library(rstan); options(mc.cores = parallel::detectCores())

require(loo)

library(jagsUI)

require(mclust)

require(flexmix)

library(bayesmix)

y=c(9.172, 9.35, 9.483, 9.558, 9.775, 10.227, 10.406, 16.084, 16.17, 18.419, 18.552, 18.6, 18.927, 19.052, 19.07, 19.33, 19.343, 19.349, 19.44, 19.473, 19.529, 19.541, 19.547, 19.663, 19.846, 19.856, 19.863, 19.914, 19.918, 19.973, 19.989, 20.166, 20.175, 20.179, 20.196, 20.215, 20.221, 20.415, 20.629, 20.795, 20.821, 20.846, 20.875, 20.986, 21.137, 21.492, 21.701, 21.814, 21.921, 21.96, 22.185, 22.209, 22.242, 22.249, 22.314, 22.374, 22.495, 22.746, 22.747, 22.888, 22.914, 23.206, 23.241, 23.263, 23.484, 23.538, 23.542, 23.666, 23.706, 23.711, 24.129, 24.285, 24.289, 24.368, 24.717, 24.99, 25.633, 26.96, 26.995, 32.065, 32.789, 34.279)

N=82

set.seed(1234)

**#**

**# mclust and flexmix**

**#**

summary(Mclust(y),parameters=T)

# integrated complete-data likelihood

plot(mclustICL(y))

summary(Mclust(y, G=4),parameters=T)

summary(Mclust(y, G=5),parameters=T)

summary(Mclust(y, G=6),parameters=T)

summary(flexmix(formula = y ~ 1, k = 4))

summary(flexmix(formula = y ~ 1, k = 5))

summary(flexmix(formula = y ~ 1, k = 6))

**#**

**# rstan using ordered vector for group means**

**#**

model <- ' data { int N;

vector[N] y;

int K;}

parameters { ordered[K] mu;

vector<lower = 0>[K] sigma;

simplex[K] pi;}

model { vector[K] contributions;

// priors

mu ~ normal(0, 10);

sigma ~ cauchy(0, 2);

pi ~ dirichlet(rep\_vector(2, K));

// likelihood

for(i in 1:N) { for(k in 1:K) {contributions[k] = log(pi[k]) + normal\_lpdf(y[i] | mu[k], sigma[k]);}

target += log\_sum\_exp(contributions); }}

generated quantities {

real tLL;

vector[K]conts[N];

vector[N] log\_lik;

for (i in 1:N) { for (k in 1:K) {

conts[i, k] = exp(log(pi[k]) + normal\_lpdf(y[i] | mu[k], sigma[k]));

log\_lik[i] =log(sum(conts[i,]));}}

tLL = sum(log\_lik[]);}'

**#**

**# Estimation, K=4, 5 and 6 groups compared**

**#**

pars= c("mu","sigma","pi","tLL","log\_lik")

D = list(N= N, y = y, K = 4)

stanfit4 = stan(model\_code=model, pars=pars, data=D, iter=50000, chains=2, seed="1234",boost\_lib = NULL)

print(stanfit4)

print(loo(extract\_log\_lik(stanfit4)))

D = list(N= N, y = y, K = 5)

stanfit5 = stan(model\_code=model, pars=pars, data=D, iter=10000, chains=2, seed="1234")

print(stanfit5)

print(loo(extract\_log\_lik(stanfit5)))

D = list(N= N, y = y, K = 6)

stanfit6 = stan(model\_code=model, pars=pars, data=D, iter=10000, chains=2, seed="1234")

print(stanfit6)

print(loo(extract\_log\_lik(stanfit6)))

**#**

**# jagsUI – Normal LKD**

**#**

cat("model { for (i in 1:N){ # conditional likelihood

y[i] ~ dnorm(m0[S0[i]], psi0[S0[i]])

# individual latent membership indicators (conditional likelihood)

S0[i] ~ dcat(P0[1:K])

ynew[i] ~ dnorm(m0[S0[i]],psi0[S0[i]])

for (j in 1:K) {d0[i,j] <- equals(S0[i],j)}

# exceedance check

exc[i] <- step(ynew[i]-y[i])

log\_lik[i] <- log(sum(L[i,]))}

P0 ~ ddirch(alpha[]); # prior for mixing proportion

# prior on unconstrained means

for (j in 1:K) { m0[j] ~ dnorm(25, 0.01) T(9.2,34.3)

# independent or hierarchical prior on group precisions

# psi0[j] ~ dgamma(1,0.01)

psi0[j] ~ dgamma(a.psi,b.psi)

# prior Dirichlet weights

alpha[j] <- 2.5

for (i in 1:N) { L[i,j] <- exp(log(P[j])+0.5\*log(psi[j])-0.919-0.5\*psi[j]\*pow(y[i]-mu[j],2))

# conditional allocation probabilities

rho[i,j] <- L[i,j]/sum(L[i,])

# entropy

ent1[i,j] <- equals(S[i],j)\*log(rho[i,j])

ent2[i,j] <- rho[i,j]\*log(rho[i,j])}}

Ent[1] <- -2\*sum(ent1[1:N,1:K])

Ent[2] <- -2\*sum(ent2[1:N,1:K])

tLL <- sum(log\_lik[])

# hyperparameters, hierarchical prior on precisions

a.psi ~ dexp(1)

b.psi ~ dexp(1)

# Processing to obtain identifiable groups, using ranks of unconstrained means

rank <- rank(m0)

# relabelled weights, means, precisions, variances

for (j in 1:K) {P[j] <- sum(P0prod[j,])

mu[j] <- sum(m0prod[j,])

psi[j] <- sum(psi0prod[j,])

s2[j] <- 1/psi[j]

for (k in 1:K) {P0prod[j,k] <- P0[k]\*equals(rank[k],j)

m0prod[j,k] <- m0[k]\* equals(rank[k],j)

psi0prod[j,k] <- psi0[k]\* equals(rank[k],j)}}

# relabelled allocation indicators

for (i in 1:N) { S[i] <- sum(dcat[i,])

for (j in 1:K) {d[i,j] <- sum(d0prod[i,j,])

dcat[i,j] <- j\*d[i,j]

for (k in 1:K) {d0prod[i,j,k] <- d0[i,k]\*equals(rank[k],j)}}}}

", file="mixnorm.jag")

############################################

**# initial values and estimation, assuming K=4 groups**

K=4

D <- list(N= N, y = y, K = K)

set.seed(1234)

inits <- function(){list(m0=rnorm(K,25,0.01),psi0=rexp(K,1))}

pars <- c("P","mu","psi","s2","tLL","S","Ent","log\_lik")

jagsmix <- autojags(D, inits, pars, model.file="mixnorm.jag",2, n.adapt=100, iter.increment=2500, n.burnin=500,Rhat.limit=1.1, max.iter=50000,seed=1234)

jagsmix$summary

**# estimate loo-IC, 2 chains, 1000 samples**

samps <- as.array(jagsmix$samples)

LL <- matrix(NA,2000,82)

for (i in 1:1000){for (j in 1:82) {LL[i,j] <- samps[i,4\*K+N+2+j,1]; LL[i+1000,j] <- samps[i,4\*K+N+2+j,2]}}

loo(LL)

#####################################

**# initial values and estimation, assuming K=5 groups**

K=5

D <- list(N= N, y = y, K = K)

set.seed(1234)

inits <- function(){list(m0=rnorm(K,25,0.01),psi0=rexp(K,1))}

pars <- c("P","mu","psi","s2","tLL","S","Ent","log\_lik")

jagsmix <- autojags(D, inits, pars, model.file="mixnorm.jag",2, n.adapt=100, iter.increment=1000, n.burnin=500,Rhat.limit=1.1, max.iter=50000,seed=1234)

jagsmix$summary

**# estimate loo-IC, 2 chains, 1000 samples**

samps <- as.array(jagsmix$samples)

LL <- matrix(NA,2000,82)

for (i in 1:1000){for (j in 1:82) {LL[i,j] <- samps[i,4\*K+N+2+j,1]; LL[i+1000,j] <- samps[i,4\*K+N+2+j,2]}}

loo(LL)

#################################

**# initial values and estimation, assuming K=6 groups**

K=6

D <- list(N= N, y = y, K = K)

set.seed(1234)

inits <- function(){list(m0=rnorm(K,25,0.01),psi0=rexp(K,1))}

pars <- c("P","mu","psi","s2","tLL","S","Ent","log\_lik")

jagsmix <- autojags(D, inits, pars, model.file="mixnorm.jag",2, n.adapt=100, iter.increment=1000, n.burnin=500,Rhat.limit=1.1, max.iter=50000,seed=1234)

jagsmix$summary

**# estimate loo-IC, 2 chains, 1000 samples**

samps <- as.array(jagsmix$samples)

LL <- matrix(NA,2000,82)

for (i in 1:1000){for (j in 1:82) {LL[i,j] <- samps[i,4\*K+N+2+j,1]; LL[i+1000,j] <- samps[i,4\*K+N+2+j,2]}}

loo(LL)

####################################

**# Assess predictive exceedance, K=4 groups**

K=4

D <- list(N= N, y = y, K = K)

set.seed(1234)

inits <- function(){list(m0=rnorm(K,25,0.01),psi0=rexp(K,1))}

pars <- c("exc","ynew")

jagsmix <- autojags(D, inits, pars, model.file="mixnorm.jag",2, n.adapt=100, iter.increment=1000, n.burnin=500,Rhat.limit=1.1, max.iter=100000,seed=1234)

jagsmix$summary

**# exceedance probabilities**

exc.mn=apply(jagsmix$sims.list$exc,2,mean)

head(exc.mn)

tail(exc.mn)