Root Cause Determination

Qi Qi

December 15, 2018

Introduction

An open challenge stated in the Quality Engineering Journal: there is a key quality characteristic that is measured once on a medical device component. The characteristic is crucial to ensure that the device will perform as intended. The value has to be sufficiently small to provide a real benefit to the patient. I used EM algorithm to determine the proportion of root cause. The maximum likelihood estimator of proportion of root cause is 0.32 and maximum likelihood estimator of proportion of not root cause is 0.68.

Data

We have extensive history on this characteristic with 1500 data points representing several years of a manufacturing process history. Close inspection of the data reveals that there are a few data points that fall outside of the specification, which is problematic. It appears that there is some skewness on the right side of the distribution, which may be the reason for the data points that fall outside of the specification. Although the frequency of points outside the specification is low, it is crucial to reduce this frequency in order to ensure a high level of quality.

A small experiment was done with 60 total samples, where 30 of them had the root cause and 30 did not. Experimental data shows significant difference in the mean response between these two groups.

Root cause is a possible factor leading to the skewness but this factor is not observed in the data set. So the purpose is to identify whether each subject belongs to root cause group or not, and estimate the frequency of root cause in the historical data set.

Method

I assumed the population with a mixture of Gaussian distribution and use EM-algorithm to estimate the parameters. As long as I can obtain the estimator of proportion fractions, then the frequency of root cause is able to be estimated. I estimated the proportion on observed data first and then pooled observed data and experimental data together to estimate the proportion.

Distribution of Experimental Data

Test the normality of experimental data set:

```r
test <- read.csv("Data for Root Cause Determination - Test Data.csv")
dat <- read.csv("Data for Root Cause Determination.csv")
shapiro.test(test[test$Group == "Test Group - Root Cause",]$Response)
```

```
##
##  Shapiro-Wilk normality test
##
```

```
## data:  test[test$Group == "Test Group - Root Cause", ]$Response
## W = 0.9609, p-value = 0.3266
```

```
shapiro.test(test[test$Group == "Test Group - No Root Cause",]$Response)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  test[test$Group == "Test Group - No Root Cause", ]$Response
## W = 0.95875, p-value = 0.2877
```

From above result, we know for each group the data is normally distributed.

Then I propose mixed normal distribution and mixed gamma distribution of original data set.

Mixture of Normal Distributions

Let z_i be the index of the Gaussian distribution from which x_i is sampled. The parameters to be estimated is $(\mu_1, \mu_2, \sigma_1^2, \sigma_2^2, \xi_1, \xi_2)$, where $\xi_1 + \xi_2 = 1$. Let $\theta_t = (\mu_{t1}, \mu_{t2}, \sigma_{t1}^2, \sigma_{t2}^2, \xi_{t1}, \xi_{t2})$

$$Q(\theta|\theta_t) = \sum_z p(z|x, \theta_t) \ln p(x, z|\theta) = \sum_{i=1}^{n} \sum_{k=1}^{2} p(z_i = k|x_i, \theta_t) \ln p(x_i, z_i = k|\theta)$$

Let $w_{ik} = p(z_i = k|x_i, \theta_t)$, then

$$w_{ik} = \frac{p(z_i = k, x_i, |\theta_t)}{\sum_{k=1}^{2} p(z_i = k, x_i|\theta_t)} = \frac{\xi_{tk}\phi(x_i|\mu_{tk}, \sigma_{tk}^2)}{\sum_{k=1}^{2} \xi_{tk}\phi(x_i|\mu_{tk}, \sigma_{tk}^2)}$$

$$Q(\theta|\theta_t) = \sum_{k=1}^{2} \sum_{i=1}^{n} w_{ik} \ln(\xi_k/\sqrt{2\pi}) - \frac{1}{2} \sum_{k=1}^{2} \sum_{i=1}^{n} w_{ik} \ln \sigma_k^2 - \frac{1}{2} \sum_{k=1}^{2} \sum_{i=1}^{n} w_{ik} \frac{(x_i - \mu_k)^2}{\sigma_k^2}$$

$$\frac{\partial Q(\theta|\theta_t)}{\partial \mu_k} = 0 \Rightarrow \mu_k = \frac{\sum_{i=1}^{n} w_{ik} x_i}{\sum_{i=1}^{n} w_{ik}}$$

$$\frac{\partial Q(\theta|\theta_t)}{\partial \sigma_k^2} = 0 \Rightarrow \sigma_k^2 = \frac{\sum_{i=1}^{n} w_{ik}(x_i - \mu_k)^2}{\sum_{i=1}^{n} w_{ik}}$$

$$\xi_k = \frac{1}{n} \sum_{i=1}^{n} w_{ik}$$

If the classification of some data points is known, then some w_{ik} are known. Suppose $z_1, ..., z_{n_1}$ are unknown, $z_{n_1+1}, ..., z_n$ are known, then $w_{1k}, ..., w_{n_1 k}$ are estimated as above approach and $w_{(n_1+1)k}, ..., w_{nk}$ are equal to either 1 or 0 according to the value of $z_{n_1+1}, ..., z_n$.

Implement in R

R Functions Creation and Evaluation by Simulation Study

```r
mynormalmixEM <- function(x, xi1, xi2, mu1, mu2, sigma1, sigma2, maxit, tol){
  n <- length(x)
  w1 <- double(n)
  w2 <- double(n)
  for (i in 1:maxit){
     for (j in 1:n){
      w1[j] <- xi1 * dnorm(x[j], mu1, sigma1) / (xi1 * dnorm(x[j], mu1, sigma1) + xi2 * dnorm(x[j], mu2
      w2[j] <- 1 - w1[j]
    }
    xi1new <- mean(w1)
    xi2new <- mean(w2)
    mu1new <- sum(w1 * x) / sum(w1)
    mu2new <- sum(w2 * x) / sum(w2)
    sigma1new <- sqrt(sum(w1 * (x - mu1new)^2) / sum(w1))
    sigma2new <- sqrt(sum(w2 * (x - mu2new)^2) / sum(w2))
    if (max(abs(xi1new - xi1), abs(xi2new - xi2), abs(mu1new - mu1), abs(mu2new - mu2), abs(sigma1new -
      xi <- c(xi1new, xi2new)
      mu <- c(mu1new, mu2new)
      sigma <- c(sigma1new, sigma2new)
      iter <- i
      return(list(xi, mu, sigma, iter))
    }
    xi1 <- xi1new
    xi2 <- xi2new
    mu1 <- mu1new
    mu2 <- mu2new
    sigma1 <- sigma1new
    sigma2 <- sigma2new
  }
}

mixnormal <- function(n, xi, mu1, mu2, sigma1, sigma2){
  x <- double(n)
  for (i in 1:n){
    u <- runif(1)
    x[i] <- rnorm(1, ifelse(u < xi, mu1, mu2), ifelse(u < xi, sigma1, sigma2))
  }
  x
}

## simulated data
data <- mixnormal(1000, .6, 3, 8, 1, 1)
mynormalmixEM(data, .5, .5, 6, 7, 1, 4, 1e5, 1e-5)
```

```
## [[1]]
## [1] 0.4154262 0.5845738
##
## [[2]]
## [1] 7.963146 2.957582
##
## [[3]]
## [1] 1.1000188 0.9867722
##
```

```
## [[4]]
## [1] 31
```

```
mynormalpoolEM <- function(x, y, xi1, xi2, mu1, mu2, sigma1, sigma2, maxit, tol){
  n <- length(x) + nrow(y)
  z <- double(n)
  n1 <- length(x)
  z[1:n1] <- x
  z[(n1+1):n] <- y$Response
  w1 <- double(n)
  w2 <- double(n)
  for (i in 1:maxit){
     for (j in 1:n1){
      w1[j] <- xi1 * dnorm(x[j], mu1, sigma1) / (xi1 * dnorm(x[j], mu1, sigma1) + xi2 * dnorm(x[j], mu2
      w2[j] <- 1 - w1[j]
     }
    for (k in (n1+1):n){
      w1[k] <- ifelse(y$Group[k - n1] == "Test Group - No Root Cause", 1, 0)
      w2[k] <- 1 - w1[k]
    }
    xi1new <- mean(w1)
    xi2new <- mean(w2)
    mu1new <- sum(w1 * z) / sum(w1)
    mu2new <- sum(w2 * z) / sum(w2)
    sigma1new <- sqrt(sum(w1 * (z - mu1new)^2) / sum(w1))
    sigma2new <- sqrt(sum(w2 * (z - mu2new)^2) / sum(w2))
    if (max(abs(xi1new - xi1), abs(xi2new - xi2), abs(mu1new - mu1), abs(mu2new - mu2), abs(sigma1new -
      xi <- c(xi1new, xi2new)
      mu <- c(mu1new, mu2new)
      sigma <- c(sigma1new, sigma2new)
      iter <- i
      return(list(xi, mu, sigma, iter))
    }
    xi1 <- xi1new
    xi2 <- xi2new
    mu1 <- mu1new
    mu2 <- mu2new
    sigma1 <- sigma1new
    sigma2 <- sigma2new
  }
}


## simulated pooled data

normmix <- function(n, xi, mu1, mu2, sigma1, sigma2){
  x <- double(n)
  y <- double(n)
  for (i in 1:n){
    u <- runif(1)
    x[i] <- rnorm(1, ifelse(u < xi, mu1, mu2), ifelse(u < xi, sigma1, sigma2))
    y[i] <- ifelse(u < xi, "Test Group - No Root Cause", "Test Group - Root Cause")
  }
  z <- cbind.data.frame(x, y)
  colnames(z) <- c("Response", "Group")
  z
}

data1 <- mixnormal(1000, .6, 3, 8, 1, 1)
data2 <- normmix(1000, .6, 3, 8, 1, 1)
mynormalpoolEM(data1, data2, .5, .5, 2, 5, 1, 1, 1e5, 1e-5)
```

```
## [[1]]
## [1] 0.6049065 0.3950935
##
## [[2]]
## [1] 3.030561 8.027167
##
## [[3]]
## [1] 1.003132 1.028546
##
## [[4]]
## [1] 11
```

From above results, the functions I created to estimate parameters of mixed normal distribution with or without partial known classification can provide accurate results.

Real Data Analysis

```
## experimental data
mynormalmixEM(test$Response, .7, .3, 100, 110, 3, 5, 1e5, 1e-5)
```

```
## [[1]]
## [1] 0.3006259 0.6993741
##
## [[2]]
## [1]  96.59844 108.39758
##
## [[3]]
## [1] 3.344419 4.533869
##
## [[4]]
## [1] 113
```

```
## experimental data (pretend unknown classification) and experimental data (known classification)
mynormalpoolEM(test$Response, test, .7, .3, 100, 110, 3, 5, 1e5, 1e-5)
```

```
## [[1]]
## [1] 0.4950712 0.5049288
##
## [[2]]
## [1] 100.5072 109.1089
##
```

```
## [[3]]
## [1] 5.752996 4.901159
##
## [[4]]
## [1] 17
```

```
## observed data (unknown classification)
mynormalmixEM(dat$Response, .7, .3, 100, 110, 3, 5, 1e5, 1e-5)
```

```
## [[1]]
## [1] 0.93670032 0.06329968
##
## [[2]]
## [1] 100.7110 113.6586
##
## [[3]]
## [1] 5.139339 3.633498
##
## [[4]]
## [1] 1672
```

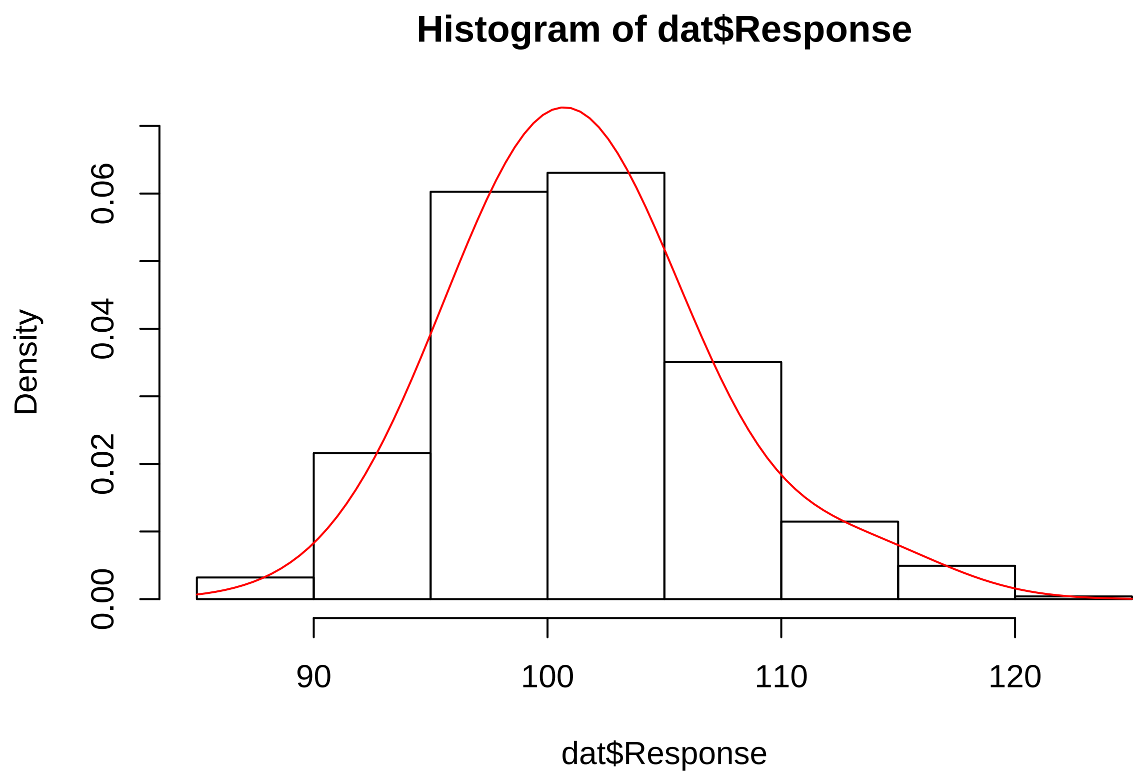
```
## observed data (unknown classification) and experimental data (known classification)
mynormalpoolEM(dat$Response, test, .7, .3, 100, 110, 3, 5, 1e5, 1e-5)
```

```
## [[1]]
## [1] 0.6731524 0.3268476
##
## [[2]]
## [1]  99.46382 106.17778
##
## [[3]]
## [1] 4.661818 6.013368
##
## [[4]]
## [1] 262
```

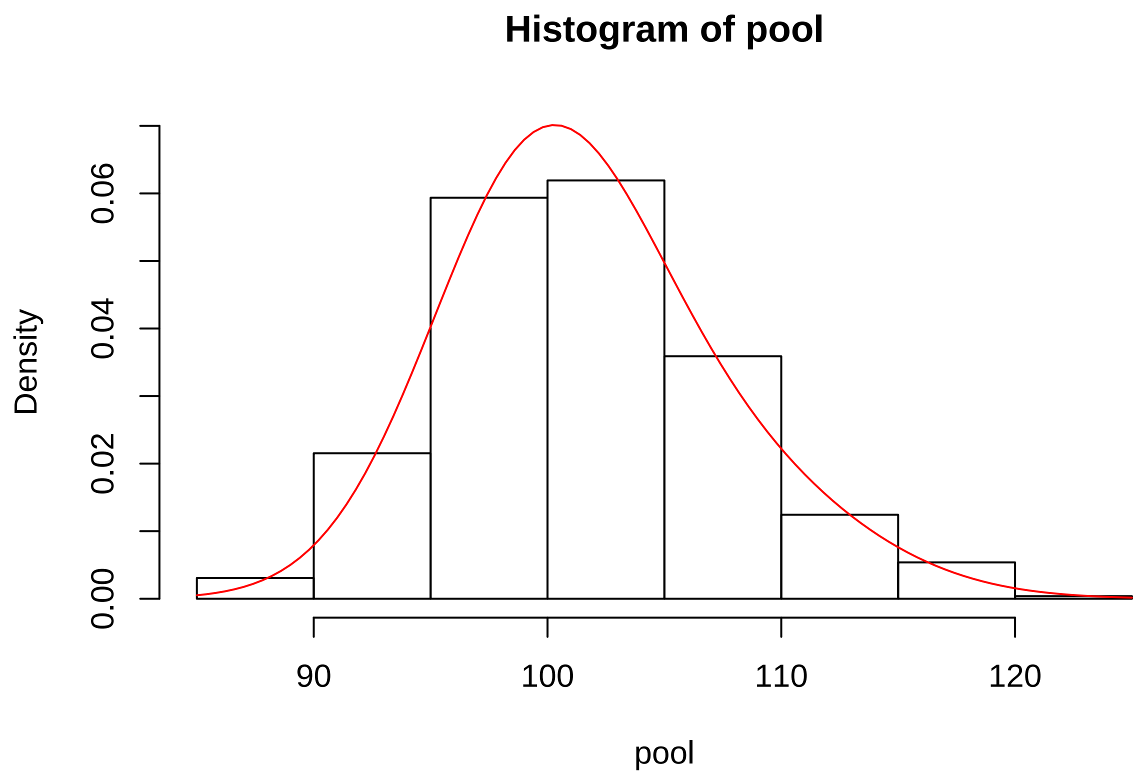
If we only use experimental data, then the estimated proportion parameters are 0.3 and 0.7, which is not close to the truth (0.5 and 0.5). If we use experimental data without classification information first and then use experimental data with classification information, the estimatied proportion parameters are very close to the truth. Therefore, partial classfication information can help to improve the accuracy of parameter estimation.

When I only use observed data, the estimated proportion parameters are 0.937 and 0.063. When I use pooled data, the estimated proportion parameters are 0.673 and 0.327, which means the estimated proportion parameters of observed data are 0.680 and 0.320.

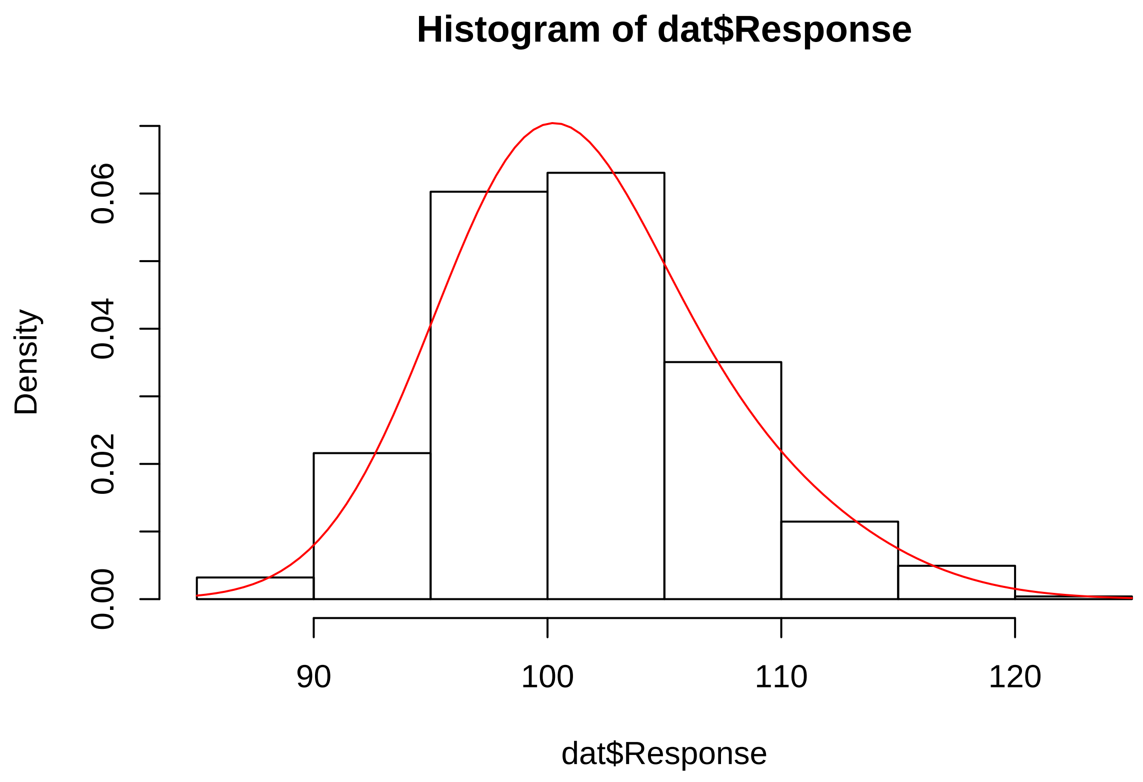
```
## overlay observed data and estimated parameters from observed data
f1 <- function(x){
  0.937 * dnorm(x, 100.711, 5.139) + 0.063 * dnorm(x, 113.659, 3.633)
}
hist(dat$Response, probability = TRUE, ylim = c(0, .07))
curve(f1, add = TRUE, col = "red")
```



```
## overlay pooled data and estimated parameters from pooled data
f2 <- function(x){
  0.673 * dnorm(x, 99.464, 4.662) + 0.327 * dnorm(x, 106.178, 6.013)
}
pool <- double(nrow(dat) + nrow(test))
pool[1:nrow(dat)] <- dat$Response
pool[-(1:nrow(dat))] <- test$Response
hist(pool, probability = TRUE, ylim = c(0, .07))
curve(f2, add = TRUE, col = "red")
```



```
## overlay observed data and estimated parameters from pooled data
f3 <- function(x){
  0.68 * dnorm(x, 99.464, 4.662) + 0.32 * dnorm(x, 106.178, 6.013)
}
hist(dat$Response, probability = TRUE, ylim = c(0, .07))
curve(f3, add = TRUE, col = "red")
```



Confidence Interval

Using bootstrap to estimate confidence interval of proportion:

```
mybootCI <- function(x, y, B){
  n1 <- length(x)
  n2 <- nrow(y)
  p <- matrix(0, B, 2)
  for (i in 1:B){
    x1 <- sample(x, n1, replace = TRUE)
    y1 <- y[sample(n2, n2, replace = TRUE), ]
    p[i, ] <- unlist(mynormalpoolEM(x1, y, .7, .3, 100, 110, 3, 5, 1e5, 1e-5)[1])
  }
  pl <- apply(p, 2, function(x) quantile(x, 0.25))
  pu <- apply(p, 2, function(x) quantile(x, 0.75))
  ci1 <- c(pl[1], pu[1])
  ci2 <- c(pl[2], pu[2])
  return(list(ci1, ci2))
}

mybootCI(dat$Response, test, 1000)
```

```
## [[1]]
## [1] 0.6676913 0.6812556
```

```
##
## [[2]]
## [1] 0.3187444 0.3323087
```

Discussion

According to the results from pooled EM algorithm on simulation study and experimental data, using partial known classification information can improve the accuracy of estimation. From the histograms of observed data, the distribution from pooled EM algorithm fits the data better. Thus, the maximum likelihood estimator of proportion of root cause is 0.32 and maximum likelihood estimator of proportion of not root cause is 0.68.

Since the Since sample size is only 30 in each group from experimental data, we do not have much power. I also tested goodness of fit of Gamma distribution and the results show gamma distribution also fits the data. I also proposed to use mixture of gamma distributions to estimate the proportion. The process and functions are attached in the appendix. The functions are not robust. For some initial value of parameters, the function will return error messages. My next step is to improve those two functions and estimate the proportion by mixed gamma distribution.

Appendix

Mixture of Gamma Distributions

Goodness of fit test of Gamma distribution:

```r
gamma_test(test[test$Group == "Test Group - Root Cause",]$Response)
```

```
##
##  Test of fit for the Gamma distribution
##
## data:  test[test$Group == "Test Group - Root Cause", ]$Response
## V = -1.2239, p-value = 0.3868
```

```r
gamma_test(test[test$Group == "Test Group - No Root Cause",]$Response)
```

```
##
##  Test of fit for the Gamma distribution
##
## data:  test[test$Group == "Test Group - No Root Cause", ]$Response
## V = -0.33574, p-value = 0.8123
```

Above results show gamma distribution also fits the data set.

Let z_i be the index of the Gaussian distribution from which x_i is sampled. The parameters to be estimated is $(\alpha_1, \alpha_2, \beta_1, \beta_2, \xi_1, \xi_2)$, where $\xi_1 + \xi_2 = 1$. Let $\theta_t = (\alpha_{t1}, \alpha_{t2}, \beta_{t1}, \beta_{t2}, \xi_{t1}, \xi_{t2})$

$$Q(\theta|\theta_t) = \sum_z p(z|x, \theta_t) \ln p(x, z|\theta) = \sum_{i=1}^{n} \sum_{k=1}^{2} p(z_i = k|x_i, \theta_t) \ln p(x_i, z_i = k|\theta)$$

Let $w_{ik} = p(z_i = k|x_i, \theta_t)$, then

$$w_{ik} = \frac{p(z_i = k, x_i, |\theta_t)}{\sum_{k=1}^{2} p(z_i = k, x_i|\theta_t)} = \frac{\xi_{tk}\Gamma(x_i|\alpha_{tk}, \beta_{tk})}{\sum_{k=1}^{2} \xi_{tk}\Gamma(x_i|\alpha_{tk}, \beta_{tk})}$$

$$Q(\theta|\theta_t) = \sum_{k=1}^{2}\sum_{i=1}^{n} w_{ik} \ln \xi_k - \sum_{k=1}^{2}\sum_{i=1}^{n} w_{ik} \ln \Gamma(\alpha_k) - \sum_{k=1}^{2}\sum_{i=1}^{n} w_{ik}\alpha_k \ln \beta_k + \sum_{k=1}^{2}\sum_{i=1}^{n} w_{ik}(\alpha_k - 1) \ln x_i - \sum_{k=1}^{2}\sum_{i=1}^{n} \frac{w_{ik}x_i}{\beta_k}$$

$$\xi_k = \frac{1}{n}\sum_{i=1}^{n} w_{ik}$$

$$\frac{\partial Q(\theta|\theta_t)}{\partial \alpha_k} = 0 \Rightarrow \frac{\Gamma(\alpha_k)'}{\Gamma(\alpha_k)} = \frac{\sum_{i=1}^{n} w_{ik} \ln x_i - \sum_{i=1}^{n} w_{ik} \ln \beta_k}{\sum_{i=1}^{n} w_{ik}}$$

$$\frac{\partial Q(\theta|\theta_t)}{\partial \beta_k} = 0 \Rightarrow \beta_k = \frac{\sum_{i=1}^{n} w_{ik}x_i}{\alpha_k \sum_{i=1}^{n} w_{ik}}$$

$$\Rightarrow \frac{\Gamma(\alpha_k)'}{\Gamma(\alpha_k)} = \frac{\sum_{i=1}^{n} w_{ik} \ln x_i - \sum_{i=1}^{n} w_{ik} \ln \frac{\sum_{i=1}^{n} w_{ik}x_i}{\alpha_k \sum_{i=1}^{n} w_{ik}}}{\sum_{i=1}^{n} w_{ik}}$$

Let $\Psi(\alpha_k) = \frac{\Gamma(\alpha_k)'}{\Gamma(\alpha_k)}$,

$$f(\alpha_k) = \Psi(\alpha_k) - \frac{b - a \ln \frac{c}{a\alpha_k}}{a}$$

where $a = \sum_{i=1}^{n} w_{ik}$, $b = \sum_{i=1}^{n} w_{ik} \ln x_i$, $c = \sum_{i=1}^{n} w_{ik}x_i$.

$$f'(\alpha_k) = \Psi'(\alpha_k) - \frac{1}{x}$$

If the classification of some data points is known, then some w_{ik} are known. Suppose $z_1, ..., z_{n_1}$ are unknown, $z_{n_1+1}, ..., z_n$ are known, then $w_{1k}, ..., w_{n_1k}$ are estimated as above approach and $w_{(n_1+1)k}, ..., w_{nk}$ are equal to either 1 or 0 according to the value of $z_{n_1+1}, ..., z_n$.

Implement in R

```
mygammamixEM <- function(x, xi1, xi2, alpha1, alpha2, beta1, beta2, maxit, tol){
  n <- length(x)
  w1 <- double(n)
  w2 <- double(n)
  for (i in 1:maxit){
     for (j in 1:n){
      w1[j] <- xi1 * dgamma(x[j], alpha1, beta1) / (xi1 * dgamma(x[j], alpha1, beta1) + xi2 * dgamma(x[
      w2[j] <- 1 - w1[j]
    }
    xi1new <- mean(w1)
    xi2new <- mean(w2)
    alpha1new <- uniroot(function(y) digamma(y) - (sum(w1 * log(x)) - sum(w1) * log(sum(w1 * x) / y / su
    alpha2new <- uniroot(function(y) digamma(y) - (sum(w2 * log(x)) - sum(w2) * log(sum(w2 * x) / y / su
    beta1new <- sum(w1 * x) / alpha1new / sum(w1)
    beta2new <- sum(w2 * x) / alpha2new / sum(w2)
    if (max(abs(xi1new - xi1), abs(xi2new - xi2), abs(alpha1new - alpha1), abs(alpha2new - alpha2), abs
      xi <- c(xi1new, xi2new)
      alpha <- c(alpha1new, alpha2new)
      beta <- c(beta1new, beta2new)
      iter <- i
      return(list(xi, alpha, beta, iter))
    }
    xi1 <- xi1new
    xi2 <- xi2new
    alpha1 <- alpha1new
    alpha2 <- alpha2new
    beta1 <- beta1new
    beta2 <- beta2new
  }
  iter <- i
  return(iter)
}

mixgamma <- function(n, xi, alpha1, alpha2, beta1, beta2){
  x <- double(n)
  for (i in 1:n){
    u <- runif(1)
    x[i] <- rgamma(1, shape = ifelse(u < xi, alpha1, alpha2), scale = ifelse(u < xi, beta1, beta2))
  }
  x
}


mygammapoolEM <- function(x, y, xi1, xi2, alpha1, alpha2, beta1, beta2, maxit, tol){
  n <- length(x) + nrow(y)
  z <- double(n)
  n1 <- length(x)
  z[1:n1] <- x
  z[(n1+1):n] <- y$Response
  w1 <- double(n)
  w2 <- double(n)
  for (i in 1:maxit){
     for (j in 1:n1){
      w1[j] <- xi1 * dgamma(x[j], alpha1, beta1) / (xi1 * dgamma(x[j], alpha1, beta1) + xi2 * dgamma(x[
      w2[j] <- 1 - w1[j]
     }
    for (k in (n1+1):n){
      w1[k] <- ifelse(y$Group[k - n1] == "Test Group - No Root Cause", 1, 0)
      w2[k] <- 1 - w1[k]
    }
    xi1new <- mean(w1)
    xi2new <- mean(w2)
    alpha1new <- uniroot(function(y) digamma(y) - (sum(w1 * log(x)) - sum(w1) * log(sum(w1 * x) / y / s
    alpha2new <- uniroot(function(y) digamma(y) - (sum(w2 * log(x)) - sum(w2) * log(sum(w2 * x) / y / s
    beta1new <- sum(w1 * x) / alpha1new / sum(w1)
    beta2new <- sum(w2 * x) / alpha2new / sum(w2)
    if (max(abs(xi1new - xi1), abs(xi2new - xi2), abs(alpha1new - alpha1), abs(alpha2new - alpha2), abs
      xi <- c(xi1new, xi2new)
      alpha <- c(alpha1new, alpha2new)
      beta <- c(beta1new, beta2new)
      iter <- i
```

```r
      return(list(xi, alpha, beta, iter))
    }
    xi1 <- xi1new
    xi2 <- xi2new
    alpha1 <- alpha1new
    alpha2 <- alpha2new
    beta1 <- beta1new
    beta2 <- beta2new
  }
  iter <- i
  return(iter)
}
```