# Chr1

## Basic definitions

1. B
2. 2.075
3. 2.6975

## Definition of Mixture Models

1. C
2. 1.1
3. True
4. FALSE

## HW：Simulating from a Mixture Model

# Generate n observations from a mixture of two Gaussian

# distributions

n = 50 # Size of the sample to be generated

w = c(0.6, 0.4) # Weights

mu = c(0, 5) # Means

sigma = c(1, 2) # Standard deviations

cc = sample(1:2, n, replace=T, prob=w)

x = rnorm(n, mu[cc], sigma[cc])

# Plot f(x) along with the observations

# just sampled

xx = seq(-5, 12, length=200)

yy = w[1]\*dnorm(xx, mu[1], sigma[1]) +

w[2]\*dnorm(xx, mu[2], sigma[2])

par(mar=c(4,4,1,1)+0.1)

plot(xx, yy, type="l", ylab="Density", xlab="x", las=1, lwd=2)

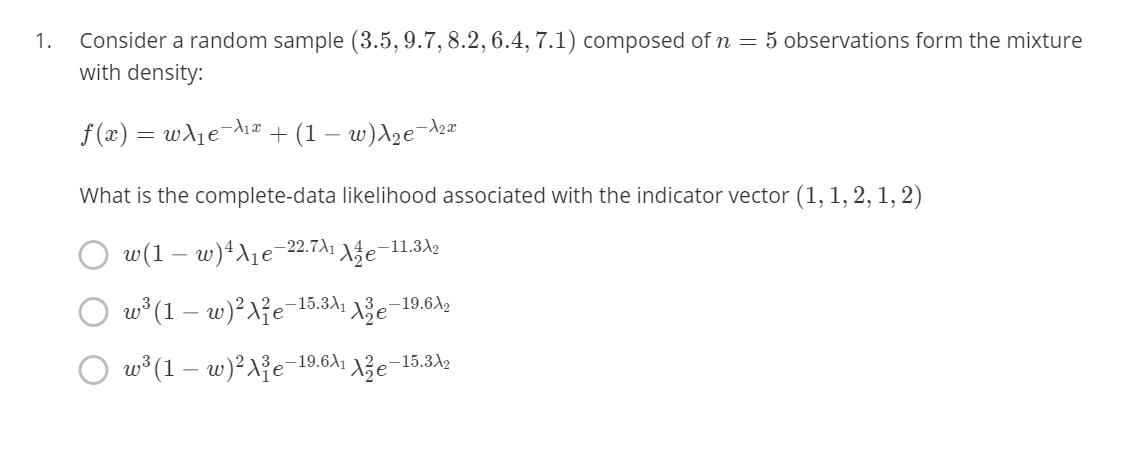
points(x, y=rep(0,n), pch=1)

Modify the code above to sample 200 random numbers from a mixture of 3 Poisson distributions with means 1, 2 and 6 and weights 0.7, 0.2 and 0.1, respectively, and generate a barplot with the empirical frequencies of all the integers included in the sample.

Review criteria

The response should follow the same template as the sample code provided above. Peer reviewers will be asked to check whether the different pieces of code have been adequately modified to reflect the fact that (1) you are working now with 3 components rather than 2, (2) that the components of the mixture are Poisson distributions rather than Gaussians, (3) that the empirical frequencies associated with the sample generated are correctly computed, (4) that a barplot is used to represent the empirical frequencies.

## The likelihood function

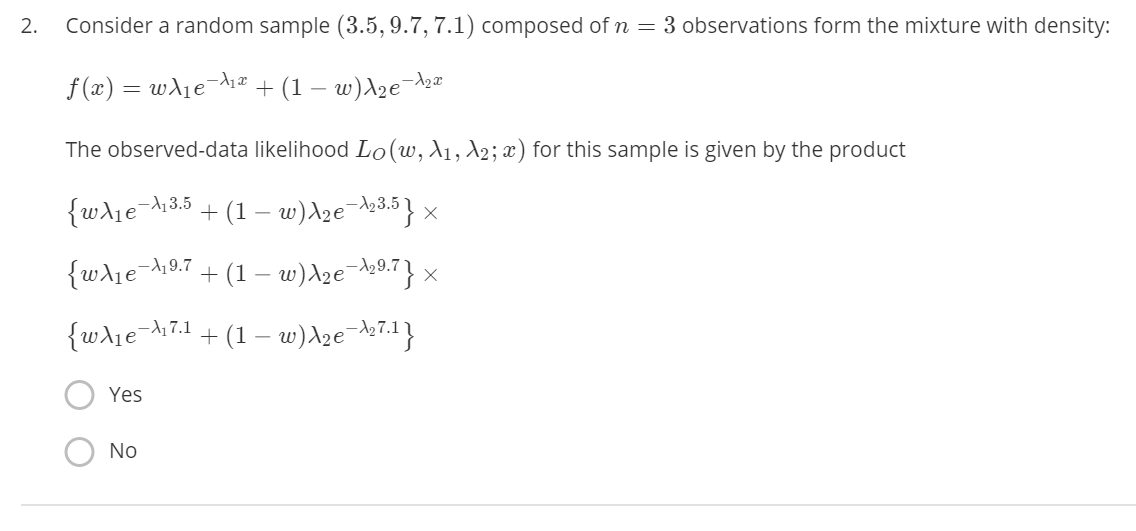


C

Recall the definition of the complete-data likelihood: it is (proportional to) the joint distribution of the indicator variables and the observation.

Since the indicator vector is (1,1,2,1,2)(1,1,2,1,2), there are 3 observations in group 1 and 2 observations. So, the first term must be w^3(1-w)^2*w*3(1−*w*)2.

Now, the distribution of the data given the indicator involves the product over two components: For the first component, we have 3 Poisson distributions (associated with the first, second and fourth observations, which add up to 19.6) , and for the second component we have 2 Poisson distributions (associated with the third and fifth observations, which add up to 15.3).



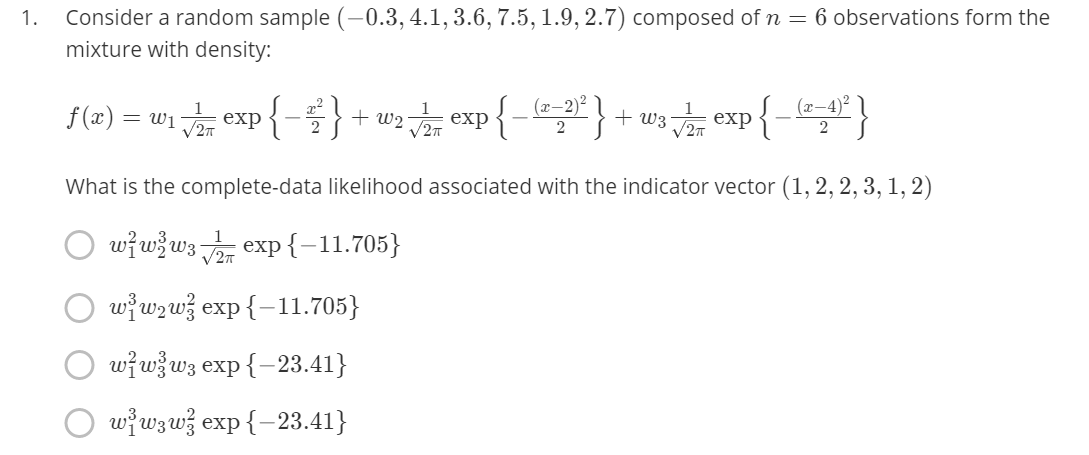
yes

The complete-data likelihood is proportional to the joint distribution of the data. Because observations are independent (this is a random sample), that corresponds to f(3.5) \times f(9.7) \times f(7.1)*f*(3.5)×*f*(9.7)×*f*(7.1).

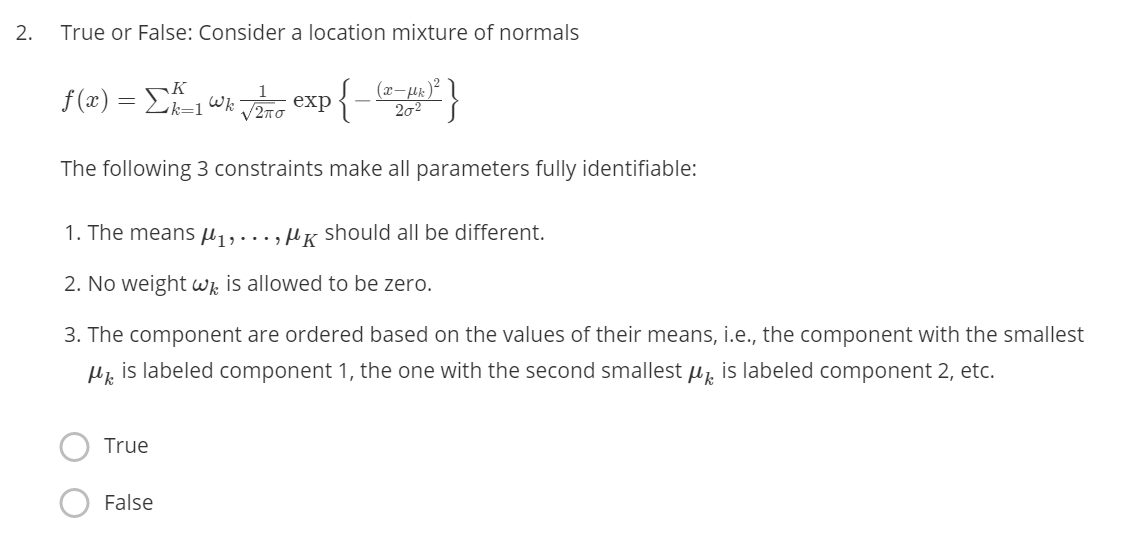
## Identifiability

1. no
2. no

## Likelihood function for mixture models



A



## HW:Likelihood function for mixture models

Consider the code provided in the lesson "Sample code for simulating from a Mixture Model":

# Generate n observations from a mixture of two Gaussian

# distributions

n = 50 # Size of the sample to be generated

w = c(0.6, 0.4) # Weights

mu = c(0, 5) # Means

sigma = c(1, 2) # Standard deviations

cc = sample(1:2, n, replace=T, prob=w)

x = rnorm(n, mu[cc], sigma[cc])

Modify the code above to sample 100 random numbers from a mixture of 4 exponential distributions with means 1, 4, 7 and 10 and weights 0.3, 0.25, 0.25 and 0.2, respectively. Use these sample to approximate the mean and variance of the mixture.

**Review criteria**

The response should follow the same template as the sample code provided above. Peer reviewers will be asked to check whether the different pieces of code have been adequately modified to reflect the fact that (1) you generate 100 random variates rather than 50, (2) you are working now with 4 components rather than 2, (3) the components of the mixture are exponential distributions rather than Gaussians, (4) the mean and variance are approximated correctly.

# Chr2

## Mixtures of log-Gaussians

If your data had support on the positive real numbers rather than the whole real line, how could you use the EM algorithm you just learned to instead fit a mixture of log-Gaussian distributions? Would you need to recode your algorithm

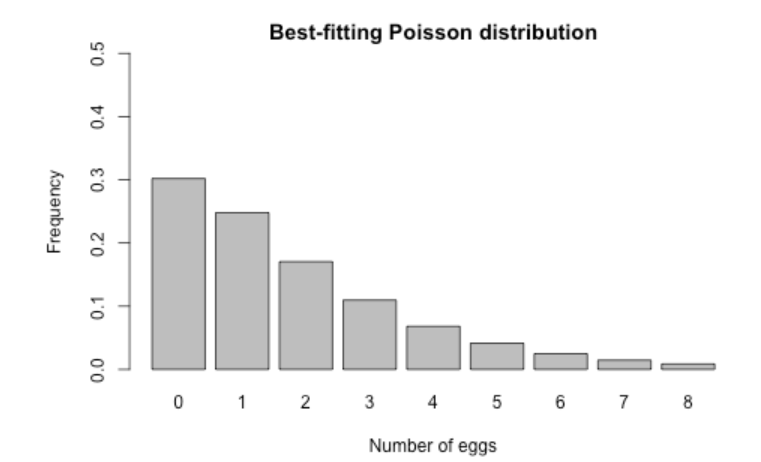
My answer：

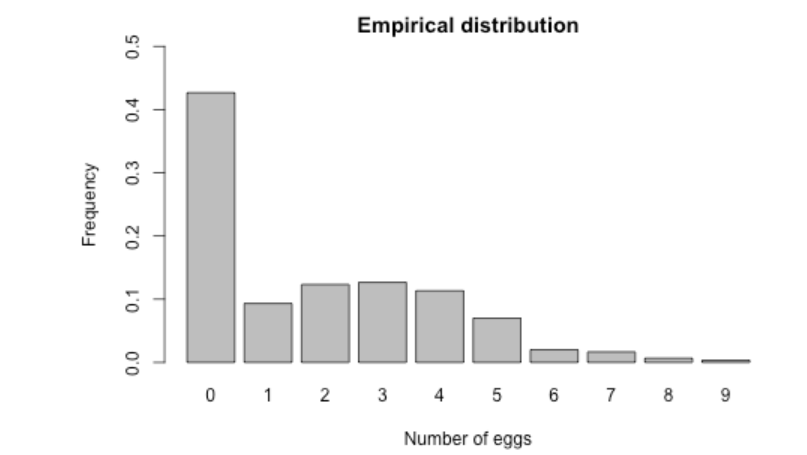
The key problem here is that the range of data values is limited, and thus GMM cannot be direcetly used for fitting. A commonly used strategy in statistics is transformation, for example, taking a log scale transformation to the data, releasing the value range to the entire real number space.

## HW： zero-inflated mixtures

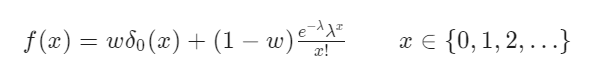
A biologist is interest in characterizing the number of eggs laid by a particular bird species. To do this, they sample n=300 nests on a site in Southern California. The observations are contained in the attached file nestsize.csv:

The following graph compares the empirical distribution of the data against a Poison distribution whose parameter has been set to its maximum likelihood estimator:





As you can see, the Poisson distribution underestimates the number of empty nests in the data, and overestimates the number of nests with either 1 or 2 eggs. To address this, you are asked to modify the implementation of the EM algorithm contained in the Reading "Sample code for EM example 1" so that you can fit a mixture between a point mass at zero and a Poisson distribution (we call this a "zero-inflated Poisson" distribution):



where *δ*0​(*x*) represents the degenerate distribution placing all of its mass at zero. You then should run your algorithm with the data contained in nestsize.csv and report the values of the estimates that you obtained, rounded to two decimal places.

Review **criteria**

The code you generate should follow the same structure as "Sample code for EM example 1". Peer reviewers will be asked to check whether the different pieces of code have been adequately modified to reflect the fact that (1) you provided a reasonable initial point for you algorithm, (2) the observation-specific weights *vi*,*k*​ are computed correctly (E step), (3) the formulas for the maximum of the *Q* functions are correct (M step), (4) the converge check is correct, and (5) the numerical values that you obtain are correct. To simplify the peer-review process, assume that component 1 corresponds to the point mass at zero, while component 2 corresponds to the Poisson distribution.

There are two things that make this problem more challenging than the ones we have used for illustrations so far: (1) the two components in the mixture belong to different families, and (2) each component has a very different support. Please keep these two circumstances in mind when working on your answer.

### Ref Answer

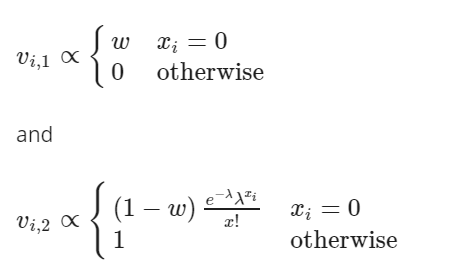
In this problem there are only two unknown parameters: the weight of the mixture (w), and the mean of the Poisson component (*λ*). As long as the parameters are in the right range (i.e., 0 < *w*<1 and *λ*>0), you should give credit for this prompt.

An example of a default choice that would likely work well in most cases would be something like:

w = 0.5

lambda = mean(x)

Note that two cases need to be considered. If the observation is a zero, then in can come from either the point mass or the Poisson distribution. On the other hand, if the observation is different from zero, it can only come from the Poisson component. Hence:



Hence, the code for the E step could look something like

## E step

v = array(0, dim=c(n,2))

for(i in 1:n){

if(x[i]==0){

v[i,1] = w

v[i,2] = (1-w)\*dpois(x[i], lambda)

v[i,] = v[i,]/sum(v[i,])

}else{

v[i,1] = 0

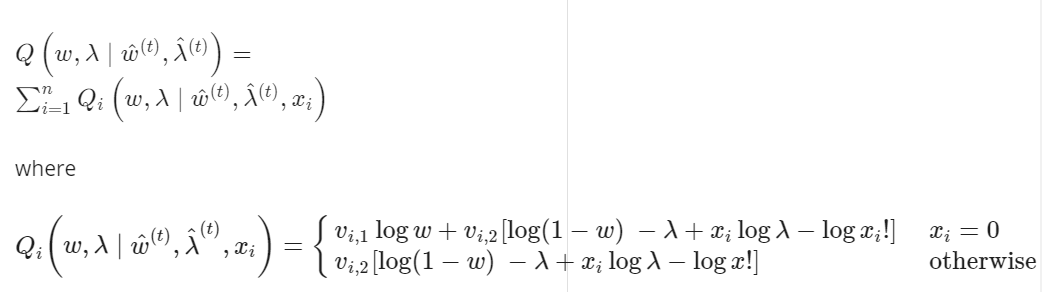
v[i,2] = 1

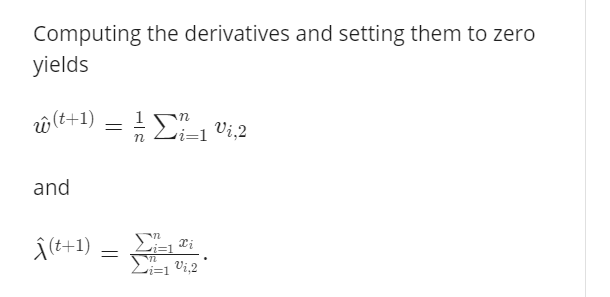
}

}

Again, when computing the *Q* you need to consider two cases.

In particular, recall that the *Q* function can be written as a sum over the observations:





Hence, the code for this section might look something like:

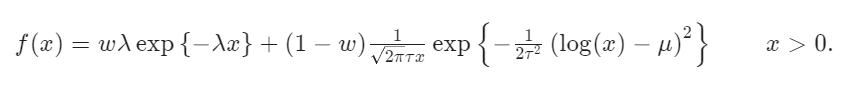
w = mean(v[,1])

lambda = sum(x)/sum(v[,2])

## HW：expo-logNormal mixture

Data on the lifetime (in years) of fuses produced by the ACME Corporation is available in the file fuses.csv:

In order to characterize the distribution of the lifetimes, it seems reasonable to fit to the data a two-component mixture of the form



The first component, which corresponds to an exponential distribution with rate *λ*, **is used to model low-quality components with a very short lifetime.** The second component, which corresponds to a [log-Gaussian distribution](https://en.wikipedia.org/wiki/Log-normal_distribution), **is used to model normal, properly-functioning components.**

You are asked to modify the implementation of the EM algorithm contained in the Reading "Sample code for EM example 1" so that you can fit this two-component mixture distributions instead. You then should run your algorithm with the data contained in *fuses.csv* and report the values of the estimates for *w*, *λ*, *μ* and *τ* that you obtained, rounded to two decimal places.

The code you generate should follow the same structure as "Sample code for EM example 1". Peer reviewers will be asked to check whether the different pieces of code have been adequately modified to reflect the fact that (1) you provided a reasonable initial point for you algorithm, (2) the observation-specific weights *vi*,*k*​ are computed correctly (E step), (3) the formulas for the maximum of the *Q* functions are correct (M step), (4) the converge check is correct, and (5) the numerical values that you obtain are correct. To simplify the peer-review process, assume that component 1 corresponds to the point mass at zero, while component 2 corresponds to the Poisson distribution.

### Ref Answer

The starting values of four parameters, w, *λ*, *μ* and *τ*, need to be specified, and the context of the problem provides some useful clues.

Because the lognormal component corresponds to the "normal" components, and we expect the majority of the observations to be in this class, it makes sense to bias the weights so that  *w*≤1/2. For example, we could use w = 0.1*w*=0.1 (but other values that satisfy  *w*≤1/2 would be reasonable too).

**For the same reason, a reasonable starting values for *μ* and *τ* correspond to their maximum likelihood estimators under the simpler log-Gaussian model. Since a random variable follows a log-Gaussian distribution if and only if its logarithm follows a Gaussian distribution**, we can use

*μ*=*mean*(*log*(*x*)) and

*τ*=*sd*(*log*(*x*))

as our starting values.

Finally, because the defective components should have shorter lifespan than normal components, it makes sense to take 1/*λ* (which is the mean of the first component) to be a small fraction of the overall mean of the data (we use 5% of the overall mean, but other similar values would all be reasonable).

In summary, you should expect an initialization such as this one:

w = 0.1

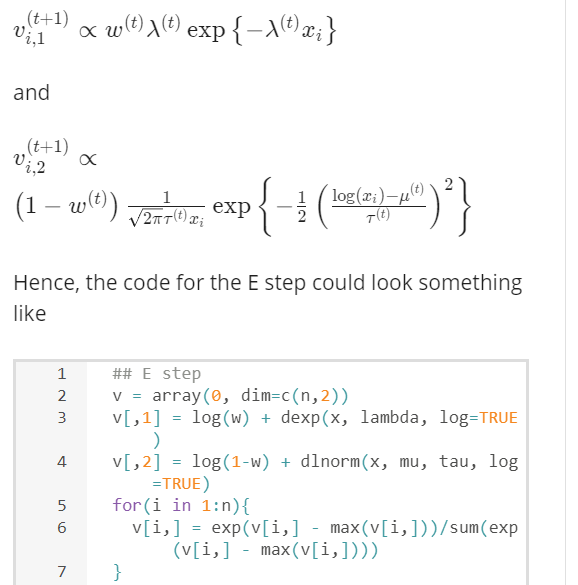
mu = mean(log(x))

tau = sd(log(x))

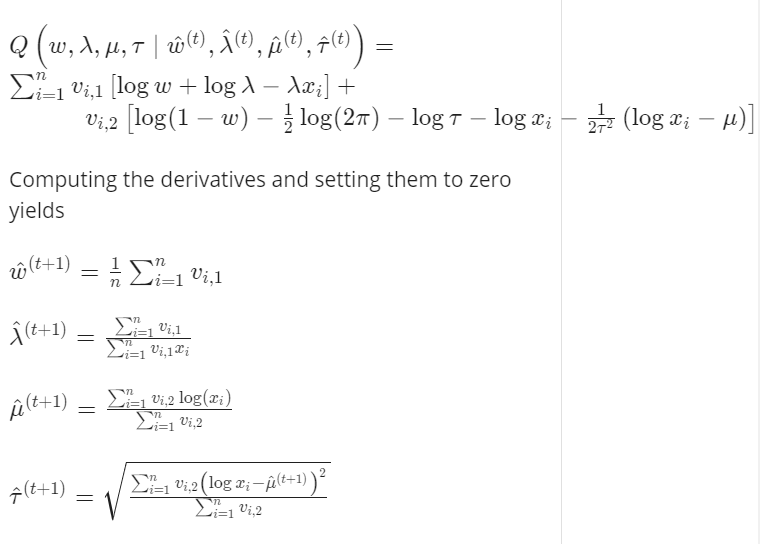
lambda = 20/mean(x)

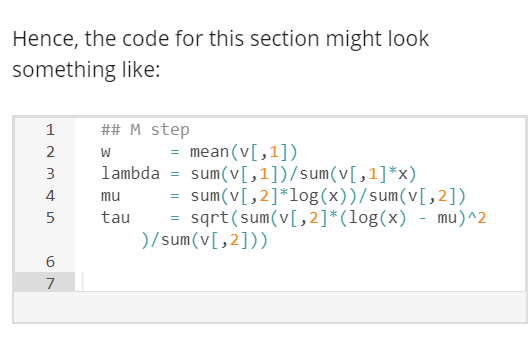
**Estep:**

In this case it is easy to see that

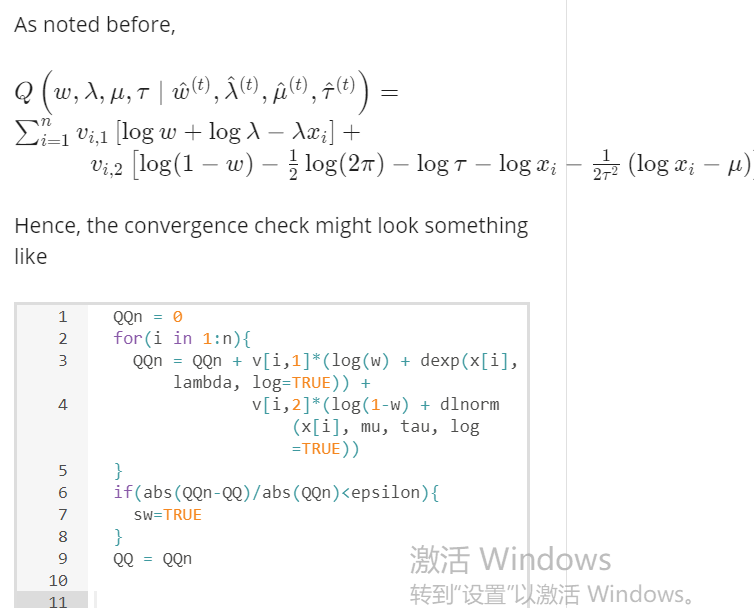


**Mstep:**





**Convergence**



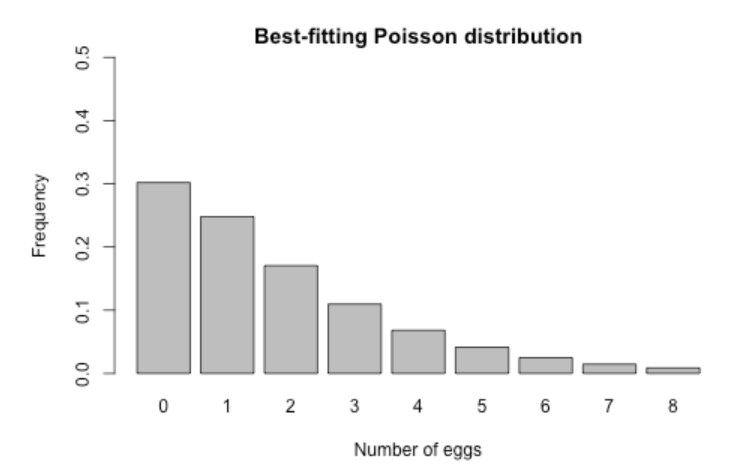
# Chr3

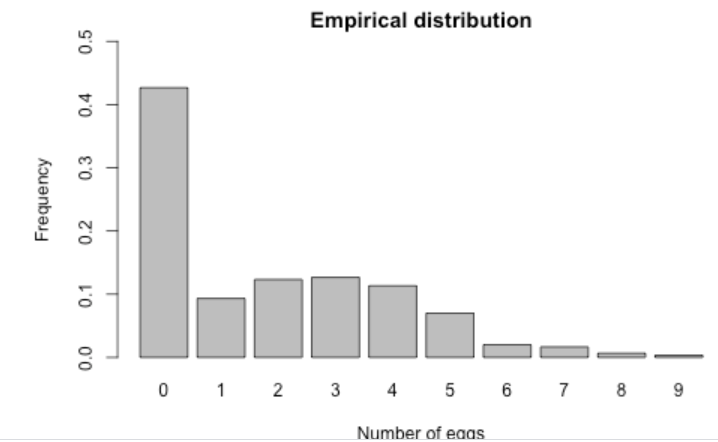
## HW：The MCMC algorithm for zero-inflated mixtures

This assignment is similar to one you already completed in Lesson 3. However, in this case you are asked to use an MCMC algorithm to perform Bayesian inference on the model parameters, instead of an EM algorithm to find maximum likelihood estimators.

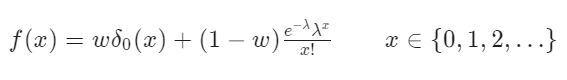
A biologist is interest in characterizing the number of eggs laid by a particular bird species. To do this, they sample n=300 nests on a site in Southern California. The observations are contained in the attached file *nestsize.csv*:

The following graph compares the empirical distribution of the data against a Poison distribution whose parameter has been set to its maximum likelihood estimator:





**As you can see, the Poisson distribution underestimates the number of empty nests in the data, and overestimates the number of nests with either 1 or 2 eggs.** To address this, you are asked to modify the implementation of the MCMC algorithm contained in the Reading "Sample code for MCMC example 1" so that you can fit a mixture between a point mass at zero and a Poisson distribution (we call this a "zero-inflated Poisson" distribution):



where *δ*0​(*x*) represents the degenerate distribution placing all of its mass at zero. You then should run your algorithm for 5,000 iterations (after a burn-in period of 1,000 iterations) with the data contained in *nestsize.csv* and report your estimates of the **posterior means**, **rounded to two decimal places.**

**In carrying out this assignment assume the following priors for your unknown parameters:  *w*∼*Uni*[0,1] and *λ*∼*Exp*(1).**

**Review criteria更少**

The code you generate should follow the same structure as "Sample code for MCMC example 1". In particular, focus on a Gibss sampler that alternates between the **full conditionals for *w*, *λ* and the latent component indicators  *c*1​,…,*cn*​.** Peer reviewers will be asked to check whether the different pieces of code have been adequately modified to reflect the fact that (1) parameters have been initialized in a reasonable way, (2) each of the two full conditional distributions associated with the sampler are correct, and (2) the numerical values that you obtain are correct. To simplify the peer-review process, assume that component 1 corresponds to the point mass at zero, while component 2 corresponds to the Poisson distribution.

### Ref Answer

**Are the parameters initialized in a reasonable way?**

Correctly initializing the indicators c\_1, \ldots, c\_n*c*1​,…,*cn*​ is key. In particular, observations such that x\_i \ne 0*xi*​​=0 must have c\_i = 2*ci*​=2. As for observations for which x\_i = 0*xi*​=0, these can be randomly initialized to either component.

For the other parameters, it is reasonable to initialize \lambda*λ* to the mean of the observations and the weight w*w* to a relatively small value (as we did in the EM algorithm).

Hence, the code for the initialization of the algorithm might look something like:

n = length(x)

cc = rep(0, n)

cc[x==0] = sample(1:2, sum(x==0), replace=T, prob=c(1/2, 1/2))

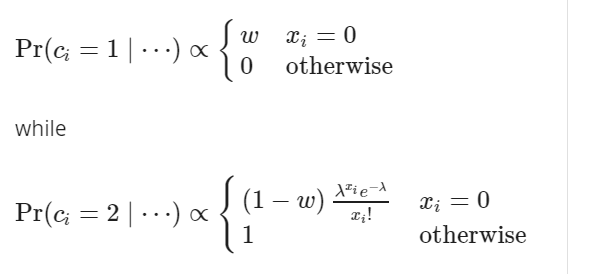
cc[x!=0] = 2

lambda = mean(x)

w = 0.2

**Is the full conditional for the indicators c\_1, \ldots, c\_n*c*1​,…,*cn*​ correct?**

The indicators c\_1, \ldots, c\_n*c*1​,…,*cn*​ are conditionally independent from each other, and we have



Hence, the code to sample the indicators might look something like this:

# Full conditional for cc

for(i in 1:n){

v = rep(0,2)

if(x[i]==0){

v[1] = log(w)

v[2] = log(1-w) + dpois(x[i], lambda, log=TRUE)

v = exp(v - max(v))/sum(exp(v - max(v)))

}else{

v[1] = 0

v[2] = 1

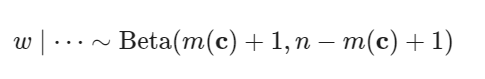
}

cc[i] = sample(1:2, 1, replace=TRUE, prob=v)

}

**Is the full conditional for the weight w*w* correct?**

This is a simple one, as its structure is common to all mixture models. Recalling that the prior on w*w* is a uniform distribution on [0,1][0,1], we have



where m(\mathbf{c})*m*(**c**) is the number of observations that \mathbf{c}**c** assigns to component 1. The associated code might look something like this

# Full conditional for w

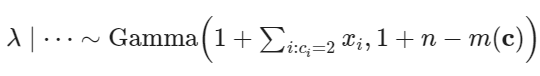
w = rbeta(1, 1+sum(cc==1), 1+n-sum(cc==1))

# Full conditional for w

w = rbeta(1, 1+sum(cc==1), 1+sum(cc==2))

**Is the full conditional for the rate \lambda*λ* correct?**

Because we use a exponential prior on \lambda*λ*, the full conditional posterior is a Gamma distribution,



where, as before, m(\mathbf{c})*m*(**c**) is the number of observations that \mathbf{c}**c** assigns to component 1 (so that n - m(\mathbf{c})*n*−*m*(**c**) is the number of observations assigned to component 2), and \sum\_{i : c\_i = 2} x\_i∑*i*:*ci*​=2​*xi*​ is the sum of the observations assigned to component 2.

Hence, the code for this prompt might look something like:

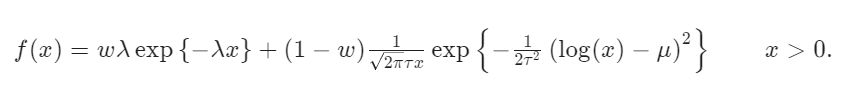
lambda = rgamma(1, sum(x[cc==2]) + 1, sum(cc==2) + 1)

## HW：Markov chain Monte Carlo algorithms for Mixture Models

This assignment is similar to one you already completed in Lesson 3. However, in this case you are asked to use an MCMC algorithm to perform Bayesian inference on the model parameters, instead of an EM algorithm to find maximum likelihood estimators.

Data on the lifetime (in years) of fuses produced by the ACME Corporation is available in the file *fuses.csv*:

In order to characterize the distribution of the lifetimes, it seems reasonable to fit to the data a two-component mixture of the form:



The first component, which corresponds to an exponential distribution with rate *λ*, is used to model low-quality components with a very short lifetime. The second component, which corresponds to a [log-Gaussian distribution](https://en.wikipedia.org/wiki/Log-normal_distribution), is used to model normal, properly-functioning components.

You are asked to modify the implementation of the MCMC algorithm contained in the Reading "Sample code for MCMC example 1" so that you can fit this two-component mixture distributions instead. You then should run your algorithm for 10,000 iterations after a burn-in period of 1,000 iterations and report your estimates of the posterior means, rounded to two decimal places. Assume the following priors:  *w*∼Uni[0,1], *λ*∼Exp(1), *μ*∼Normal(0,1) and *τ*2∼IGam(2,1).

**Review criteria更少**

The code you generate should follow the same structure as "Sample code for MCMC example 1".  In particular, focus on a Gibss sampler that alternates between the full conditionals for w*w*, \lambda*λ*, \mu*μ*, \tau^2*τ*2 and the latent component indicators *c*1​,…,*cn*​.  Peer reviewers will be asked to check whether the different pieces of code have been adequately modified to reflect the fact that (1) parameters have been initialized in a reasonable way, (2) each of the two full conditional distributions associated with the sampler are correct, and (2) the numerical values that you obtain are correct.  To simplify the peer-review process, assume that component 1 corresponds to the exponential distribution, while component 2 corresponds to the log-Gaussian distribution.

### Ref answer

**Are the initial values appropriate?**

The starting values of four parameters, *w*, *λ*, *μ* and *τ*, need to be specified, and the context of the problem provides some useful clues.

Because the lognormal component corresponds to the "normal" components, and we expect the majority of the observations to be in this class, it makes sense to bias the weights so that  *w*≤1/2. For example, we could use  *w*=0.1, or simply sample a random starting from something like a Beta(1,9) distribution (which has expectation 0.1).

For the same reason, a reasonable starting values for *μ* and *τ* correspond to their maximum likelihood estimators under the simpler log-Gaussian model. Since a random variable follows a log-Gaussian distribution **if and only if its logarithm follows a Gaussian distribution,** we can use *μ*=*mean*(*log*(*x*)) and *τ*=*sd*(*log*(*x*)) as our starting values. Alternatively, the values could be sampled from distributions centered around these values in order to make the starting values random.

Because the defective components should have shorter lifespan than normal components, it makes sense to take 1/*λ* (which is the mean of the first component) to be a small fraction of the overall mean of the data (we use 5% of the overall mean, but other similar values would all be reasonable).

Finally, in this case the indicators c\_1, \ldots, c\_n*c*1​,…,*cn*​ could be randomly initialized to either component of the mixture.

In summary, you should expect an initialization such as this one:

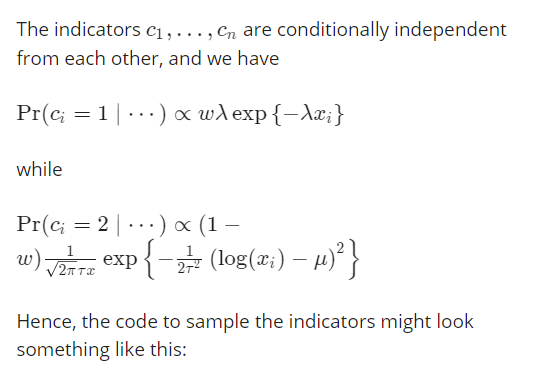
w = 0.1

mu = mean(log(x))

tau = sd(log(x))

lambda = 20/mean(x)

cc = sample(1:2, n, TRUE, c(1/2, 1/2))



# Full conditional for cc

v = rep(0,2)

for(i in 1:n){

v[1] = log(w) + dexp(x[i], lambda, log=TRUE)

v[2] = log(1-w) + dlnorm(x[i], mu, tau, log=TRUE)

v = exp(v - max(v))/sum(exp(v - max(v)))

cc[i] = sample(1:2, 1, replace=TRUE, prob=v)

}

Please be open-minded when reviewing this code, as there are many ways to accomplish the same thing in R. For example, the calculation could be vectorized to increase efficiency.

**Is the full conditional for the weight w*w* correct?**

This is a simple one, as its structure is common to all mixture models. Recalling that the prior on w*w* is a uniform distribution on [0,1][0,1], we have

*w*∣⋯∼Beta(*m*(**c**)+1,*n*−*m*(**c**)+1)

where m(\mathbf{c})*m*(**c**) is the number of observations that \mathbf{c}**c** assigns to component 1. The associated code might look something like this

# Full conditional for w

w = rbeta(1, 1+sum(cc==1), 1+sum(cc==2))

**Is the full conditional for the weight \lambda*λ* correct?**

Since the prior is conditionally conjugate, it is easy to see that

*λ*∣⋯Gam(1+*m*(**c**),1+∑*i*:*ci*=1*xi*)

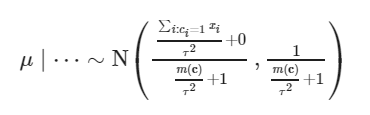
Hence, the code might look something like

# Full conditional for lambda

lambda = rgamma(1, 1 + sum(cc==1), 1 + sum(x[cc==1])

**Is the full conditional for the weight \mu*μ* correct?**

The full-conditional for \mu*μ* is a Gaussian distribution,



The corresponding R code might look like:

# Full conditional for mu

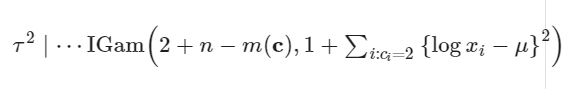
mean.post = (sum(log(x[cc==2]))/tau^2 + 0)/(sum(cc==2)/tau^2 + 1)

std.post = sqrt(1/(sum(cc==2)/tau^2 + 1))

mu = rnorm(1, mean.post, std.post)

**Is the full conditional for the weight \tau*τ* correct?**

The full conditional for \tau^2*τ*2 is an inverse Gamma distribution,



# Full conditional for tau

tau = sqrt(1/rgamma(1, 2 + sum(cc==2), 1 + sum((log(x[cc==2]) - mu)^2)))

**Are the posterior means of the parameters generated by the algorithm correct?**

The posterior means, rounded to two decimal places, are E{*w*}≈0.10, E{*λ*}≈2.29, E{*μ*}≈0.79 and E{*τ*}≈0.38

# Chr4

## HW：The EM algorithm and density estimation

The R dataset *faithful* contains data on waiting time between eruptions (the column named *waiting*) and the duration of the eruption (the column named *eruptions*) for the famous Old Faithful geyser in Yellowstone National Park, Wyoming, USA.

喷发间隔时间，喷发持续时间

You are asked to modify the EM algorithm provided in "Sample code for density estimation problems" to provide a density estimate the marginal distribution of the duration of **the eruptions** using a location-and-scale mixture of 2 univariate Gaussian distributions (as opposed to the location mixture of 6 univariate Gaussian distributions that we used for the *galaxies* dataset).

Reviewers will check whether the code has been modified correctly, and whether the density estimate you generate appears correct. Please remember that you are being asked to use a **location-and-scale mixture to generate the density estimate,** so the "Sample code for density estimation problems" cannot be used directly and requires some modification. Before submitting your answer, it might be useful to compare the density estimate generated by your algorithm against a kernel density estimate generated by the R function density(). While they should not be identical, they should be similar.

### Ref Answer

**Is the E step implemented correctly?**

There are a couple of ways in which the E step can be implemented. This is an example that uses w*w* and 1-w1−*w* as the weights of the two components

## E step

v = array(0, dim=c(n,2))

for(i in 1:n){

v[i,1] = log(w) + dnorm(x[i], mu[1], sigma[1], log=T)

v[i,2] = log(1-w) + dnorm(x[i], mu[2], sigma[2], log=T)

v[i,] = exp(v[i,] - max(v[i,]))/sum(exp(v[i,] - max(v[i,])))

}

**Is the M step implemented correctly?**

The exact form of the implementation of the M step also depends on how the weights are parameterized. Assuming that w*w* and 1-w1−*w* are used,

## M step

# Weights

w = mean(v[,1])

# Means

mu = rep(0, 2)

for(k in 1:2){

for(i in 1:n){

mu[k] = mu[k] + v[i,k]\*x[i]

}

mu[k] = mu[k]/sum(v[,k])

}

# Variances

sigma = rep(0,2)

for(k in 1:2){

for(i in 1:n){

sigma[k] = sigma[k] + v[i,k]\*(x[i] - mu[k])^2

}

sigma[k] = sqrt(sigma[k]/sum(v[,k]))

}

## HW：MCMC algorithms and density estimation

The R dataset *faithful* contains data on waiting time between eruptions (the column named *waiting*) and the duration of the eruption (the column named *eruptions*) for the famous Old Faithful geyser in Yellowstone National Park, Wyoming, USA.

In this case, you are asked to modify the MCMC algorithm provided in "Sample code for density estimation problems" (as opposed to the EM algorithm you used in the previous peer assignment) to provide a (Bayesian) **density estimate the marginal distribution of the duration of the eruptions u**sing a location-and-scale mixture of **2 univariate Gaussian distributions** (as opposed to the location mixture of 6 univariate Gaussian distributions that we used for the *galaxies* dataset). Assume that the priors are***w*∼Beta(1,1), *μk*∼Normal(*η*,*τ*2) and 1/*σ*2*k*∼Gamma(*d*,*q*), where *η*, *τ*2, *d* and *q* a**re selected using an empirical Bayes approach similar to the one we used in "Sample code for density estimation problems".

Reviewers will check whether the code has been modified correctly, and whether the density estimate you generate appears correct. Please remember that you are being asked to use a location-and-scale mixture to generate the density estimate, so the "Sample code for density estimation problems" cannot be used directly and requires some modification. Before submitting your answer, it might be useful to compare the density estimate generated by your algorithm against a kernel density estimate generated by the R function density(), and agains the answer to the previous peer assignment.

## HW：Classification

The data set banknote contains six measurements **(length of bill, width of left edge, width of right edge, bottom margin width, top margin width, and length of diagonal, all in mm)** made on 100 genuine and 100 counterfeit old-Swiss 1000-franc bank notes:

数据集钞票包含六种测量值（钞票长度，左边缘宽度，右边缘宽度，下边距宽度，上边距宽度和对角线长度，均以毫米为单位），这些测量是在100枚真钞和100枚伪造的旧瑞士1000硬币上进行的 法郎钞票。

To load the dataset in R, use the command *load("banknoteclassification.Rdata")*, but first make sure that your working directory is set to the directory containing the file. You should see four objects:

* *banknote.training* contains the characteristics for 30 notes (15 genuine and 15 counterfeit) in the training set.

训练集包含30张纸币(15张真钞和15张假钞)的特征。

* *banknote.training.labels* contains the labels ("genuine" or "counterfeit") for the 30 notes in the training set
* *banknote.test* contains the characteristics for 170 notes (85 genuine and 85 counterfeit) in the test set.
* *banknote.test.labels* contains the labels ("genuine" or "counterfeit") for the 170 notes in the test set. **These are provided only for validation purposes.**

You are asked to modify the MCMC algorithm in "Sample code for MCMC example 2" to create an algorithm for **semi-supervised classification** that is the Bayesian equivalent of that provided under "Sample EM algorithm for classification problems" and **apply it to classify the observations contained in the test set.** You are then asked to compare your results against those generated by the***qda* function in R**.

As your priors, **use distributions in the same families as in "Sample code for MCMC example 2"**. **In particular, use a uniform distribution for the weights, multivariate normal distributions for the means of the components, and inverse Wishart priors for the variance-covariance matrices of the components.** The parameters of the priors should be se**t using the same empirical Bayes approach used in that example.**

### Ref Answer

**Is the setup of the algorithm correct?**

Recall that the semisupervised version of the algorithm uses all observations but treats the labels for he training set as known. There are many ways to set this up, but one that requires minimal changes to the algorithm defines the *x* object in the code as the combination of the observations in the training and test sets, then defines *n, m*, *K* and *p*based on the dimensions of the original objects, and initialize the component indicators of the observations in the training set to their true (known) values:

## All observations used for calculation

x = rbind(banknote.training,banknote.test)

## Size of the training and test sets

n = dim(banknote.training)[1]

m = dim(unique(banknote.test))[1]

## Number of components and dimensionality of the components

KK = length(unique(banknote.training.labels))

p = dim(banknote.training)[2]

## Starting value for the indicators

## Note that for the training set we use the true values, while for the

## test set the values are initialized randomly

cc = c(as.numeric(banknote.training.labels), sample(1:KK, m, replace=TRUE, prob=w))

**Is the sampler for c*c* correct?**

The full conditional for the indicators is identical to the one in the original code. The only difference is that the labels for the training set are considered known, and therefore not sampled. In practice, this means a simple change to the values over which the *for* loop iterates:

# Sample the indicators

for(i in (n+1):(n+m)){

v = rep(0,KK)

for(k in 1:KK){

v[k] = log(w[k]) + dmvnorm(x[i,], mu[k,], Sigma[k,,], log=TRUE) #Compute the log of the weights

}

v = exp(v - max(v))/sum(exp(v - max(v)))

cc[i] = sample(1:KK, 1, replace=TRUE, prob=v)

}

Because the code never changes the values of the first n entries of cc, it is important that the setup of the algorithm (see previous prompt in the rubric) initializes them to the true known values.

**Is the sampler for the \mu\_k*μk*​s correct?**

Again, the sampler is identical to the one used before. The only difficulty here is that, **because *n* has a slightly different meaning here than in the original code (it is the size of the training set rather than the total sample size) you need to be careful to ensure the code uses all observations to compute the parameters of the full conditionals.** In the case of the means that is easy (no change is required, as the sample size is not used explicitly).

# Sample the means

DD.st = matrix(0, nrow=p, ncol=p)

for(k in 1:KK){

mk = sum(cc==k)

xsumk = apply(x[cc==k,], 2, sum)

DD.st = solve(mk\*solve(Sigma[k,,]) + solve(DD))

dd.st = DD.st%\*%(solve(Sigma[k,,])%\*%xsumk + solve(DD)%\*%dd)

mu[k,] = as.vector(rmvnorm(1,dd.st,DD.st))

}

**Is the sampler for the \Sigma\_kΣ*k*​s correct?**

Again, the sampler is identical to the one used before. The only difficulty is that, because *n* has a slightly different meaning here than in the original code (it is the size of the training set rather than the total sample size) you need to be careful to ensure the code uses all observations to compute the parameters of the full conditionals. In this case, a change in the code **is** needed as the sample size is explicitly used in the code (see the upper limit of the for loop in line 3 below).

# Sample the variances

xcensumk = array(0, dim=c(KK,p,p))

for(i in 1:(n+m)){ ## Need to loop over all (n+m) observations, not just the first n

xcensumk[cc[i],,] = xcensumk[cc[i],,] + (x[i,] - mu[cc[i],])%\*%t(x[i,] - mu[cc[i],])

}

for(k in 1:KK){

Sigma[k,,] = riwish(nu + sum(cc==k), SS + xcensumk[k,,])

}

**Is the code used to classify observations in the test set correct?**

The classification is based on the probabilities that each observation is classified as "genuine" and "counterfeit". Those probabilities can be estimated using the frequencies for which each c\_i*ci*​ is equal to each of the categories:

probgenuine = rep(NA, m)

for(i in 1:m){

probgenuine[i] = sum(cc.out[-seq(1,burn),n+i]==2)/(rrr-burn)

}

**Is the classification error for the "genuine" class generated by the algorithm correct?**

The value should be 0% (the algorithm perfectly classifies genuine banknotes in the test set).

**Is the classification error for the "counterfeit" class generated by the algorithm correct?**

The value should also be 0% (the algorithm perfectly classifies counterfeit banknotes in the test set).

**Is the classification error for the "counterfeit" class generated by the algorithm correct?**

The value should also be 3.52% (3 errors out of 85 observations). In this case, the function *qda* underperforms compared with the semisupervised Bayesian QDA algorithm.

# Chr5

## Computational considerations for Mixture Models

1.Consider a mixture of three Gaussian distribution with common identity covariance matrix and means \mu\_1 = (0,0)'*μ*1​=(0,0)′, \mu\_2 = (1/3,1/3)'*μ*2​=(1/3,1/3)′ and \mu\_3 = (-2/3,1/3)'*μ*3​=(−2/3,1/3)′.

For an observation x\_i = (31,-23)'*xi*​=(31,−23)′, what is the value of v\_{i,2}*vi*,2​, the probability of the observation being generated by the second component (rounded to three decimal places)?

0.928

mu1=c(0,0)

mu2=c(1/3,1/3)

mu3=c(-2/3,1/3)

xi=c(31,-23)

Sigma=diag(2)

Sigma

d1 = mvtnorm::dmvnorm(xi,mu1,Sigma,log = T)

d2 = mvtnorm::dmvnorm(xi,mu2,Sigma,log = T)

d3 = mvtnorm::dmvnorm(xi,mu3,Sigma,log = T)

d = c(d1,d2,d3)

exp(d2-max(d))/sum(exp(d - max(d)))

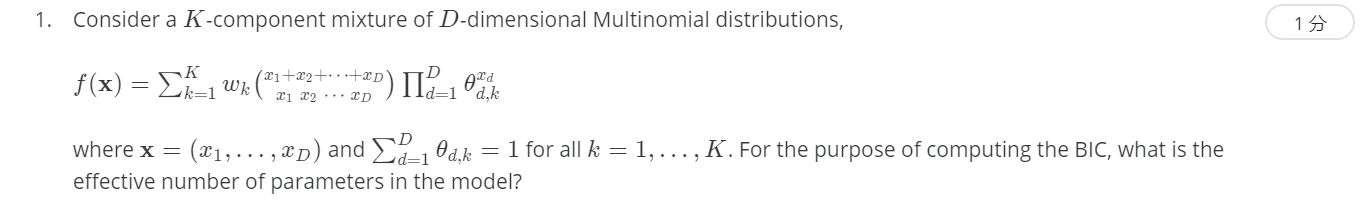
2.True or False: The starting value for the parameters of the mixture model in the EM algorithm could have an impact on the solution you obtain.

True

3.True or False: Consider a Bayesian formulation of a Mixture Model that uses informative priors for all the parameters. A Markov chain Monte Carlo (MCMC) algorithm for fitting such model will fail to work if no observations are allocated to a component of the mixture.

False

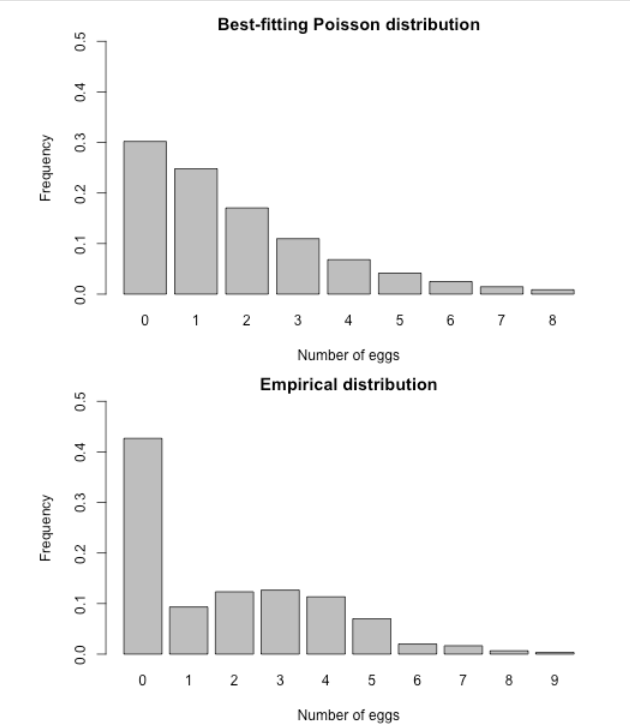
## Bayesian Information Criteria (BIC)



## HW: BIC for zero-inflated mixtures

In week 2, you considered the problem faced by a biologist is interest in characterizing the number of eggs laid by a particular bird species. The data consisted of a sample *n*=300 nests on a site in Southern California, which were contained in the file nestsize.csv:

At the time we visually compared the empirical distribution of the data against a Poison distribution whose parameter has been set to its maximum likelihood estimator as a justification for using a mixture model of the form f(x) = w \delta\_0(x) + (1-w) \frac{e^{-\lambda} \lambda^x}{x!} \quad \quad x \in \{0,1,2,\ldots\}*f*(*x*)=*wδ*0​(*x*)+(1−*w*)*x*!*e*−*λλx*​*x*∈{0,1,2,…} where \delta\_0(x)*δ*0​(*x*) represents the degenerate distribution placing all of its mass at zero.



You are asked to build on the EM algorithm you constructed in week 2 to **compute the BIC associated with this model and contrast it against the BIC for the simpler Poisson model.**

**Review criteria**

You will be asked to provide (1) code to compute the BIC for the zero-inflated mixture (which requires an EM algorithm), (2) code to compute the BIC for the Poisson model (which does not require an EM algorithm) , (3) provide numerical results for the BIC for each of the two models, and (4) interpret the results in terms of the original problem.

## Estimating the number of components in Bayesian settings

.

第 1 个问题

Let K^{\*}*K*∗ be the prior expected number of occupied components in a mixture model with K*K* components where the weights are given a Dirichlet prior (*w*1,…,*wK*)∼Dir(2*K*,…,2*K*). If you have n=400*n*=400 observations, what is the expected number of occupied components, E(K^{\*})*E*(*K*∗) according to the **exact** formula we discussed in the lecture? Round your answer to one decimal place.

11.1

n = 400

alpha = 2

EKstar = 0

for(i in 1:n){

EKstar = EKstar + alpha/(alpha + i -1)

}

print(EKstar)

2.

Consider the same setup as the previous question, what is the expected number of occupied components, E(K^{\*})*E*(*K*∗) according to the **exact** formula we discussed in the lecture if n=100*n*=100 instead? Round your answer to one decimal place.

8.4

n = 100

alpha = 2

EKstar = 0

for(i in 1:n){

EKstar = EKstar + alpha/(alpha + i -1)

}

print(EKstar)

第 3 个问题

What would be the answer to the previous question if you used the **approximate** formula instead of the exact formula? Remember to round your answer to one decimal place.

7.8

n = 100

alpha = 2

alpha\*log((n+alpha-1)/alpha)

第 4 个问题

If you have n=200*n*=200 observations and a priori expect the mixture will have about 2 occupied components (i.e., E(K^{\*}) \approx 2*E*(*K*∗)≈2 a priori), what value of \alpha*α* should you use for the prior (*w*1,…,*wK*)∼Dir(*α/K*,…,*α/K*). Use the approximation E(K^{\*}) \approx \alpha \log\left( \frac{n+\alpha-1}{\alpha} \right)*E*(*K*∗)≈*α*log(*αn*+*α*−1​) to provide an answer, which should be rounded to two decimal places.