MATH 598: TOPICS IN STATISTICS

BAYESIAN MIXTURE MODELLING OF THE GALAXY DATA

The finite mixture model with K components has density

$$f(y;\omega,\theta,K)=\sum_{k=1}^{K}\omega_k f_k(y;\theta_k)$$

where θ_k are the parameters for component density k, and $0<\omega_k<1, k=1,\ldots,K$ with $\sum_{k=1}^{K}\omega_k=1$. Most typically the component densities are from the same parametric location-scale family (say Gaussian) which differ in location and scale. The model can be represented by introducing latent component indicators $z_1,\ldots,z_n$, so that

$$f(y_i;\omega,\theta,K)=\sum_{k=1}^{K}f(y|Z_i=k;\boldsymbol{\theta})\Pr[Z_i=k]$$

and $\Pr[Z_i=k]=\omega_k$ for $k=1,\ldots,K$.

For exchangeable data $(y_1,\ldots,y_n)$ from this model, the likelihood arising from this model does not permit analytical calculation of the posterior distribution, so MCMC is commonly used.

- If K is **known**, MCMC is typically implemented using a completed data model
  - Auxiliary variables $z_1,\ldots,z_n$,
    $$f(y,z;\omega,\theta,K)=\omega_z f_z(y;\theta_z)\qquad z\in\{1,2,\ldots,K\}$$
  - Gibbs sampler strategy updates the $\mathbf{z}$ and (ω,θ) from their full conditional posterior distributions.
  - Conditional on (ω,θ), the posterior distribution for each z_i is a discrete distribution on $\{1,2,\ldots,K\}$, whereas conditional on the $\mathbf{z}$, the posterior for the mixture weights ω is a density on the K dimensional simplex, and the posterior for the components of θ_k proceeds using only those y_i for which $z_i=k$, for $k=1,\ldots,K$.
- If K is **unknown**, then transdimensional MCMC is also needed.
  - A discrete prior on K, typically on some finite set, completes the posterior specification, and yields a posterior distribution $\pi_n(K,\theta^{(K)},\omega^{(K)})$.
  - In this model, dimension-changing moves correspond to changing the value of K, and in order to construct an efficient and tunable MCMC algorithm, moves which perturb K by ±1 are typically considered.
  - For example, moves of type $K\longrightarrow K+1$ are termed "birth" moves, and $K\longrightarrow K-1$ are termed "death" moves.
  - The forward/reverse move pairs are then births from K and deaths from $K+1$.
  - Inference about the parameter K may be of primary interest in a given application.

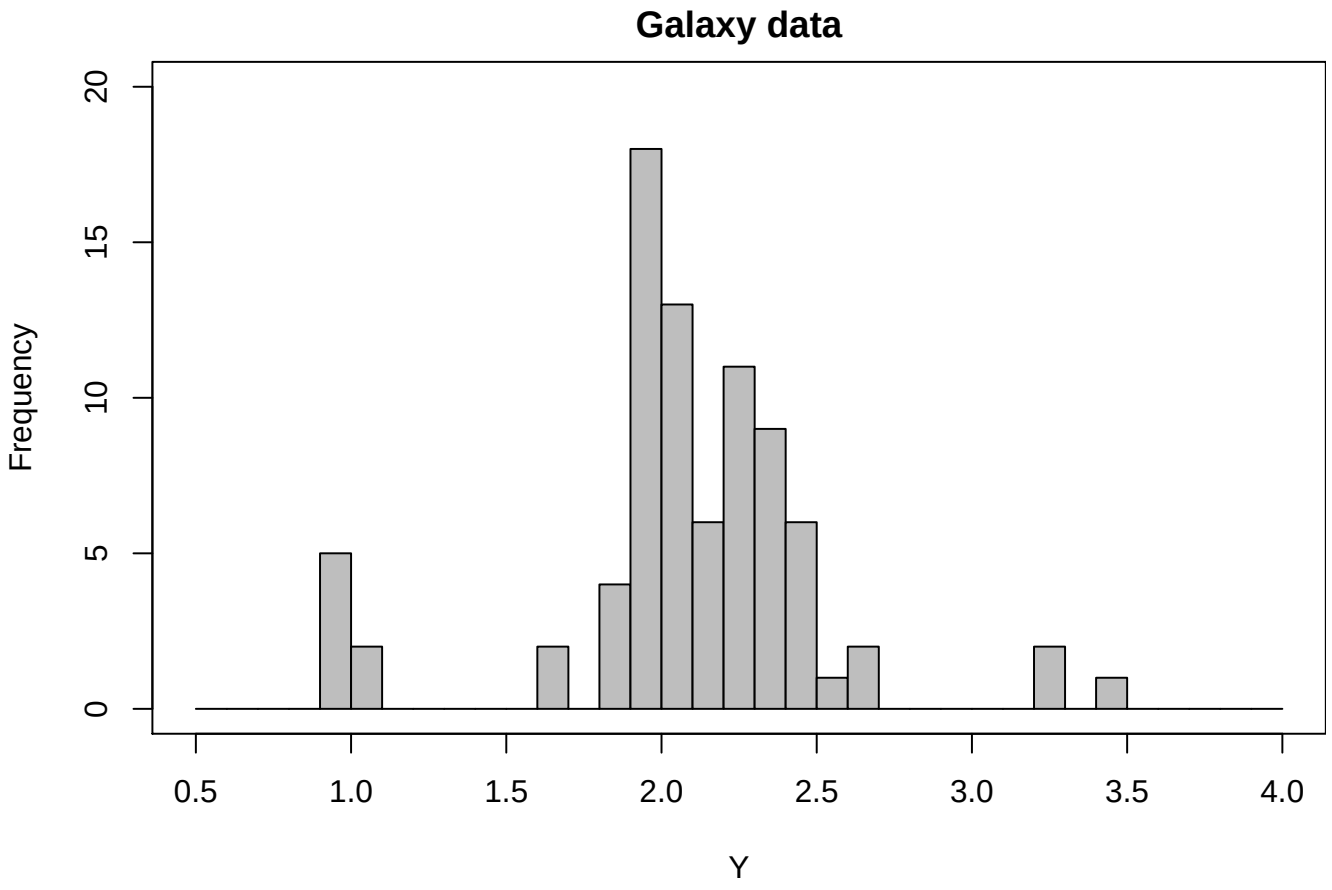
Normal Model: For a Normal mixture model:

- If y is one dimensional, each component density has two parameters, so a birth/death pair requires the introduction of three latent variables $\mathbf{u}$, a new mean and variance, and also a parameter to introduce a new component weight.
- If y is d-dimensional, $d+d(d+1)/2+1$ new scalar parameters are needed, that is, a new mean and covariance matrix, and a new component weight parameter.
- **New location/scale parameters:** Either these new parameters are generated independently of the current model parameters from proposal distribution, or a subset of the current components are selected and used to generate a new additional component.
- **New weight parameters:** A random split or rescaling of the currently existing weights is used.

Galaxies Data: The galaxies data are commonly used to illustrate mixture modelling. From the R help:

```
galaxies                  package:MASS                  R Documentation

Velocities for 82 Galaxies

Description:

     A numeric vector of velocities in km/sec of 82 galaxies from 6
     well-separated conic sections of an 'unfilled' survey of the
     Corona Borealis region.  Multimodality in such surveys is evidence
     for voids and superclusters in the far universe.
```

```
(Y<-MASS::galaxies/10000);n<-length(Y)

+  [1] 0.9172 0.9350 0.9483 0.9558 0.9775 1.0227 1.0406 1.6084 1.6170 1.8419
+ [11] 1.8552 1.8600 1.8927 1.9052 1.9070 1.9330 1.9343 1.9349 1.9440 1.9473
+ [21] 1.9529 1.9541 1.9547 1.9663 1.9846 1.9856 1.9863 1.9914 1.9918 1.9973
+ [31] 1.9989 2.0166 2.0175 2.0179 2.0196 2.0215 2.0221 2.0415 2.0629 2.0795
+ [41] 2.0821 2.0846 2.0875 2.0986 2.1137 2.1492 2.1701 2.1814 2.1921 2.1960
+ [51] 2.2185 2.2209 2.2242 2.2249 2.2314 2.2374 2.2495 2.2746 2.2747 2.2888
+ [61] 2.2914 2.3206 2.3241 2.3263 2.3484 2.3538 2.3542 2.3666 2.3706 2.3711
+ [71] 2.4129 2.4285 2.4289 2.4366 2.4717 2.4990 2.5633 2.6690 2.6995 3.2065
+ [81] 3.2789 3.4279

par(mar=c(4,4,2,0))
low<-0.5;high<-4;del<-0.1
hist(Y,breaks=seq(low,high,by=del),col='gray',ylim=range(0,20),main='Galaxy data')
box()
```



We can analyze these data using the EM algorithm

```
mix.like<-function(muv,sigv,omv,yv){
        dvec<-rep(0,length(yv))
        for(i in 1:length(yv)){
                dvec[i]<-sum(omv*dnorm(yv[i],muv,sigv))
        }
        return(sum(log(dvec)))
}

em.analysis<-function(Yv,Kv,tolv=1e-10,iseed=10101){
    nv<-length(Yv)
    #Starting values
    omega.r<-omega.r1<-rep(1/Kv,Kv)
    if(iseed == 0){
        mu.vec.r<-mu.vec.r1<-quantile(Yv,prob=c(1:Kv)/(Kv+1))
    }else{
        set.seed(iseed)
        mu.vec.r<-mu.vec.r1<-runif(Kv,min(Yv),max(Yv))
    }
    sig.vec.r<-sig.vec.r1<-rep(sqrt(var(Yv)),Kv)
    varomega.r<-matrix(0,nrow=nv,ncol=Kv)

    Ym<-matrix(Yv,ncol=1)
    #Run the EM algorithm until convergence.
    del<-1
    while(del > tolv){
        for(i in 1:nv){
            for(k in 1:Kv){
                varomega.r[i,k]<-omega.r[k]*dnorm(Yv[i],mu.vec.r[k],sig.vec.r[k])
            }
            varomega.r[i,]<-varomega.r[i,]/sum(varomega.r[i,])
        }

        omega.r1<-apply(varomega.r,2,sum)
        omega.r1<-omega.r1/sum(omega.r1)
        for(k in 1:Kv){
            mu.vec.r1[k]<-sum(varomega.r[,k]*Yv)/sum(varomega.r[,k])
            sig.vec.r1[k]<-sqrt(sum(varomega.r[,k]*(Yv-mu.vec.r1[k])^2)/sum(varomega.r[,k]))
        }
        del<-sum((omega.r-omega.r1)^2)+sum((mu.vec.r-mu.vec.r1)^2)+
             sum((sig.vec.r-sig.vec.r1)^2)
        omega.r<-omega.r1
        mu.vec.r<-mu.vec.r1
        sig.vec.r<-sig.vec.r1
        like<-mix.like(mu.vec.r,sig.vec.r,omega.r,Yv)
    }
    return(list(mu=mu.vec.r,sig=sig.vec.r,omega=omega.r,Like=like))
}
```

The EM algorithm exploits the complete data likelihood obtained by including the latent component indictor variables $\mathbf{Z} = (Z_1, \ldots, Z_n)$, and recursively solves

$$\theta^{(r+1)} = \arg\max_{\theta} Q(\theta; \theta^{(r)}) = \arg\max_{\theta} \mathbb{E}_{\mathbf{Z}|\mathbf{Y}} \left[\log f(\mathbf{Y}, \mathbf{Z}; \theta) | \mathbf{y}; \theta^{(r)} \right]$$

for $r = 1, 2, \ldots,$ where

$$f(\mathbf{y}, \mathbf{z}; \theta) = \prod_{i=1}^{n} \prod_{k=1}^{K} \left\{ \omega_k f(y_i; \mu_k, \sigma_k^2) \right\}^{\mathbb{1}_{\{k\}(z_i)}}$$

is the complete data likelihood.

The EM algorithm can be sensitive to starting values due to the multi-modal nature of the likelihood, so we choose 20 random collections of starting means, and take the maximum value achieved across the different runs.

```
Like.mat<-matrix(0,nrow=8,ncol=20)
Like.vec<-j.vec<-rep(0,8)
for(k in 1:8){
    rmax<--Inf
    for(j in 1:20){
        f.em<-em.analysis(Y,tol=1e-10,Kv=k,iseed=j^3+j^2+j+1)
        Like.mat[k,j]<-f.em$Like
        if(f.em$Like > rmax){
            Like.vec[k]<-f.em$Like
            rmax<-f.em$Like
            j.vec[k]<-j
        }
    }
}
```

We can perform model selection using AIC or BIC

$$\text{AIC} : -2\log \mathcal{L}_n^{(K)}(\widehat{\theta}^{(K)}) + 2d_K$$

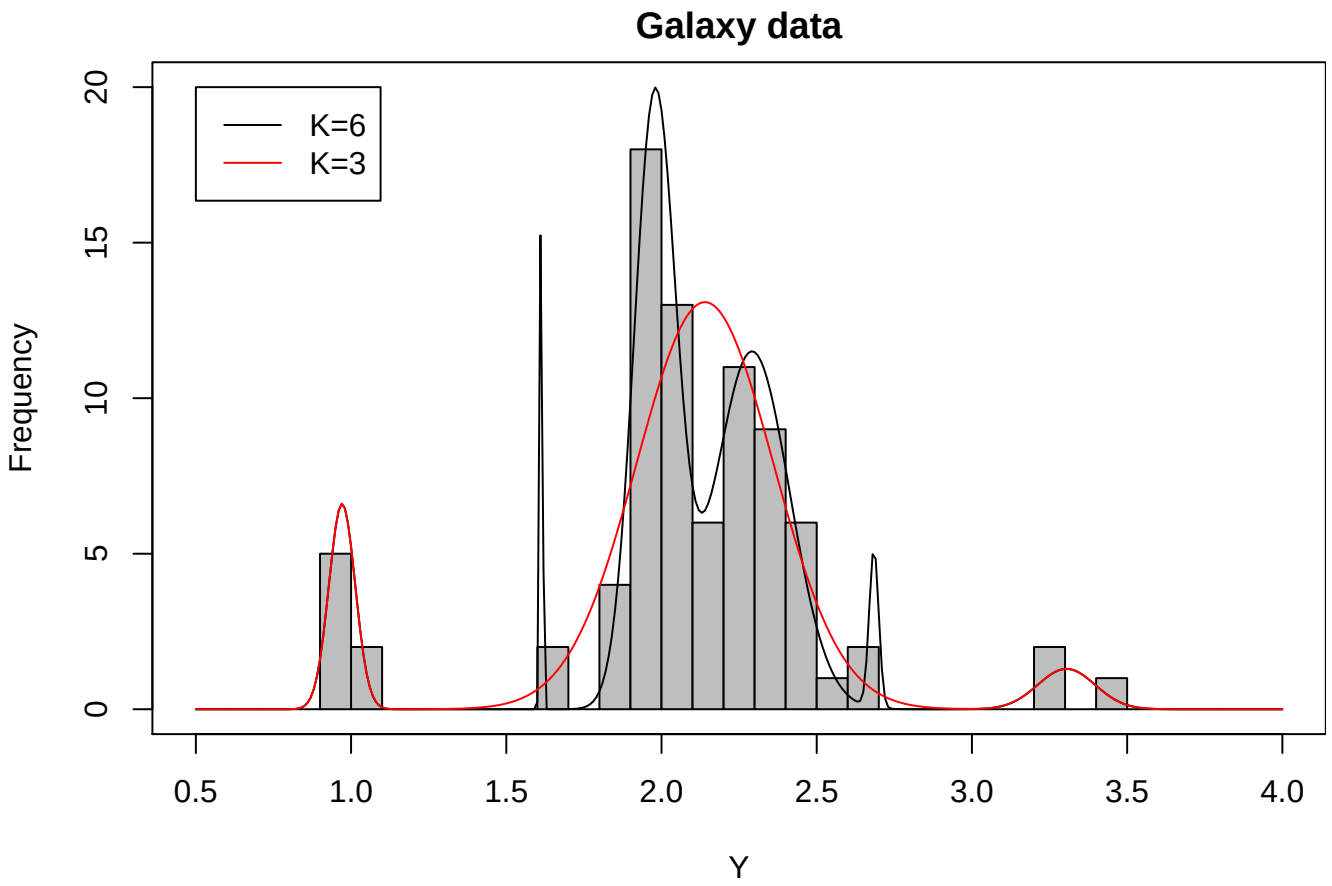
$$\text{BIC} : -2\log \mathcal{L}_n^{(K)}(\widehat{\theta}^{(K)}) + d_K \log n$$

where $d_K = 3K - 1$ – we select the model for which each quantity is smallest.

```
np<-3*c(1:8)-1
Res<-rbind(Like.vec,-2*Like.vec+np*2,-2*Like.vec+np*log(n))
colnames(Res)<-1:8
row.names(Res)<-c("Max logLik.","AIC","BIC")
round(Res,3)

+                   1       2       3      4      5      6      7      8
+ Max logLik. -51.526 -31.246 -14.367 -8.642 -1.259  1.945  4.714  7.822
+ AIC         107.052  72.492  44.735 39.284 30.518 30.111 30.571 30.355
+ BIC         111.865  84.526  63.988 65.757 64.212 71.025 78.706 85.710
```

By AIC, there is little difference between fits for $k = 5, 6, 7, 8$, whereas for BIC, $k = 3$ seems optimal.

```
mix.like.par<-function(muv,sigv,omv,yv,Kv){

        dvec<-rep(0,length(yv))
        for(k in 1:Kv){
                dvec<-dvec+omv[k]*dnorm(yv,muv[k],sigv[k])
        }
        return(sum(log(dvec)))
}
f.emA<-em.analysis(Y,tolv=1e-10,Kv=6,iseed=j.vec[6]^3+j.vec[6]^2+j.vec[6]+1)
f.emB<-em.analysis(Y,tolv=1e-10,Kv=3,iseed=j.vec[3]^3+j.vec[3]^2+j.vec[3]+1)
xvec<-seq(low,high,by=0.01)
yvec.A<-yvec.B<-xvec*0
for(k in 1:6){
        yvec.A<-yvec.A+f.emA$omega[k]*dnorm(xvec,f.emA$mu[k],f.emA$sig[k])
}
for(k in 1:3){
        yvec.B<-yvec.B+f.emB$omega[k]*dnorm(xvec,f.emB$mu[k],f.emB$sig[k])
}
par(mar=c(4,4,2,0))
low<-0.5;high<-4;del<-0.1
hist(Y,breaks=seq(low,high,by=del),col='gray',main='Galaxy data',ylim=range(0,20));box()
lines(xvec,yvec.A*n*del)
lines(xvec,yvec.B*n*del,col='red')
legend(0.5,20,c('K=6','K=3'),lty=1,col=c('black','red'))
```



For the Bayesian solution, we specify the following hierarchical prior

$$\pi_0(K)\pi_0(\sigma^2_{K1},\ldots,\sigma^2_{KK}|K)\pi_0(\mu^2_{K1},\ldots,\mu^2_{KK}|\sigma^2_{K1},\ldots,\sigma^2_{KK},K)\pi_0(\omega_{K1},\ldots,\omega^2_{KK}|K)$$

- Prior for K: we specify a $Poisson(\gamma)$ prior restricted to the positive integers, with pmf

$$\pi_0(K) = \frac{\gamma^K \exp\{-\gamma\}}{(1-\exp\{-\gamma\})K!} \qquad K = 1,2,\ldots$$

  for parameter $\gamma > 0$.

- Prior for σ^2_{Kk}: we specify independent priors

$$\pi_0(\sigma^2_{K1},\ldots,\sigma^2_{KK}|K) = \prod_{k=1}^{K} \pi_0(\sigma^2_{Kk})$$

  where $\pi_0(\sigma^2_{Kk}) \equiv InverseGamma(a/2, b/2)$ for hyperparameters $a, b > 0$.

- Prior for μ_{Kk} given σ^2_k: we specify conditionally independent priors

$$\pi_0(\mu^2_{K1},\ldots,\mu^2_{KK}|\sigma^2_{K1},\ldots,\sigma^2_{KK},K) = \prod_{k=1}^{K} \pi_0(\mu_{Kk}|\sigma^2_{Kk})$$

  where $\pi_0(\mu_{Kk}|\sigma^2_{Kk}) \equiv Normal(\eta, \sigma^2_{Kk}/\lambda)$ for hyperparameters η and $\lambda > 0$.

- Prior for $\omega_{(K)} = (\omega_1^{(K)},\ldots,\omega_K^{(K)})$: we specify a symmetric Dirichlet prior

$$\pi_0(\omega_{K1},\ldots,\omega_{KK}|K) = \frac{\Gamma(K\alpha)}{\{\Gamma(\alpha)\}^K} \prod_{k=1}^{K} \omega_{Kk}^{\alpha-1}$$

  on the K-dimensional unit simplex.

For a reversible jump MCMC solution, we consider Birth and Death move pairs for the transdimensional moves:

- **Birth Move:** We consider an update that changes $K \longrightarrow K+1$, requiring the need to generate a new component. We simulate the new mean and variance parameters from the prior distribution:

$$\sigma^2_{(K+1)(K+1)} \sim InverseGamma(a/2, b/2)$$

$$\mu_{(K+1)(K+1)} \sim Normal(\eta, \sigma^2_{(K+1)(K+1)}/\lambda)$$

For the new component weight we generate $\psi \sim Beta(1, K)$, and then set the new component weight vector $\omega_{(K+1)}$ as

$$(\omega_{(K+1)1}, \ldots, \omega_{(K+1)(K+1)}) = ((1-\psi)\omega_{K1}, \ldots, (1-\psi)\omega_{KK}, \psi).$$

The Jacobian of the transform is

$$\frac{1}{(1-\omega_{(K+1)(K+1)})^K}$$

and therefore the acceptance probability for the Birth move is the minimum of 1 and

$$\frac{\pi_n(K+1, \mu_{(K+1)}, \sigma^2_{(K+1)}, \omega_{(K+1)})}{\pi_n(K, \mu_K, \sigma^2_K, \omega_K)\, p(\mu_{(K+1)(K+1)}, \sigma^2_{(K+1)(K+1)})\, p(\psi)} \times \frac{1}{(1-\omega_{(K+1)(K+1)})^K}$$

- **Death Move:** For the (reverse) Death move we consider the dimension change $K+1 \longrightarrow K$ by picking a component at random from the $K+1$ existing components to eliminate, and then reversing the transformation from the Birth move to set

$$(\omega_{K1}, \ldots, \omega_{KK}, \psi) = \frac{1}{(1-\psi)}(\omega_{(K+1)1}, \ldots, \omega_{(K+1)K})$$

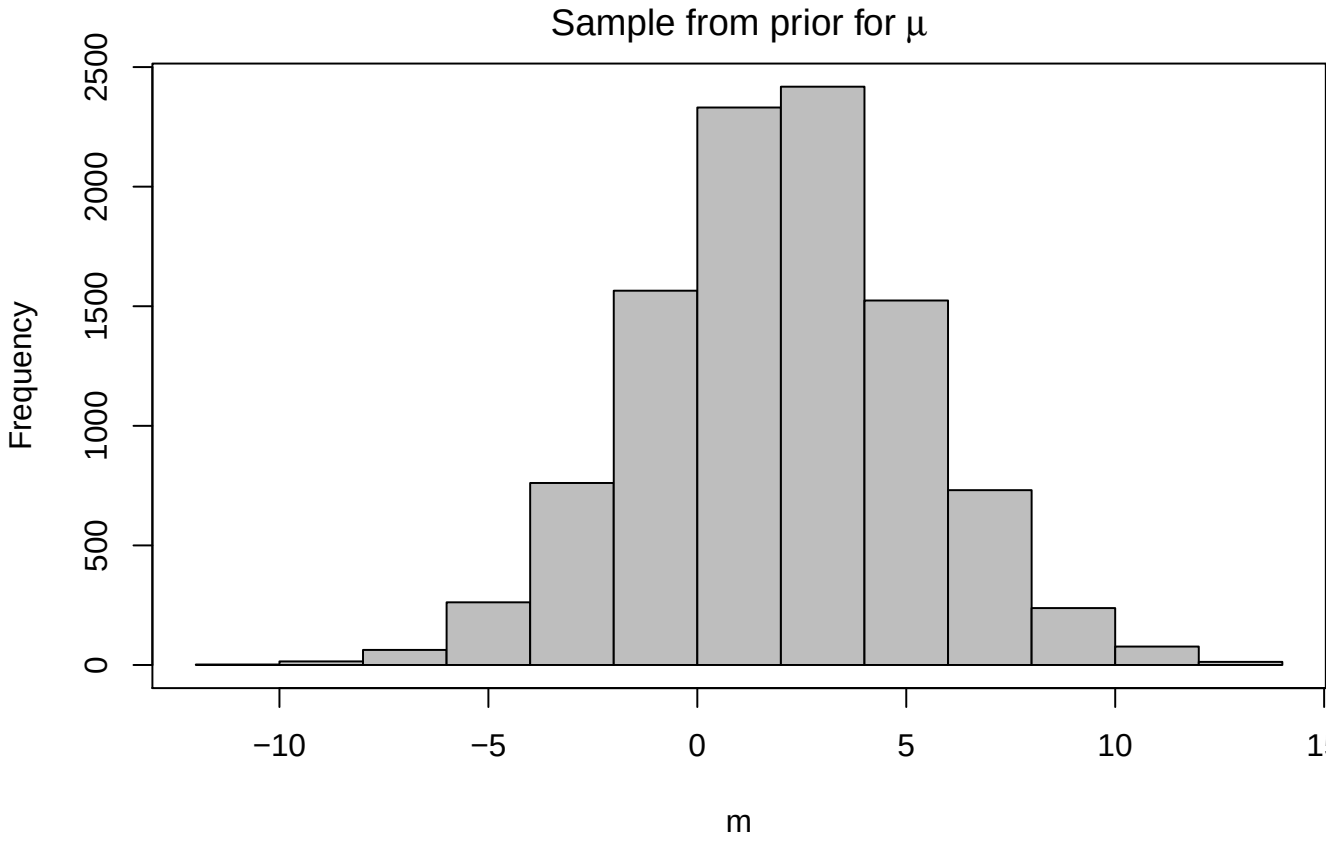
for which the Jacobian is $(1-\psi)^K$, and the acceptance probability is the minimum of 1 and

$$\frac{\pi_n(K, \mu_K, \sigma^2_K, \omega_K)\, p(\mu_{(K+1)(K+1)}, \sigma^2_{(K+1)(K+1)})\, p(\psi)}{\pi_n(K+1, \mu_{(K+1)}, \sigma^2_{(K+1)}, \omega_{(K+1)})} \times (1-\psi)^K$$

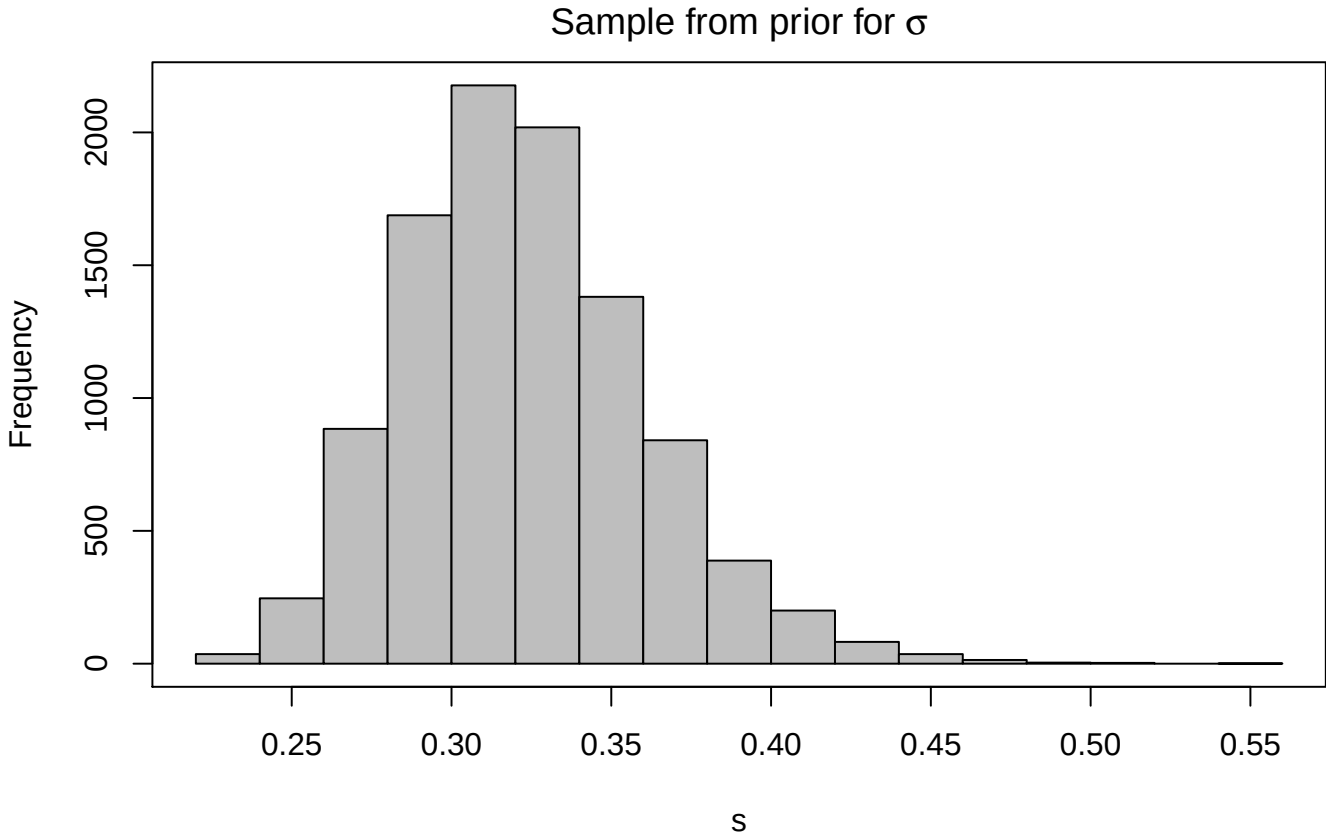
```
mix.like.par<-function(muv,sigv,omv,yv,Kv){
        dvec<-rep(0,length(yv))
        for(k in 1:Kv){
                dvec<-dvec+omv[k]*dnorm(yv,muv[k],sigv[k])
        }
        return(sum(log(dvec)))
}

dinvgamma<-function(xv,a,b,log=F){
        lv<-a*log(b)-lgamma(a)-(a+1)*log(xv)-b/xv
        if(log){
                return(lv)
        }else{
                return(exp(lv))
        }
}

dpois0<-function(xv,lam,log=F){ #Poisson pmf with zero removed.
        fv<-dpois(xv,lam,log=log)-log(1-exp(-lam))
        if(log){return(fv)
        }else{return(exp(fv))
        }
}
```

Choosing the hyperparameters for this conjugate prior can be achieved using simulation methods.

```r
#Prior elicitation
prior.al<-1
prior.eta<-2
prior.lam<-0.01
prior.a<-40
prior.b<-4
prior.K.gam<-2

s1<-1
a1<-1

s<-(1/sqrt(rgamma(10000,prior.a/2,prior.b/2)))
m<-rnorm(10000)*s/sqrt(prior.lam)+prior.eta
par(mar=c(4,4,2,0))
hist(m,col='gray',main=expression(paste('Sample from prior for ',mu)));box()
```




```r
hist(s,col='gray',main=expression(paste('Sample from prior for ',sigma)));box()
```



In order to facilitate ease of posterior sampling for the component parameters, we again introduce the latent component indicators $Z_1, \ldots, Z_n$, and sample their values from their full conditional posterior distributions as part of the Gibbs sampler. Conditional on the component parameters, we have that

$$\Pr[Z_i = k|\mathbf{y}, \boldsymbol{\theta}, K] = \frac{\omega_{Kk} f_k(y_i; \mu_{Kk}, \sigma^2_{Kk})}{\sum_{j=1}^{K} \omega_{Kj} f_j(y_i; \mu_{Kj}, \sigma^2_{Kj})} \qquad k = 1, \ldots, K$$

independently for $i = 1, \ldots, n$. Conditional on the sampled Z values we have a conjugate model: specifically the posterior distribution of σ^2_{Kk} is $InverseGamma(a_{nk}/2, b_{nk}/2)$ distribution with parameters

$$a_{nk} = a + n_{Kk} \qquad b_{nk} = b + \frac{n_{Kk}\lambda}{n_{Kk} + \lambda}(\overline{y}_{nKk} - \eta)^2 + \sum_{i:z_i=k} (y_i - \overline{y}_{nKk})^2$$

and the conditional posterior for μ_{Kk} is $Normal(\eta_{nk}, \sigma^2_{Kk}/\lambda_{nk})$ where

$$\eta_{nk} = \frac{\lambda\eta + \sum_{i:z_i=k} y_i}{\lambda + n_{Kk}} \qquad \lambda_{nk} = \lambda + n_{Kk}$$

where

$$n_{Kk} = \sum_{i=1}^{n} \mathbb{1}_{\{k\}}(z_i) \qquad \overline{y}_{nKk} = \frac{\sum_{i=1}^{n} \mathbb{1}_{\{k\}}(z_i) y_i}{\sum_{i=1}^{n} \mathbb{1}_{\{k\}}(z_i)}$$

are the component specific sample size and sample mean at the current iteration. Note that these formulae hold if a component is 'empty', that is, if no data are allocated to a given cluster, as in such cases the conditional posterior and conditional prior are identical. For ω_K we have a simple conjugate analysis given the latent indicators

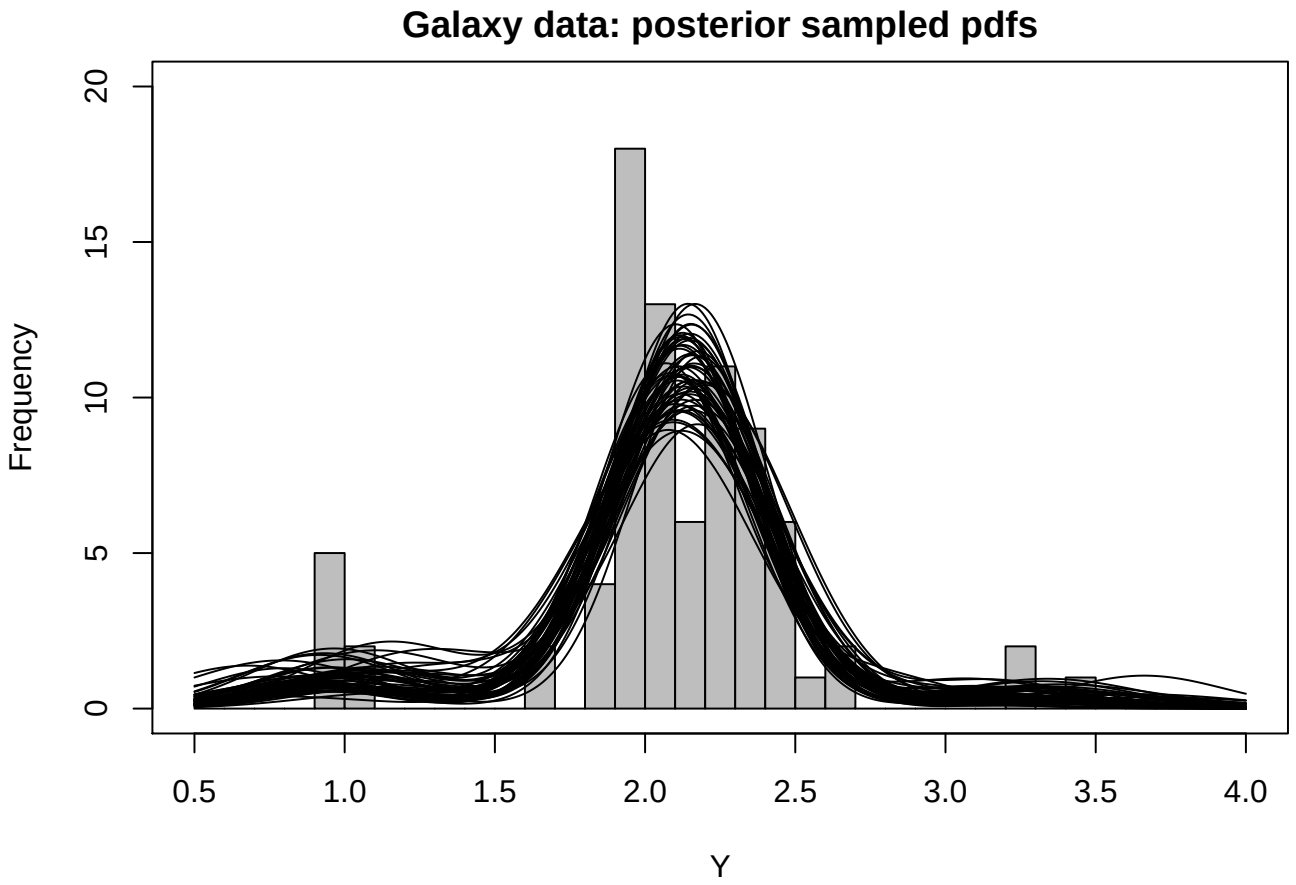
$$\omega_K|- \sim Dirichlet(K; \alpha + n_{K1}, \ldots, \alpha + n_{KK})$$

Note that the Birth and Death steps can be implemented on the collapsed form, by summing out over the possible values of the latent component indicators.

```
set.seed(234)

#Starting values
old.K<-4
old.omega<-old.omega1<-rep(1/old.K,old.K)
old.mu<-old.mu1<-as.numeric(quantile(Y,prob=c(1:old.K)/(old.K+1)))
old.sig<-old.sig1<-rep(sqrt(var(Y)),old.K)
old.z<-rep(0,n)

#Run the MCMC algorithm
nburn<-5000
nsamp<-10000
nthin<-5
nits<-nburn+nsamp*nthin

#Galaxy plot
low<-0.5
high<-4
del<-0.1

par(mar=c(4,4,2,1))
hist(Y,breaks=seq(low,high,by=del),ylim=range(0,20),col='gray',
        main='Galaxy data: posterior sampled pdfs');box()

mu.samp<-matrix(0,nrow=nsamp,ncol=100)
sig.samp<-matrix(0,nrow=nsamp,ncol=100)
omega.samp<-matrix(0,nrow=nsamp,ncol=100)
K.samp<-rep(0,nsamp)
ico<-0
for(iter in 1:nits){
        pvec<-rep(0,old.K)
        for(i in 1:n){
                for(k in 1:old.K){
                        pvec[k]<-old.omega[k]*dnorm(Y[i],old.mu[k],old.sig[k])
                }
                old.z[i]<-sample(1:old.K,prob=pvec,size=1)
        }

        for(k in 1:old.K){
                old.omega1[k]<-sum(old.z==k)
        }
        old.omega1<-rgamma(old.K,old.omega1+prior.al,1)
        old.omega<-old.omega1/sum(old.omega1)

        for(k in 1:old.K){
                yk<-Y[old.z==k]
                nk<-length(yk)
                if(nk == 0){
                        old.sig[k]<-1/sqrt(rgamma(1,prior.a/2,prior.b/2))
                        old.mu[k]<-rnorm(1,prior.eta,old.sig[k]/sqrt(prior.lam))
                }else{
                        ssq<-sum((yk-mean(yk))^2)
                        post.a<-prior.a+nk
                        post.b<-prior.b+ssq+(nk*prior.lam)*(mean(yk)-prior.eta)^2/(nk+prior.lam)
                        old.sig[k]<-1/sqrt(rgamma(1,post.a/2,post.b/2))
                        post.sig<-old.sig[k]/sqrt(nk+prior.lam)
                        post.mu<-(sum(yk)+prior.lam*prior.eta)/(nk+prior.lam)
                        old.mu[k]<-rnorm(1,post.mu,post.sig)
                }
        }
```

```
        old.like<-mix.like.par(old.mu,old.sig,old.omega,Y,old.K)
        old.prior<-sum(dinvgamma(old.sig^2,prior.a/2,prior.b/2,log=T))+
                        sum(dnorm(old.mu,prior.eta,old.sig/sqrt(prior.lam),log=T))+
                        lgamma(old.K*prior.al)-old.K*lgamma(prior.al)+
                    (prior.al-1)*sum(log(old.omega))+dpois0(old.K,prior.K.gam,log=T)

        #Transdimensional moves: Birth and Death
        if(runif(1) < 0.5){
                #Death
                new.K<-old.K-1
                if(new.K == 0){ #Cannot remove last component, prior prob is zero
            mh<--Inf
                }else{
                        k<-sample(1:old.K,size=1)
                        new.mu<-old.mu[-k]
                        new.sig<-old.sig[-k]
                        new.omega<-old.omega[-k]/sum(old.omega[-k])
                        new.like<-mix.like.par(new.mu,new.sig,new.omega,Y,new.K)
                        new.prior<-sum(dinvgamma(new.sig^2,prior.a/2,prior.b/2,log=T))+
                sum(dnorm(new.mu,prior.eta,new.sig/sqrt(prior.lam),log=T))+
                                lgamma(new.K*prior.al)-new.K*lgamma(prior.al)+
                (prior.al-1)*sum(log(new.omega))+dpois0(new.K,prior.K.gam,log=T)
                        new.q<-dnorm(old.mu[k],prior.eta,old.sig[k]/sqrt(prior.lam),log=T)+
                dinvgamma(old.sig[k]^2,prior.a/2,prior.b/2,log=T)+
                dbeta(old.omega[k],1,new.K,log=T)
            logJac<-new.K*log(1-old.omega[k])
                        mh<-(new.like+new.prior)-(old.like+old.prior)+new.q+logJac
                }
                if(log(runif(1)) < mh){
                        old.K<-new.K
                        old.mu<-new.mu
                        old.sig<-new.sig
                        old.omega<-new.omega
                        old.like<-new.like
                        old.prior<-new.prior
                }
        }else{
                #Birth
                new.K<-old.K+1
                new.s<-1/sqrt(rgamma(1,prior.a/2,prior.b/2))
                new.m<-rnorm(1,prior.eta,new.s/sqrt(prior.lam))
                new.mu<-c(old.mu,new.m)
                new.sig<-c(old.sig,new.s)
                psi<-rbeta(1,1,old.K)
                new.omega<-c(old.omega*(1-psi),psi)
                new.like<-mix.like.par(new.mu,new.sig,new.omega,Y,new.K)
                new.prior<-sum(dinvgamma(new.sig^2,prior.a/2,prior.b/2,log=T))+
            sum(dnorm(new.mu,prior.eta,new.sig/sqrt(prior.lam),log=T))+
            lgamma(new.K*prior.al)-new.K*lgamma(prior.al)+
            (prior.al-1)*sum(log(new.omega))+dpois0(new.K,prior.K.gam,log=T)
                new.q<-dnorm(new.m,prior.eta,new.s/sqrt(prior.lam),log=T)+
            dinvgamma(new.s^2,prior.a/2,prior.b/2,log=T)+
            dbeta(psi,1,old.K)
                logJac<--old.K*log(1-psi)
                mh<-(new.like+new.prior)-(old.like+old.prior)-new.q+logJac
                if(log(runif(1)) < mh){
                        old.K<-new.K
                        old.mu<-new.mu
                        old.sig<-new.sig
                        old.omega<-new.omega
                        old.like<-new.like
```

```
                        old.prior<-new.prior
                }
        }

        if(iter %% 1000 == 0){
                Ly<-mix.like.par(old.mu,old.sig,old.omega,Y,old.K)
                old.Py<-sum(dinvgamma(old.sig^2,prior.a/2,prior.b/2,log=T))+
                        sum(dnorm(old.mu,prior.eta,old.sig/sqrt(prior.lam),log=T))+
                        lgamma(old.K*prior.al)-old.K*lgamma(prior.al)+
                    (prior.al-1)*sum(log(old.omega))+dpois0(old.K,prior.K.gam,log=T)

                xvec<-seq(low,high,by=0.01)
                yvec<-xvec*0
                for(k in 1:old.K){
                        yvec<-yvec+old.omega[k]*dnorm(xvec,old.mu[k],old.sig[k])
                }
                lines(xvec,yvec*n*del)
        }

        if(iter > nburn & iter %% nthin == 0){
                ico<-ico+1
                mu.samp[ico,1:old.K]<-old.mu
                sig.samp[ico,1:old.K]<-old.sig
                omega.samp[ico,1:old.K]<-old.omega
                K.samp[ico]<-old.K
        }
}
```



The approximate posterior distribution (under this prior specification) is computed from the sampled values of K.

```
K.post<-table(K.samp)/nsamp
K.post

+ K.samp
+      2      3      4      5      6
+ 0.0003 0.9168 0.0795 0.0033 0.0001
```

Therefore the posterior probability on $k = 3$ is 0.916800.