

2022년 2학기

# Computational Statistics

## HW#4

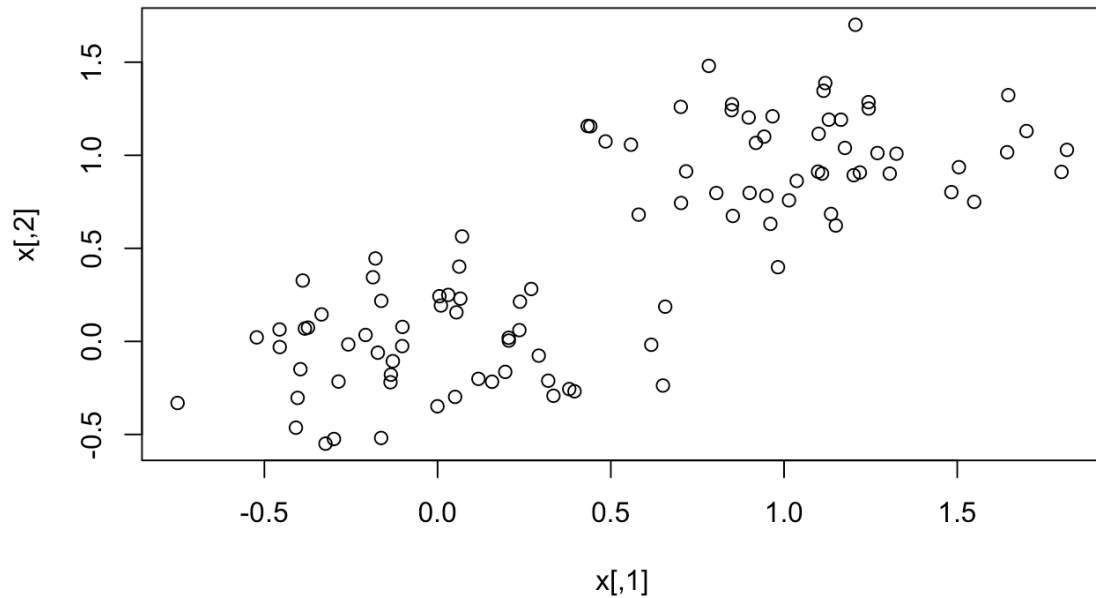
222STG10

김희숙

## Problem

### 1. Kmeans function 구현

#### X의 Graph

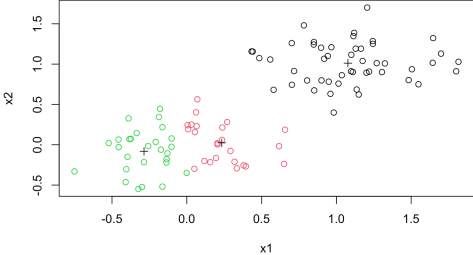
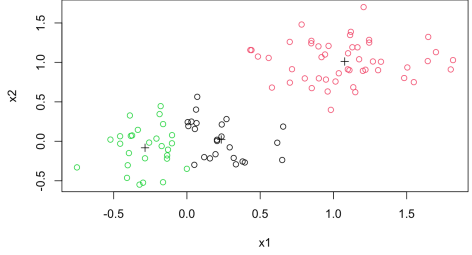


K=2

	myKmeans	kmeans
k=2		
maxiter=10 , nstart=1		
	cluster_mean: (-0.04114767, -0.03242972) (1.07717633, 1.01160708)  tot.withinss: 8.436569, 8.630755	cluster_mean: (1.07717633, 1.01160708) (-0.04114767, -0.03242972)  tot.withinss: 8.630755, 8.436569
niter	1	1
time	0.043	0.000162

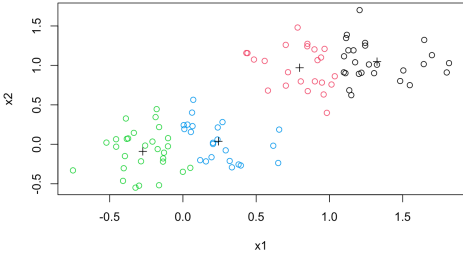
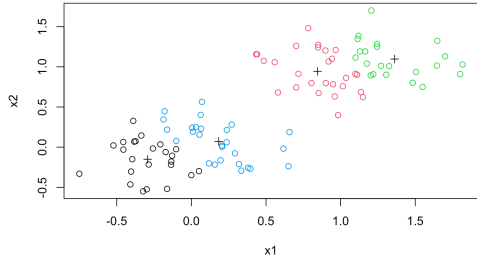
군집의 수를 2개로 clustering을 진행한 결과를 보면 두 함수로 생성된 군집이 center의 위치와 total withinss가 똑같은 형태로 구성되는 것을 볼 수 있다. 하지만 myKmeans의 시간이 훨씬 오래 걸린다.

### K=3

	myKmeans	kmeans
k=3		
maxiter=10 , nstart=10		
	cluster_mean: (1.0771763, 1.01160708) (0.2341462, 0.02353442) (-0.2858533, -0.08217562)  tot.withinss: 8.630755, 2.283761, 2.575162	cluster_mean: (0.2341462, 0.02353442) (1.0771763, 1.01160708) (-0.2858533, -0.08217562)  tot.withinss: 2.283761, 8.630755, 2.575162
niter	2	3
time	0.283	0.000884

군집의 개수가 3개인 경우에도 두 함수를 통해 clustering을 진행한 결과가 동일한 것을 알 수 있다. 다만 시행횟수에서 차이가 있음을 알 수 있는데, 이는 랜덤하게 초기 위치가 선정되는 과정에서 myKmeans가 clustering하기 좋은 위치부터 시작했다고 생각된다.

### K=4

	myKmeans	kmeans
k=4		
maxiter=10 , nstart=10		
	cluster_mean: (1.3251582, 1.04804832) (0.7968490, 0.97041263) (-0.2738366, -0.08988823) (0.2421258, 0.03751978)  tot.withinss: 2.82776, 2.323148, 2.729299, 2.140646	cluster_mean: (-0.2932244, -0.15012314) (0.8452025, 0.94223850) (1.3618715, 1.09674125) (0.1829205, 0.07218666)  tot.withinss: 1.976050, 2.877092, 2.228236, 2.951973

niter	5	3
time	0.508	0.001012

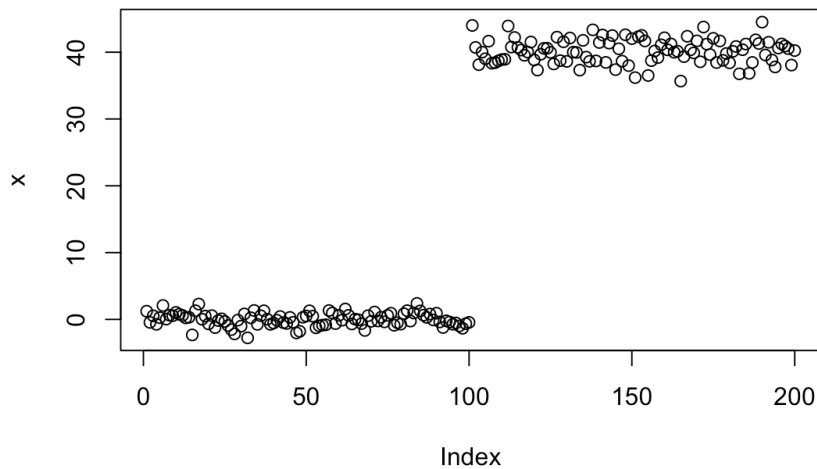
군집의 수를 4로 지정했을 때, 데이터가 밀접하게 위치한 경계에 대해서 다른 그룹을 부여하고 있는 것을 볼 수 있다. 두 결과를 보면 그룹 중심의 위치가 다른 것을 볼 수 있다. x1이 1.1, -0.1 부분에서 다르게 그룹을 배정한 것을 볼 수 있다. 시행 횟수는 myKmeans가 5회로 R에서 제공 중인 kmeans보다 2번 더 멤버쉽과 센터를 업데이트 하는 과정을 진행했다. 시간은 여전히 myKmeans가 오래걸린다.

## 2. GMM(Gaussian Mixture Model) function 구현

\*Code에서 사용된 변수명 의미: mu=평균, sigma=분산

### 1) well separated (pi=0.5)

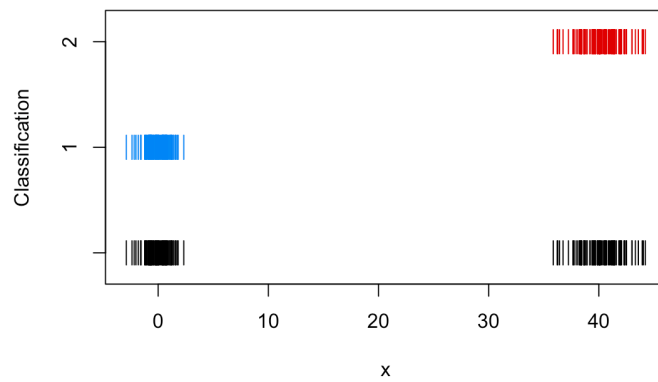
x의 Graph



x는  $N(0, 1)$ 과  $N(40, 4)$ 인 정규분포에서 각 100개씩 랜덤하게 추출하여 만든 data로, index가 1~100까지는  $N(0, 1)$ 분포에서 101~200까지는  $N(40, 4)$ 분포에서 발생한 데이터 이다.

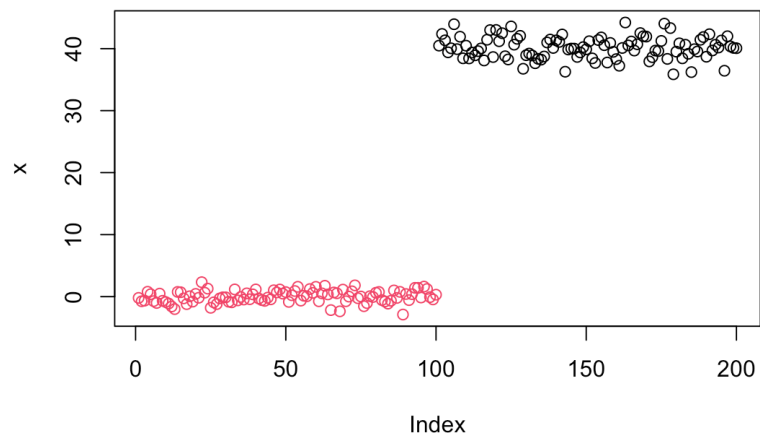
**Mclust**

k	df	log-likelihood	n	BIC	ICL
2	5	-477.6769	200	-981.8453	-981.8453



## Mygmm

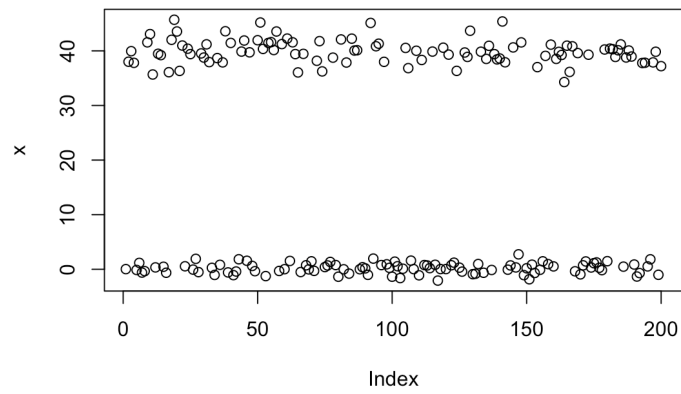
mu	sigma	log-likelihood
40.118186004, -0.009119946	3.15905, 0.9559238	20.06013



Mygmm의 결과를 보면 mu와 sigma를 추출한 분포와 유사하게 추정된 것을 알 수 있다. 또한 clustering결과를 시각화해보면 index 100을 기준으로 명확하게 분리가 된 것을 볼 수 있다.

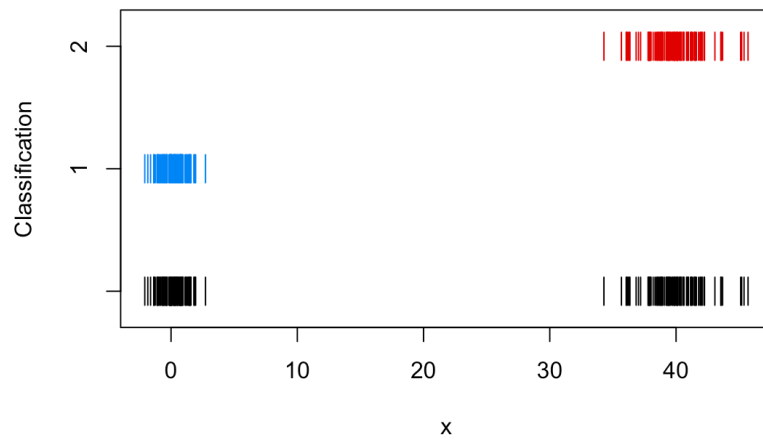
## x의 Graph2

다음으로 아까와 마찬가지로 방법으로 x를 발생시켜보았다. 단, 이전에는 Index 100을 기준으로 두 데이터가 나뉘었다면 인덱스 구분이 생기지 않게 더 자유로운 data를 통해 진행해보았다.



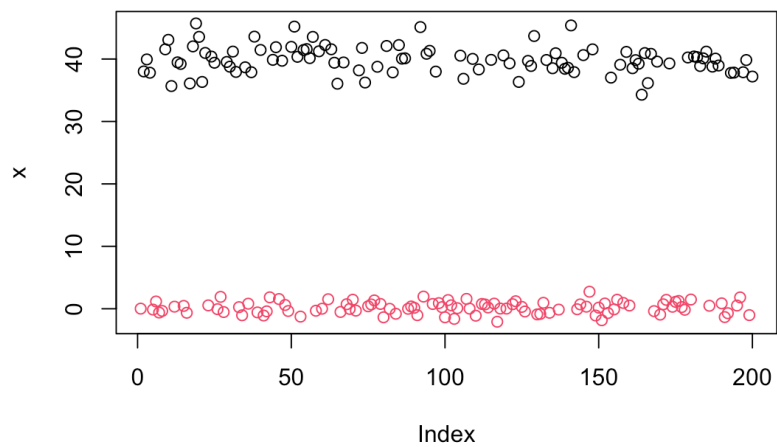
### Mclust

k	df	log-likelihood	n	BIC	ICL
2	5	-492.9708	200	-1012.433	-1012.433



### Mygmm

mu	sigma	log-likelihood
39.8749363, 0.1916755	4.701201, 0.8721931	19.26924



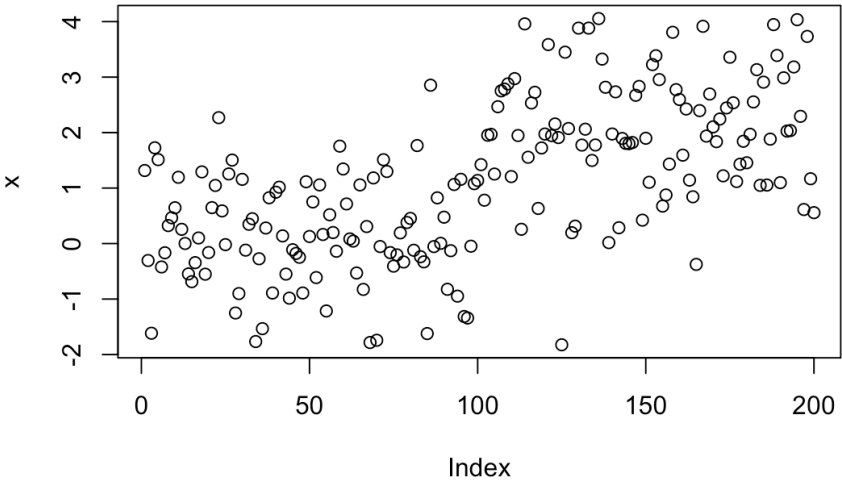
for문을 돌아가는 데이터 순서에 상관없이 clustering이 잘 되는 것을 확인할 수 있다.  $\mu$ 와  $\sigma$ 의 추정도 비슷하게 잘 추정하는 것으로 보인다.

실행 시간			
Mclust	Mygmm	Mclust_2	Mygmm_2
0.004325	0.126	0.004488	0.082

Mygmm의 시간이 훨씬 오래 걸린다.

### 2) 두 분포가 가까이 있는 경우

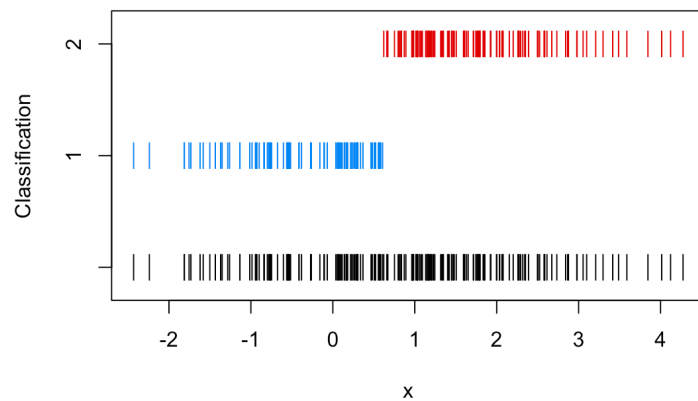
x의 Graph



x는 가까운 두 분포  $N(0, 1)$ 과  $N(2, 1)$ 인 정규분포에서 각 100개씩 랜덤하게 추출하여 만든 data로, index가 1~100까지는  $N(0, 1)$ 분포에서 101~200까지는  $N(2, 1)$ 분포에서 발생한 데이터이다.

#### Mclust

k	df	log-likelihood	n	BIC	ICL	time
2	5	-344.3929	200	-709.9791	-802.1871	0.004544



Clustering table을 보면 83, 117로 분포에 대한 군집화로 볼 때 몇개의 데이터에 대해서 다른 분포로 clustering되는 것을 볼 수 있었다.

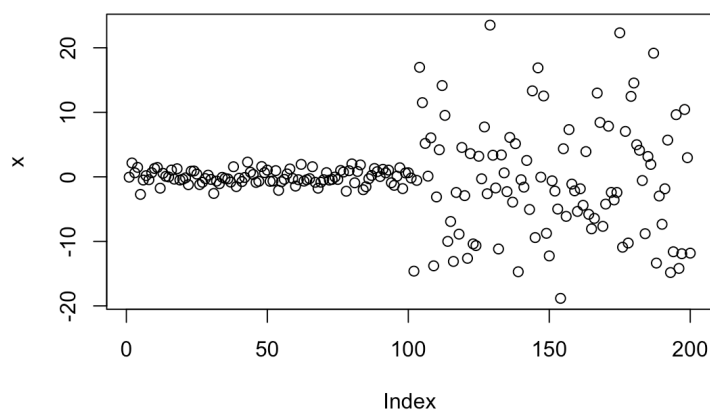
### Mygmm

mu	sigma	log-likelihood	time
0.8299132, 3.1910327	1.494224, 0.3463174	30.89151	19.978

평균과 분산 추정인 경우 실제 평균과 분산 비교를 했을 때 오차가 큰 것을 확인할 수 있다. 또한, clustering이 이루어지지 않았다.(maxiter=1000)

### 3) 평균이 같고 분산은 다른 경우

#### x의 Graph

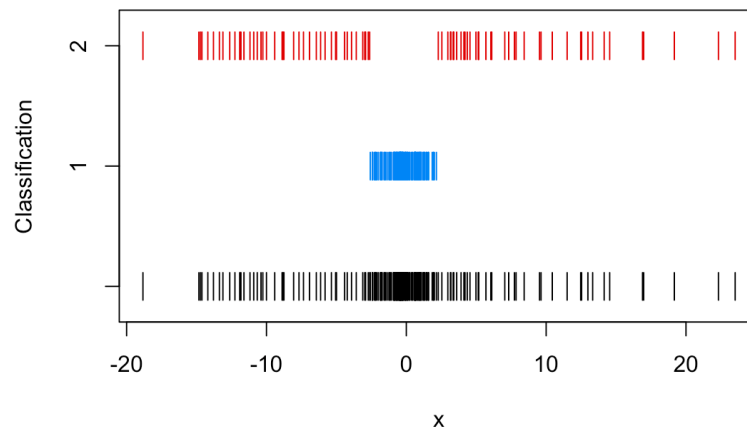


x는  $N(0, 1)$ 과  $N(0, 81)$ 인 정규분포에서 각 100개씩 랜덤하게 추출하여 만든 data로, index가 1~100까지는  $N(0, 1)$ 분포에서 101~200까지는  $N(0, 81)$ 분포에서 발생한 데이터 이다.

### Mclust



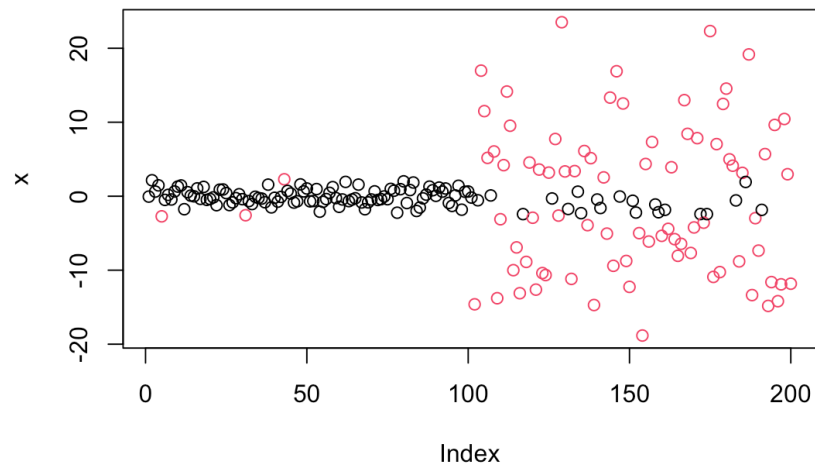
k	df	log-likelihood	n	BIC	ICL	time
2	5	-595.1434	200	-1216.778	-1274.04	0.005104



Clustering table을 보면 81, 119로 분포에 대한 군집화로 볼 때 몇개의 데이터에 대해서 다른 분포로 clustering되는 것을 볼 수 있었다.

### Mygmm

mu	sigma	log-likelihood	time
-0.1935885, -0.3785310	1.3373, 81.98512	14.06269	1.085



평균과 분산 추정의 경우 실제 평균과 분산 비교를 했을 때 비슷하게 잘 추정하는 것을 확인할 수 있다. 하지만 clustering된 결과를 시각화해보면, 분포별로 군집이 생성된 모습은 아닌 것을 볼 수 있다. Em알고리즘을 이용한 Gmm clustering의 경우 명확하게 층이 구분된 데이터에 대해서 clustering을 잘 수행하는 것으로 보인다.

## Code appendix

```
##### Kmeans #####
```

```
set.seed(1021)

x <- rbind(matrix(rnorm(100, sd = 0.3), ncol = 2),
            matrix(rnorm(100, mean = 1, sd = 0.3), ncol = 2))

plot(x)

mykmeans <- function(x,k,maxiter){

  niter<-1; cl = rep(NA,nrow(x)); dis = rep(NA,nrow(x)) #군집/거리저장공간

  init.mean <- x[sample(nrow(x), k, replace=FALSE), ];init.mean2 <- x[sample(nrow(x), k,
replace=FALSE), ]

  while(niter<=maxiter){

    tmp =numeric()

    for(i in 1:nrow(x)) {

      for(j in 1:k) tmp[j] = dist(rbind(x[i,],init.mean[j,]), method = "euclidean") #, method =
"euclidean"

      cl[i] = which.min(tmp); dis[i] = tmp[which.min(tmp)] #그룹배정/거리저장

    }

    # membership update

    for(i in 1:k) init.mean2[i,] = apply(x[which( cl == i),],2,mean) #같은 그룹 내 mean으로
init.mean 업데이트

    if (all.equal(init.mean,init.mean2) == TRUE) break

    init.mean = init.mean2; niter = niter+1

  }

  withinss = dis*dis #within-cluster sum of squares

  lst = list(cluster_membership = cl,

             cluster_mean = init.mean,

             tot.withinss = tapply(as.data.frame(cbind(withinss, cl))$withinss,
as.data.frame(cbind(withinss, cl))$cl, sum),

             niter = niter-1)

  return(lst)

}

myKmeans <- function(x,k,maxiter,nstart){

  for(i in 1:nstart) {

    assign(paste0("mycluster_",i),mykmeans(x,k,maxiter))

  }

  min = 10000000

  for(i in 1:nstart) {

    if(sum(get(paste0("mycluster_",i))$tot.withinss) <= min) {
```

```

    min = sum(get(paste0("mycluster_",i))$tot.withinss)
    num = i
    print(num)
  }
}

point_data=as.data.frame(get(paste0("mycluster_",num))$cluster_mean,make.names = TRUE)
plot_data=as.data.frame(cbind(x,get(paste0("mycluster_",num))$cluster_membership),make.names
= TRUE)

plot(plot_data$V1, plot_data$V2, col=plot_data$V3, xlab="x1", ylab="x2")
points(point_data$V1, point_data$V2,type="p",pch=3)

return(get(paste0("mycluster_",num)))
}

```

myKmeans(x,k=4, maxiter=10, nstart=10) #값이 조금씩 달라서 plot 그려보면 괜찮음

```
km=kmeans(x, 4, iter.max = 10,nstart = 10)
```

```
km
```

```
km_plot_data = as.data.frame(cbind(x,km$cluster,make.names = TRUE))
```

```
plot(km_plot_data$V1, km_plot_data$V2, col=km_plot_data$V3, xlab="x1", ylab="x2")
```

```
km_point_data=as.data.frame(km$centers,make.names = TRUE)
```

```
points(km_point_data$V1, km_point_data$V2,type="p",pch=3)
```

```
system.time(myKmeans(x,k=4, maxiter=10, nstart=10))
```

```
system.time(for(i in 1:1000){km=kmeans(x, 4, iter.max = 10,nstart = 10)})
```

```
##### GMM #####
```

```
x<-as.matrix(c(rnorm(n=100,0,1),rnorm(n=100,40,2))); plot(x)
```

```
mgm <- mygmm(x,k=2,epsilon=10^(-1000),maxiter=1000); mgm
```

```
plot(x, col=mgm$group_num) # 시각화
```

# 섞여있는 데이터에 대해서 실행

```
x<-as.matrix(sample(as.matrix(c(rnorm(n=100,0,1),rnorm(n=100,40,2))), 200, replace = FALSE))
```

```
mgm2 <- mygmm(x,k=2,epsilon=10^(-1000),maxiter=1000)
```

```
mgm2
```

```
plot(x, col=mgm2$group_num) # 시각화
```

```
multi_norm <- function(x,mu,sigma,p){
```

```
  1/((2*pi)^(p/2)*sqrt(abs(det(sigma))))*exp((-1/2)*t(x-mu)%*%solve(sigma)%*(x-mu))
```

```
}
```

```
mygmm <- function(data,k,epsilon,maxiter){
```

```

niter<-1;err<-1; #threshold<-10^(-10);
num_data <- nrow(data); num_var <- ncol(data)
# 초기값 설정

mu <- matrix(0, k, num_var); sigma <- matrix(0, num_var, num_var*k)
div_group <- sample(num_data,k); mu <- as.matrix(data[div_group,])
for(i in 1:k){
  start <- ((num_data/k)*(i-1))+1
  end <- ((num_data/k)*(i))
  if(num_var == 1) sigma[,i] <- sd(data[start:end])
  else sigma[(num_var*(i-1)+1):(num_var*i)] <- cov(data[start:end,])
}
#print(mu)
p <- rep(0.5, k)
pr_x <- matrix(0,num_data,1); pr_xi <- matrix(0,num_data,k)
g_num <- c();last_E <- 0
likeli <- sum( p*dnorm(data,mu[1,],sigma[,1]) + (1-p)*dnorm(data,mu[2,],sigma[,2]) )

while(niter<=maxiter && err >= epsilon ){ # && err >= epsilon
  for(i in 1:num_data){
    for(j in 1:k){
      pr_xi[i,j] <-
multi_norm(as.matrix(data[i,]),as.matrix(mu[j,]),as.matrix(sigma[(num_var*(j-
1)+1):(j*num_var)]),num_var) * p[j]
    }
    pr_x[i] <- sum(pr_xi[i,])
  }
  pr_g <- matrix(0, num_data, k)
  for(i in 1:num_data) for(j in 1:k) pr_g[i,j] <- as.matrix(pr_xi[i,j]/pr_x[i])
  E <- sum(log(pr_x))
  if(E == last_E){
    for(i in 1:num_data) g_num[i] <- which.max(pr_g[i,])
    break
  }
  mu <- matrix(0, k, num_var); sigma <- matrix(0, num_var, num_var*k)
  for(j in 1:k){
    pr <- sum(pr_g[,j])
    p[j] <- pr/num_data
    if(num_var == 1){
      mu[j,] <- sum(data*pr_g[,j])/pr
      sigma[,j] <- sum(pr_g[,j]*((data-mu[j,])^2))/pr
    }else{

```

```

mu[j,] <- mu[j,] + (t(pr_g[,j])%*%as.matrix(data))/p
for(i in 1:num_data){
  sigma[(num_var*(j-1)+1):(num_var*j)] <- sigma[(num_var*(j-1)+1):(num_var*j)] +
pr_g[i,j] * (t((data[i,]-t(mu[j,]))) %*% (data[i,]-t(mu[j,]))) / pr
}
}
}
likeli_new <- sum( (1-p)*dnorm(data,mu[1,],sigma[,1]) + p*dnorm(data,mu[2,],sigma[,2]) )
err <- abs(likeli_new-likeli); likeli <- likeli_new; last_E <- E
niter = niter+1
}
z <- list(mu = mu, sigma = sigma, group_num = g_num,likeli=likeli)
return(z)
}

```

```
library (mclust)
```

```
mc <- Mclust(x,2) #클러스터개수 2개
```

```
summary(mc)
```

```
plot(mc, what = "classification")
```

```
system.time(for(i in 1:1000){Mclust(x,2)})
```

```
system.time(mygmm(x,k=2,epsilon=10^(-1000),maxiter=1000))
```

```
x<-as.matrix(c(rnorm(n=100,0,1),rnorm(n=100,2,1))) ;plot(x)
```

```
mgm <- mygmm(x,k=2,epsilon=10^(-1000),maxiter=1000); mgm
```

```
plot(x, col=mgm$group_num)
```

```
system.time(mygmm(x,k=2,epsilon=10^(-1000),maxiter=1000))
```

```
mc <- Mclust(x,2) #클러스터개수 2개
```

```
summary(mc)
```

```
plot(mc, what = "classification")
```

```
system.time(for(i in 1:1000){Mclust(x,2)})
```

```
x<-as.matrix(c(rnorm(n=100,0,1),rnorm(n=100,0,9))); plot(x)
```

```
mgm <- mygmm(x,k=2,epsilon=10^(-1000),maxiter=1000); mgm
```

```
plot(x, col=mgm$group_num)
```

```
system.time(mygmm(x,k=2,epsilon=10^(-1000),maxiter=1000))
```

```
mc <- Mclust(x,2) #클러스터개수 2개
```

```
summary(mc)
```

```
plot(mc, what = "classification")
```

```
system.time(for(i in 1:1000){Mclust(x,2)})
```