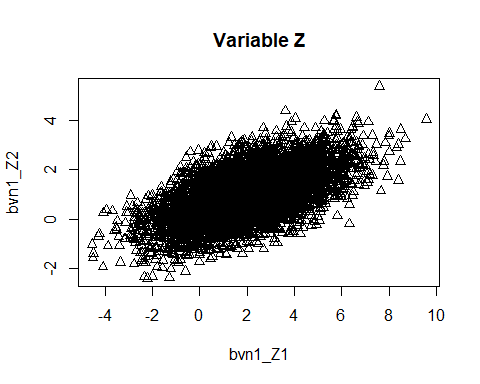
### 建立X,Z兩個二維常態分配

首先我們先建立X、Z變數，可見兩變數散佈圖如下：

變數X

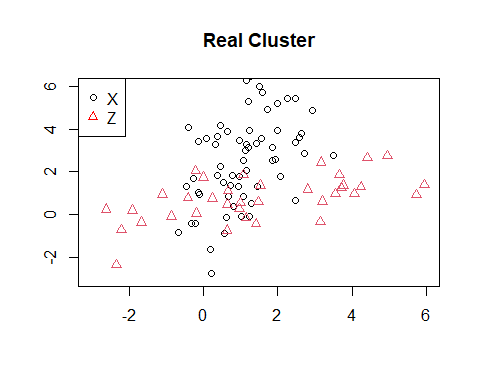


變數Z



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：

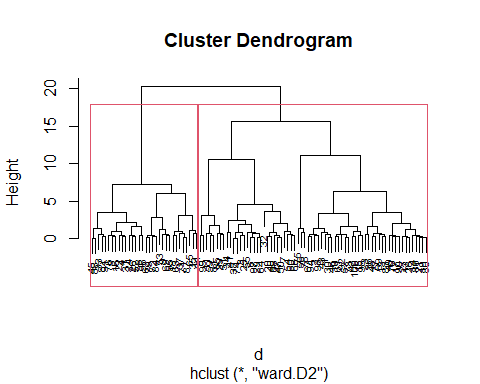


#### (b)

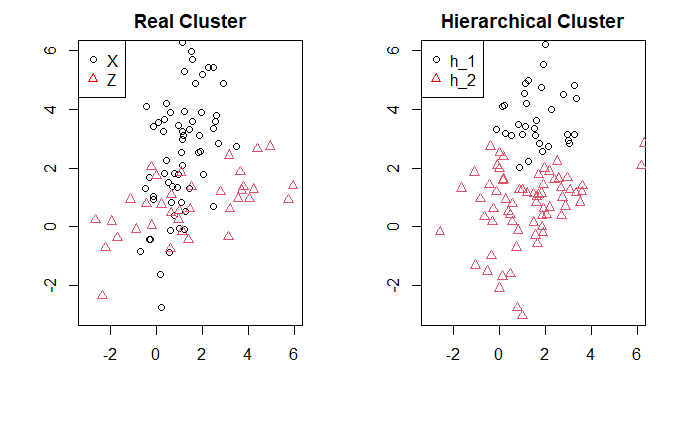
對混合樣本採用,並選取作為合併標準，為最小組內變異法，其定義如下：

˙ 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：



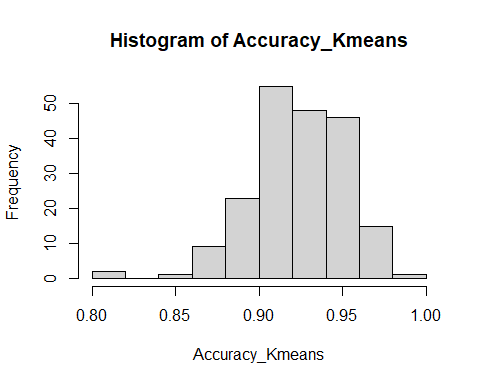
下圖左邊為真實兩變數在隨機樣本的分布，下圖右方為分群結果，可發現其分群的結果與真實的分群差異很大，可能來自於所選的合併標準是按照組內變異最小進行的，而真實樣本並不太符合兩組群組內變異較小的特性。



## Problem 3

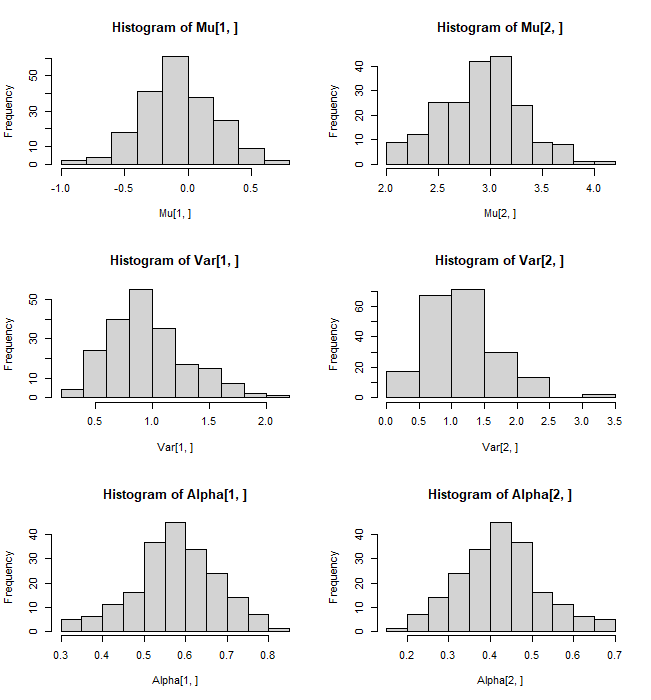
我們先重複的作法，先從,兩分配中，製造100筆隨機樣本，並重複200次，形成200組樣本數為100，由,組成的隨機樣本。再來對每組樣本進行的分群，設定

我們將200組Kmeans分群結果，於其正確分群計算，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。

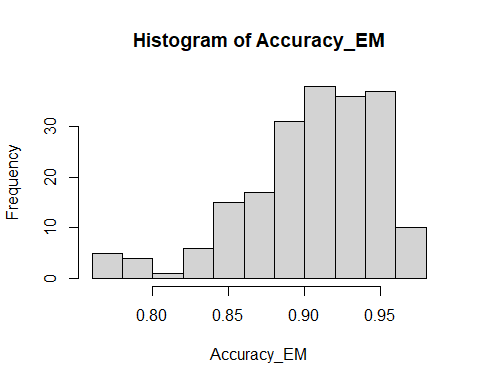


接著設定EM演算法的公式，我們將Kmeans分群結果，計算的平均數、期望值，作為起始值，在中計算以及兩群的機率，再來進入以MLE估計新的平均數、變異數，重複迭代此兩作法直到的改變量收斂至，結束演算法。

可看到200組收斂後所估計的所畫的直方圖，與我們真實分配 、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)