Reporte de estancia

Alejandro Nieto Ramos, Marc Lavielle (asesor)

1 de agosto de 2016

El algoritmo EM

El algoritmo EM en general

Sea $\boldsymbol{Y}$ un vector aleatorio que corresponde con los datos observados $\boldsymbol{y}$ y cuya función de densidad de probabilidad es $g(\boldsymbol{y};\boldsymbol{\Psi})$, donde $\boldsymbol{\Psi} = (\Psi_1, \ldots, \Psi_d)$ es un vector de parámetros desconocidos con espacio parametral $\boldsymbol{\Omega}$.

El algoritmo Esperanza Maximización (**EM**) permite calcular estimadores de máxima verosimilitud (**EML**) a través de un proceso iterativo, cuando por falta de datos adicionales, la estimación por máxima verosimilitud sería directa. Desde este punto de vista, al vector de datos observados $\boldsymbol{y}$ se le considera como incompleto y a su vez, se toma como una función observable de los datos completos. Ahora bien, cuando se habla de datos incompletos, no solo se hace referencia a la falta de datos en sí, sino también a situaciones donde los datos incompletos representen lo que se podría observar en un experimento hipotético. En este último contexto, se define a $\boldsymbol{y_c}$ como el vector de los datos completos y a $\boldsymbol{z}$ como al vector de los datos adicionales, es decir, los no observables o faltantes.

Sea entonces $g_c(\boldsymbol{x};\boldsymbol{\Psi})$ la función de densidad de probabilidad del vector aleatorio $\boldsymbol{Y_c}$ que corresponde a los datos completos. Por lo tanto, la función de verosimilitud de $\boldsymbol{\Psi}$ si se tuvieran los datos completos, estaría dada por $\log L_c(\boldsymbol{\Psi}) = \log g_c(\boldsymbol{y_c};\boldsymbol{\Psi})$.

El algoritmo EM resuelve indirectamente el problema de hallar la solución a la ecuación

$$\frac{\partial}{\partial \boldsymbol{\Psi}} \log L(\boldsymbol{\Psi}) = 0, \tag{1}$$

iterando en términos de la función de verosimilitud de los datos completos $\log g_c(\boldsymbol{y_c})$. Como no se observa, se cambia por su esperanza condicional dado $\boldsymbol{y}$, usando la aproximación actual de $\boldsymbol{\Psi}$.

Es decir, sea $\boldsymbol{\Psi}^{(0)}$ un valor inicial de $\boldsymbol{\Psi}$. En la primera iteración, el paso E requiere del cálculo de

$$Q(\boldsymbol{\Psi};\boldsymbol{\Psi}^{(0)}) = E_{\boldsymbol{\Psi}^{(0)}}[\log L_c(\boldsymbol{\Psi})|\boldsymbol{y}].$$

El paso M requiere de la maximización de $Q(\boldsymbol{\Psi};\boldsymbol{\Psi}^{(0)})$ con respecto a $\boldsymbol{\Psi}$ sobre el espacio parametral $\boldsymbol{\Omega}$. Es decir, se toma a $\boldsymbol{\Psi}^{(1)}$ tal que $Q(\boldsymbol{\Psi}^{(1)};\boldsymbol{\Psi}^{(0)}) \geq Q(\boldsymbol{\Psi};\boldsymbol{\Psi}^{(0)})$ para todos los $\boldsymbol{\Psi} \in \boldsymbol{\Omega}$. Los pasos E y M se aplican nuevamente cambiando ahora la aproximación actual por $\boldsymbol{\Psi}^{(1)}$.

En general, en la iteración $k+1$, el algoritmo es el siguiente:

Paso E. Se calcula $Q(\boldsymbol{\Psi};\boldsymbol{\Psi}^{(k)})$, donde $Q(\boldsymbol{\Psi};\boldsymbol{\Psi}^{(k)}) = E_{\boldsymbol{\Psi}^{(k)}}[\log L_c(\boldsymbol{\Psi})|\boldsymbol{y}]$.

Paso M. Se elige $\boldsymbol{\Psi}^{k+1}$ como cualquier valor $\boldsymbol{\Psi} \in \boldsymbol{\Omega}$ que maximice $Q(\boldsymbol{\Psi};\boldsymbol{\Psi}^{(k)})$, es decir, $Q(\boldsymbol{\Psi}^{(k+1)};\boldsymbol{\Psi}^{(k)}) \geq Q(\boldsymbol{\Psi};\boldsymbol{\Psi}^{(k)})$ para todo $\boldsymbol{\Psi} \in \boldsymbol{\Omega}$. Se alternan repetidamente ambos pasos hasta que $|L(\boldsymbol{\Psi}^{(k+1)}) - L(\boldsymbol{\Psi}^{(k)})| < \epsilon$, para un $\epsilon > 0$ previamente dado, en caso de que la sucesión $L(\boldsymbol{\Psi})$ converja. Para asegurar que la sucesión sea convergente, basta escoger que esté acotada superiormente[1].

[1] Teorema de Bolzano.

El algoritmo EM para modelos de mezclas

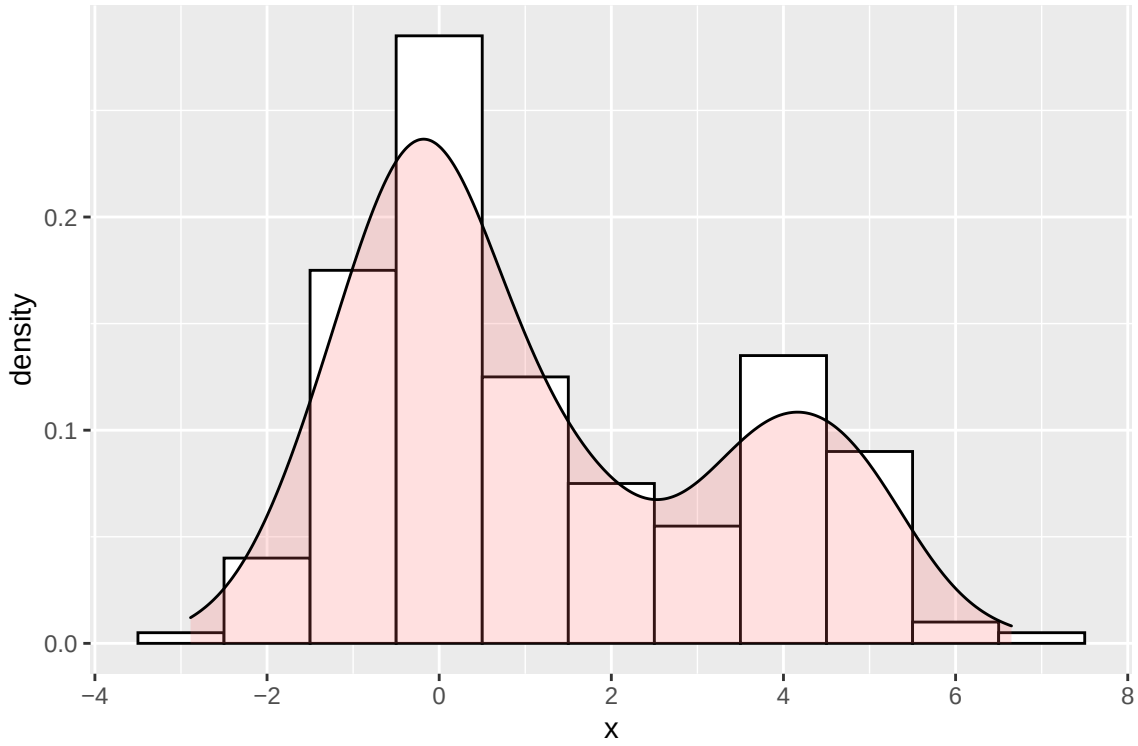
Sea $(Y_1, \ldots, Y_n)$ una muestra aleatoria de tamaño n, donde Y_j es un vector de dimensión p con función de densidad de probabilidad $f(y_j)$ en $\mathbb{R}^p$. Sea $\boldsymbol{Y} = (Y_1^T, \ldots, Y_n^T)^T$ y defínase una observación de $\boldsymbol{Y}$ como $\boldsymbol{y} = (y_1^T, \ldots, y_n^T)^T$. Supóngase que que la densidad $f(y_j)$ de Y_j se puede escribir en la forma

$$f(y_j) = \sum_{i=1}^{g} \pi_i \, f_i(y_j), \tag{2}$$

donde las f_i son las densidades y los valores π_i son tales que $\Sigma_{i=1}^{g} \pi_i = 1$ con $0 \leq \pi_i \leq 1$ para $i = 1, \ldots, g$.

A las cantidades $\pi_1, \ldots, \pi_g$ se les conoce como proporciones de la mezcla o pesos. Dado que las funciones $f_1(y_j), \ldots, f_g(y_j)$ son densidades, la ecuación (**??**) define una densidad. A las $f_i(y_j)$ se les conoce como densidades componentes de la mezcla. Cabe señalar que en esta formulación, el número de componentes se considera fijo, aunque en muchas aplicaciones pueda ser desconocido.[2] A la densidad (**??**) se le conoce como densidad de una mezcla finita de g componentes.

```
mezcla<-function(n,pesos,mu,sigma)
{
  G <- length(mu)
  Z <- sample(1:G, n, prob=pesos, replace=T)
  x<-NULL
  for (g in 1:G)
  {
    x<-c(x,rnorm(length(which(Z==g)),mu[g],sigma[g]))
  }
  return(x)
}
```



Gráfica de la mezcla de dos normales, la primera con $\mu = 0, \sigma = 1$ y una proporción del 70%, y la segunda con $\mu = 4, \sigma = 1$ (y una proporción del 30%).

[2] En un caso así, el número de componentes se tendría que aproximar.

Interpretación

Una manera clara de generar un vector aleatorio Y_{ij} con la mezcla de g densidades componente $f(y_j)$ es la siguiente. Defínase a Z_j como la variable aleatoria categórica que toma valores en $1, \ldots, g$ con probabilidades respectivas $\pi_1, \ldots, \pi_g$, y supóngase que la densidad condicional de Y_j dado $Z_j = i$ es $f_i(y_j)$ para $i = 1, \ldots, g$. Entonces la densidad marginal de Y_j está dada por $f(y_j)$. Considerando este enfoque, la variable Z_j se puede pensar como la componente de pertenencia del vector Y_j. Se define entonces un vector de componentes de pertenencia de dimensión g, $\boldsymbol{Z_j}$, en lugar de la variable categórica única Z_j, donde el i-ésimo elemento de $\boldsymbol{Z_j}$, $Z_{ij} = (\boldsymbol{Z_j})_i$ es 0 o 1 de acuerdo a si la componente de origen de Y_j en la mezcla es igual o no a i, $i = 1, \ldots, g$. Por lo tanto, $\boldsymbol{Z_j}$ se distribuye de acuerdo a una distribución multinomial que consiste de un experimento en g categorías con probabilidades $\pi_1, \ldots, \pi_g$, es decir, $\Pr(\boldsymbol{Z_j} = \boldsymbol{z_j}) = \pi_1^{z_{1j}} \ldots \pi_g^{z_{gj}}$; ésta se representa como $Mult_g(1, \boldsymbol{\pi})$ donde $\boldsymbol{\pi} = (\pi_1, \ldots \pi_g)$.

Enfoque directo

Sea

$$f(y_j; \boldsymbol{\Psi}) = \sum_{i=1}^{g} \pi_i f_i(y_j; \theta_i) \tag{3}$$

un modelo de mezcla paramétrico de una muestra aleatoria observada $\boldsymbol{y} = (y_i^T, \ldots, y_n^T)^T$, donde $\boldsymbol{\Psi} = (\pi_1, \ldots, \pi_{g-1}, \xi^T)^T$ es el vector que contiene todos los parámetros desconocidos de la mezcla del modelo y ξ es el vector de los parámetros $\theta_1, \ldots, \theta_g$ a priori tomados como distintos. La log verosimilitud de Ψ que puede obtenerse de los datos observados está dada por

$$\log L(\boldsymbol{\Psi}) = \sum_{j=1}^{n} \log(f(y_j; \boldsymbol{\Psi})) \tag{4}$$

$$= \sum_{j=1}^{n} \log \left\{ \sum_{i=1}^{g} \pi_i f_i(y_i; \theta_i) \right\} \tag{5}$$

Si se calcula el EMV de $\boldsymbol{\Psi}$ tomando esta log verosimilitud, es preciso resolver $\frac{\partial}{\partial \boldsymbol{\Psi}} \log L(\boldsymbol{\Psi}) = 0$. Esta ecuación se puede manipular de modo que el EMV, $\hat{\boldsymbol{\Psi}}$, satisfaga

$$\hat{\pi_i} = \sum_{j=1}^{n} \frac{1}{n} \tau_i(y_j; \hat{\boldsymbol{\Psi}}), \tag{6}$$

para todo $i = 1, \ldots, g$, y

$$\sum_{i=1}^{g} \sum_{j=1}^{n} \tau_i(y_j; \hat{\boldsymbol{\Psi}}) \frac{\partial}{\partial \xi} \log f_i(y_j; \hat{\theta_i}) = 0, \tag{7}$$

donde

$$\tau_i(y_j; \boldsymbol{\Psi}) = \frac{\pi_i f_i(y_j; \theta_i)}{\sum_{h=1}^{g} \pi_h f_h(y_j; \theta_h)} \tag{8}$$

es la probabilidad posterior de que y_j pertenezca a la componente i de la mezcla.

Enfoque donde se ve a los datos como incompletos

Retómese el mismo enfoque del que se habló en el algoritmo EM, donde el vector $\boldsymbol{y} = (y_1^T, \ldots, y_n^T)^T$ se toma como incompleto, ya que los vectores de pertenencia a cierta componente asociados, $z_1, \ldots, z_n$, no están disponibles. Con este enfoque, en donde cada y_j se ve como que viene de una de los componentes del modelo de mezcla que se desea ajustar, z_j es un vector de dimensión g con $z_{ij} = (z_j)_i$ igual a 1 o 0 de de acuerdo a si y_j vino o no del componente i-ésimo de la mezcla, para todo $i = 1, \ldots, g$ y $j = 1, \ldots, n$. El vector de datos completo es por lo tanto, $\boldsymbol{y_c} = (y^T, z^T)^T$, donde $\boldsymbol{z} = (z_1^T, \ldots, z_n^T)^T$.

Los vectores de pertenencia a cierta componente $z_1, \ldots, z_n$ se toman como realizaciones de los vectores aleatorios $Z_1, \ldots, Z_n$, donde por la característica de independencia de los datos, se puede asumir que se distribuyen independientes con distribución $Mult_g(1, \pi)$. Asumir esto significa que la distribución del vector de los datos completos $\boldsymbol{Y_c}$ da origen a la distribución apropiada para los datos incompletos $\boldsymbol{Y}$. La log verosimilitud de los datos completos para $\boldsymbol{\Psi}$, $\log L_c(\boldsymbol{\Psi})$ es

$$\log L_c(\boldsymbol{\Psi}) = \sum_{i=1}^{g} \sum_{j=1}^{n} z_{ij} \{\log \pi_i + \log f_i(y_j; \theta_i)\}. \tag{9}$$

Paso E

Se aplica el algoritmo EM a este problema tratando a los z_{ij} como datos faltantes. Como la log verosimilitud de los datos completos, L_c, es lineal con respecto a los datos no observados z_{ij}, el paso E en la iteración $k+1$ requiere únicamente del cálculo de la esperanza condicional actual de Z_{ij} dados los datos observados y, donde Z_{ij} es la variable aleatoria correspondiente a z_{ij}. Ahora bien,

$$\begin{aligned} \mathbb{E}_{\Psi^{(k)}}(Z_{ij}|\boldsymbol{y}) =& \Pr_{\Psi^{(k)}}(Z_{ij} = 1|\boldsymbol{y}) & (10)\\ =& \tau_i(y_j; \boldsymbol{\Psi}^{(k)}), & (11) \end{aligned}$$

y de acuerdo a la ecuación (**??**),

$$\begin{aligned} \tau_i(y_j; \boldsymbol{\Psi}^{(k)}) &= \frac{\pi_i^{(k)} f_i(y_j; \theta_i^{(k)})}{f(y_j; \boldsymbol{\Psi}^{(k)})} & (12)\\ &= \frac{\pi_i^{(k)} f_i(y_j; \theta_i^{(k)})}{\sum_{h=1}^{g} \pi_h^{(k)} f_h(y_j; \theta_h^{(k)})} & (13) \end{aligned}$$

para todo $i = 1, \ldots, g$, $j = 1, \ldots, n$. La cantidad $\tau_i(y_j; \boldsymbol{\Psi}^{(k)})$ es la probabilidad posterior de que el j-ésimo miembro de la muestra con valor observado y_j pertenezca al componente i-ésimo de la mezcla. Aplicando los cálculos hechos en el párrafo anterior, tomando la esperanza condicional de (**??**) dado y se tiene que

$$Q(\boldsymbol{\Psi}; \boldsymbol{\Psi}^{(k)}) = \sum_{i=1}^{g} \sum_{j=1}^{n} \tau_i(y_j; \boldsymbol{\Psi}^{(k)}) \{\log \pi_i + \log f_i(y_j; \theta_i)\}. \tag{14}$$

Paso M

El paso M en la iteración $k+1$, requiere que se maximice de manera global $Q(\boldsymbol{\Psi}; \boldsymbol{\Psi}^{(k)})$ con respecto a $\boldsymbol{\Psi}$ sobre el espacio parametral $\boldsymbol{\Omega}$ para obtener la estimación actualizada $\boldsymbol{\Psi}^{(k+1)}$. Para el modelo de mezcla finita, los estimadores actualizados $\pi_i^{(k+1)}$ de las proporciones de la mezcla π_i se calculan independientemente de los estimadores actualizados $\xi^{(k+1)}$ del vector de parámetros ξ que contiene a los parámetros desconocidos en las densidades componentes.

Si los z_{ij} fueran observables, entonces el EVM de π_i estaría dado por la ecuación (**??**), es decir, $\hat{\pi}_i = \sum_{j=1}^{n} \frac{1}{n} z_{ij}$ para $i = 1, \ldots, g$.

Como el paso E solo reemplaza cada z_{ij} por su esperanza condicional $\tau_i(y_j; \boldsymbol{\Psi}^{(k)})$ en la log verosimilitud de los datos completos, el estimador actualizado de π_i está dado al reemplazar cada z_{ij} en la ecuación anterior por $\tau_i(y_j; \boldsymbol{\Psi}^{(k)})$, es decir

$$\pi_i^{(k+1)} = \sum_{j=1}^{n} \frac{1}{n} \tau_i(y_j; \boldsymbol{\Psi}^{(k)}) \tag{15}$$

para todo $i = 1, \ldots, g$.

Con respecto a la actualización de ξ en el paso M de la iteración $k+1$, se puede ver de la ecuación (**??**) que $\xi^{(k+1)}$ se obtiene como una raíz apropiada de

$$\sum_{i=1}^{g}\sum_{j=1}^{n}\tau_i(y_j;\mathbf{\Psi}^{(k)})\frac{\partial}{\partial\xi}\log f_i(y_j;\theta_i)=0, \tag{16}$$

En ciertos casos, la solución se puede encontrar en forma analítica, como se verá en la siguiente sección. Finalmente, el algoritmo termina al igual que como se explicó al final de la sección **??**.

Mezcla de normales

Sea ahora

$$f(y_j;\mathbf{\Psi})=\sum_{i=1}^{g}\pi_i\,\phi_i(y_j;\mu_i,\Sigma_i), \tag{17}$$

donde $\phi(y_j;\mu_i,\Sigma_i)=\frac{1}{(\sqrt{2\pi})^n\sqrt{|\Sigma_i|}}\exp(-\frac{1}{2}(y_j-\mu_i)^T\Sigma_i^{-1}(y_j-\mu_i))$.

En esta caso, $\mathbf{\Psi}=(\pi_1,\ldots,\pi_g,\xi^T)^T$, donde ξ contiene a los elementos de las componente medias μ_i y las distintas matrices componentes de varianza-covarianza Σ_i, $(i=1,\ldots,g)$.

Se ha visto que en la iteración $k+1$ del paso E, se reemplazan las variables de pertenencia a cierta componente, es decir, los z_{ij}, por sus esperanzas condicionales dadas por sus probabilidades posteriores de pertenecer a los datos observados y_j, $\tau_i(y_j;\Psi^{(k)})$. Por lo tanto,

$$\tau_i(y_j;\mathbf{\Psi})=\frac{\pi_i\,\phi_i(y_j;\mu_i,\Sigma_i)}{\sum_{h=1}^{g}\pi_h\,\phi_i(y_j;\mu_h,\Sigma_h)} \tag{18}$$

para $i=1,\ldots,g$, $j=1,\ldots,n$.

A continuación se presenta el código en R para calcular τ_i en el caso de una de mezcla de funciones qe se distribuyen normales.

```
calculotau<-function(x,teta)
{
  n<-length(x)
  G<-length(teta$p)
  tau<-matrix(NA,n,G)
  for (g in 1:G)
    tau[,g]<-teta$p[g]*dnorm(x,teta$mu[g],teta$sigma[g])
  tau=tau/matrix(rep(rowSums(tau),G),nrow=n)
  return(tau)
}
```

El paso M para componentes normales existe en forma analítica. Las actualizaciones de las medias componentes μ_i y las matrices de varianza covarianza Σ_i son:

$$\mu_i^{(k+1)}=\frac{\sum_{j=1}^{n}\tau_{ij}^{(k)}y_j}{\sum_{j=1}^{n}\tau_{ij}^{(k)}} \tag{19}$$

$$\Sigma_i^{(k+1)}=\frac{\sum_{j=1}^{n}\tau_{ij}^{(k)}(y_j-\mu_i^{(k+1)})(y_j-\mu_i^{(k+1)})^T}{\sum_{j=1}^{n}\tau_{ij}^{(k)}} \tag{20}$$

para $i=1,\ldots,g$, donde $\tau_{ij}^{(k)}=\tau_i(y_j;\Psi^{(k)})$ con $i=1,\ldots,g$ y $j=1,\ldots,n$.

Para simplificar los cálculos computacionales, es conveniente escribir las actualizaciones anteiores (**??**) y (**??**) en términos de las esperanzas condicionales actuales de los estadísticos suficientes S_{i1}, S_{12} y S_{i3} para $\mathbf{\Psi}$ para el enfoque de los datos completos. Es decir,

$$S_{i1}^{(k)} = \sum_{j=1}^{n} \tau_{ij}^{(k)}$$

$$S_{i2}^{(k)} = \sum_{j=1}^{n} \tau_{ij}^{(k)} y_j$$

$$S_{i3}^{(k)} = \sum_{j=1}^{n} \tau_{ij}^{(k)} y_j^2.$$

De este modo, el paso E implementado en R queda como sigue:

```
estadistica<-function(x,Z)
{
  G<-dim(Z)[2]
  M<-dim(Z)[3]
  if (is.na(M))
  {
    M <- 1
    dim(Z) <- c(dim(Z),1)
  }
  s1 <- 0
  s2 <- 0
  s3 <- 0
  for (m in 1:M)
  {
    Z.m <- Z[,,m]
    s1 <- s1 + colSums(Z.m)
    s2 <- s2 + x %*% Z.m
    s3 <- s3 + (x^2) %*% Z.m
  }
  est<-list(s1=s1/M,s2=s2/M,s3=s3/M)
  return(est)
}

pasoE<-function(x,teta)
{
  tau <- calculotau(x,teta)
  s <- estadistica(x,tau)
  return(s)
}
```

Ahora bien, el código en R para calcular el paso M es

```
pasoM<-function(S,n)
{
  p<-S$s1/n
  mu<-S$s2/S$s1
  sigma<-sqrt(S$s3/S$s1-(mu)^2)
  teta<-list(p=p,mu=mu,sigma=sigma)
  return(teta)
}
```

donde la versoilmilitud está dada por

```
logLikelihood <- function(x,df)
{
  K <- dim(df)[1]
  G <- (dim(df)[2]-1)/3
  ll <- NULL
  for (k in (1:K))
  {
    lk <- 0
    for (g in (1:G))
    {
      pi.k <- df[k,g+1]
      mu.k <- df[k,G+g+1]
      sigma.k <- df[k,2*G+g+1]
      lk <- lk + pi.k * dnorm(x, mean=mu.k, sd=sigma.k)
    }
    ll <- c(ll, sum(log(lk)))
  }
  df$deviance <- -2*ll
  return(df)
}
```

El estimador de la i-ésima proporción π_i está dado por la ecuación (**??**).

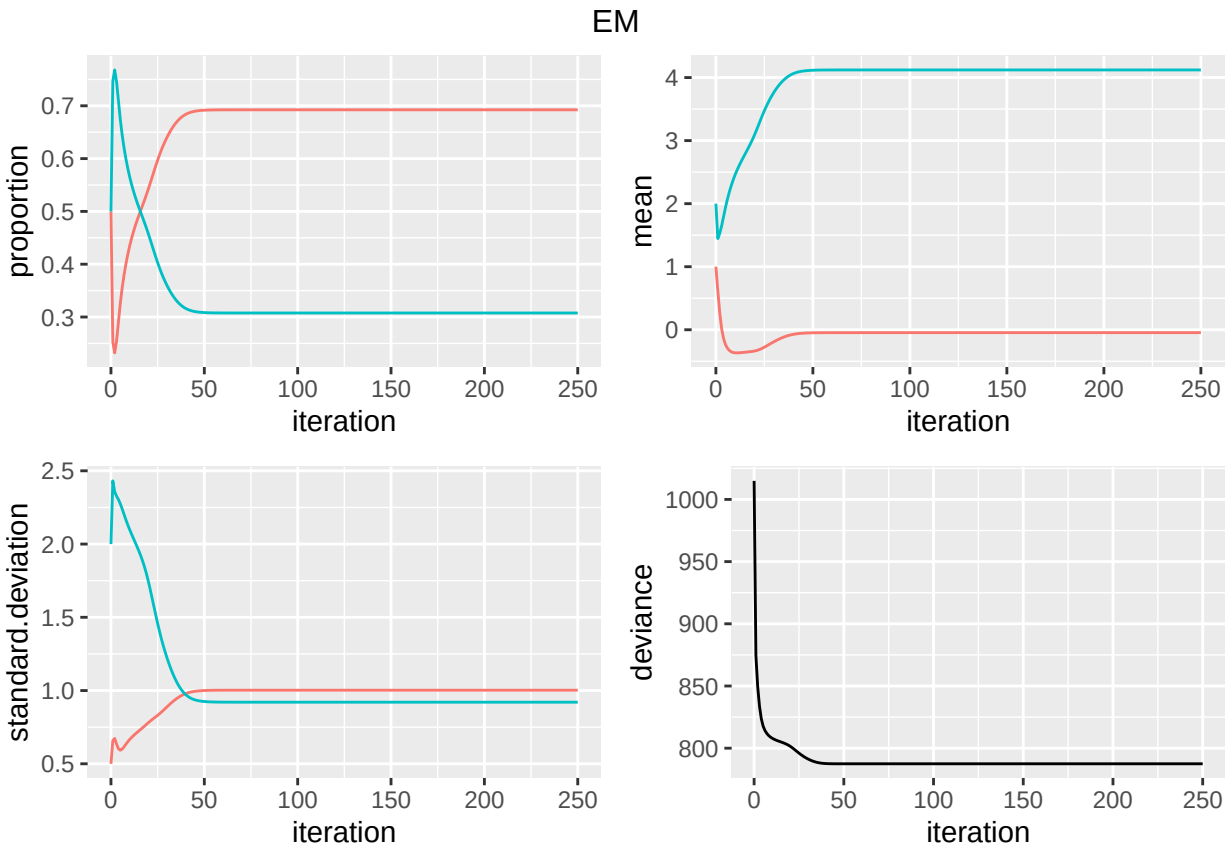
Finalmente, el programa iterativo que hace uso de las funciones anteriores se presenta a continuación:

```
mixt1.em <- function(x, teta0, K)
{
  G<-length(mu)
  col.names <- c("iteration", paste0("p",1:G), paste0("mu",1:G), paste0("sigma",1:G))

  teta.est <- matrix(NA,K+1,3*G+1)
  teta.est[1,] <- c(0, teta0$p, teta0$mu, teta0$sigma)

  teta<-teta0
  for (k in 1:K)
  {
    s<-pasoE(x,teta)
    teta<-pasoM(s,n)
    teta.est[k+1,] <- c(k, teta$p, teta$mu, teta$sigma)
  }
  df.em <- as.data.frame(teta.est)
  names(df.em) <- col.names
  return(df.em)
}

# valores iniciales para el algoritmo
pesos0<-c(.5,.5)
mu0<-c(1,2)
sigma0<-c(.5,2)

K <- 250
M <- 1

G<-length(mu)
```

```r
col.names <- c("iteration", paste0("p",1:G), paste0("mu",1:G), paste0("sigma",1:G))
teta0<-list(p=pesos0,mu=mu0,sigma=sigma0)
s0<-list(s1=0,s2=0,s3=0)

##  EM
df.em <- mixt1.em(x, teta0, K)
df.em <- logLikelihood(x,df.em)
print(round(df.em[nrow(df.em),],3))
```

```
##     iteration    p1    p2    mu1   mu2 sigma1 sigma2 deviance
## 251       250 0.692 0.308 -0.046 4.121  1.002  0.921  787.436
```

```r
graphConvergence <- function(df, title=NULL)
{
  G <- (dim(df)[2]-1)/3
  df1<-melt(df[,c(1,2,3)],id.vars="iteration",variable="proportion")
  graf1 <- ggplot(df1,aes(iteration,value))+geom_line(aes(color=proportion))  +
    ylab("proportion") + theme(legend.position="none")
  df2<-melt(df[,c(1,4,5)],id.vars="iteration",variable="mean")
  graf2 <- ggplot(df2,aes(iteration,value))+geom_line(aes(color=mean))  +
    ylab("mean") +  theme(legend.position="none")
  df3<-melt(df[,c(1,6,7)],id.vars="iteration",variable="standard.deviation")
  graf3 <- ggplot(df3,aes(iteration,value))+geom_line(aes(color=standard.deviation))  +
    ylab("standard.deviation") +  theme(legend.position="none")
  graf4 <- ggplot(df,aes(iteration,deviance))+geom_line()
  grid.arrange(graf1,graf2,graf3,graf4,nrow=2, top=title)
}
graphConvergence(df.em, title="EM")
```



Ejemplo donde se calculan los parámetros de la mezcla del ejemplo presentado al principio ($pesos = (70, 30), \mu = (0, 4), \sigma = (1, 1)$). Los valores iniciales dados son $\pi_0 = (.5, .5), \mu = (1, 2), \sigma = (.5, 2)$

El algoritmo EM estocástico

En el paso E del algoritmo EM, cada vector z_j se reemplaza por su esperanza condicional dados los datos observados: $z_j^{(k)} = (z_{1j}^{(k)}, \ldots, z_{gj}^{(k)})^T$, donde $z_{ij}^{(k)}$ se define como en la ecuación (**??**) y es la probabilidad posterior actual de que la j-ésima observación venga del componente i-ésimo de la mezcla ($i = 1, \ldots, g; j = 1, \ldots, n$). Existen varias versiones del algoritmo EM estocático y se verán dos de ellas en este documento.

El algoritmo EM estocástico de Monte Carlo (MCEM)

Para este caso, la probabilidad posterior actual se usa en un paso estocástico E, donde se obtiene una sola muestra[3] de la distribución condicional actual de z dados los datos observados y. Por la independencia de los datos completos, esto se lleva a cabo obteniendo una muestra para cada j con $j = 1, \ldots, n$; es decir, se obtiene una muestra $z_j^{(1_k)}$ de una distribución multinomial con g categorías que tienen probabilidad $z_{ij}^{(k)}$.

El paso M consiste en encontrar el EMV de $\mathbf{\Psi}$ como si los $y_1, \ldots, y_n$ fueran clasificados de manera determinista de acuerdo a $z_1^{(1_k)}, \ldots, z_n^{(1_k)}$.

En resumen, el algoritmo estocástico empieza con probabilidades multinomiales arbitrarias $\tau_{ij}^{(0)}$, con $i = 1, \ldots, g$, para cada observación j. Luego, si $\mathbf{\Psi}^{(k)}$ es el valor de $\mathbf{\Psi}$ en la iteración k, se llevan a cabo los siguientes pasos:

Paso estocástico E.

[3]Se pueden obtener varias, y de hecho, en el algoritmo programado para este documento, se tiene esta opción.

Para cada j, se obtiene una muestra[4] de la distribución multinomial con probabilidades $\tau_{ij}^{(k)}$ con $i = 1, \ldots, g$, las cuales pueden ser arbitrarias cuando $k = 0$ (por simplicidad, se pueden tomar todas iguales) o están dadas por el paso M previo cuando $k > 0$. En la k-ésima iteración, se define $z_j^{(1_k)} = (z_1^{(1_k)}, \ldots, z_n^{(1_k)})$ la muestra de la j-ésima observación y_j. Esto resulta en una partición de las n observaciones $y_1, \ldots, y_n$ con respecto a las g componentes de la mezcla, porque para cada j, exactamente una de estas $z_{ij}^{(1_k)}$ es 1 y las otras 0.

Paso M.

Calcular

$$\pi_i^{(k+1)} = \sum_{j=1}^{n} \frac{1}{n} z_{ij}^{(1_k)}, \tag{21}$$

con $i = 1, \ldots, g$.

Si las f_i son normales de dimensión p, $\mu_i^{(k+1)}$ y $\Sigma_i^{(k+1)}$ están dadas, respectivamente, por las ecuaciones (**??**) y (**??**). Las probabilidades posteriores para el paso E estocástico en la siguiente iteración se actualizan de la siguiente forma:

$$\tau_{ij}^{(k+1)} = \tau_i(y_j; \mathbf{\Psi}^{(k+1)}) \tag{22}$$

$$= \frac{\pi_i^{(k+1)} f_i(y_j; \theta^{(k+1)})}{f(y_j; \mathbf{\Psi}^{(k+1)})} \tag{23}$$

para $i = 1, \ldots, g$.

Se presenta el código en R del paso estocástico para el caso de una mezcla de distribuciones normales escalares:

```
pasoS<-function(x,teta,M)
{
  n<-length(x)
  G<-length(teta$p)
  Z<-array(NA,c(n,G,M))
  tau<-calculotau(x,teta)
  for (i in 1:n)
  {
    Z[i,,]<-rmultinom(n=M,size=1,prob=tau[i,])
  }
  if (M==1) Z <- Z[,,1]
  return(Z)
}
```

El algoritmo EM de aproximación estocástica (SAEM)

Como se observó en la sección anterior, el algoritmo de EM de Monte Carlo reemplaza el paso E por una aproximación de Monte Carlo basada en un gran número de simulaciones independientes de los parámetros individuales no observados z. En el algoritmo de EM de aproximación estocástica (**SAEM**) se reemplaza el paso E por una aproximación estocástica basada en una sola simulación de z. Este algoritmo requiere de un valor inicial θ_0. En la k-ésima iteración se tienen tres pasos: un paso de simulación que llamaremos paso S, un paso de aproximación estocástica y finalmente el paso M de maximización.

Paso S.

Se obtiene una muestra $z_i^{(k)}$ de una distribución condicional $p(z_i|y_i; \Psi^{(k-1)})$.

[4]Éste es un caso particular del algoritmo EM de Monte Carlo, en el cual, se obtienen M muestras y la función Q se aproxima por medio de la suma $Q(\mathbf{\Psi}, \mathbf{\Psi}^{(k)}) = \frac{1}{M} \sum_{m=1}^{M} \log p(\mathbf{\Psi}|z^{(m_k)}; \boldsymbol{y})$.

Paso SA.

Se actualiza $Q_{k-1}(\Psi)$ de acuerdo a

$$Q_k(\Psi) = Q_{k-1}(\Psi) + \gamma_k(\log p(z_i|y_i;\Psi) - Q_{k-1}(\Psi)),$$

donde $\{\gamma_k\}_{k=1}^{\infty}$ es una sucesión decreciente tal que $\gamma_k \in \mathbb{R}$ con $\gamma_1 = 1$.

Paso M

Se actualiza $\mathbf{\Psi}^{(k+1)}$ igual que antes.

Se presenta el código en R para el caso del paso de aproximación estocástica.

```
pasoSA<-function(x,Z,s.old,gamma)
{
  S<-estadistica(x,Z)
  s11<-s.old$s1+gamma*(S$s1-s.old$s1)
  s12<-s.old$s2+gamma*(S$s2-s.old$s2)
  s13<-s.old$s3+gamma*(S$s3-s.old$s3)
  s.new<-list(s1=s11,s2=s12,s3=s13)
  return(s.new)
}
```

Se tiene entonces el algoritmo SAEM para la mezcla presentada para el algoritmo determinista, donde $\{\gamma_k\}_{k=1}^{\infty} = \{1\}_{k=1}^{\infty}$ (es decir, éste es un algoritmo MCEM puro) y $M = 1$.

```
mixt1.saem <- function(x, teta0, K, K1, M)
{
  G<-length(mu)
  col.names <- c("iteration", paste0("p",1:G), paste0("mu",1:G), paste0("sigma",1:G))

  K2 <- K - K1
  gamma<-c(rep(1,K1),1/(1:K2))

  teta.est <- matrix(NA,K+1,3*G+1)
  teta.est[1,] <- c(0, teta0$p, teta0$mu, teta0$sigma)

  teta<-teta0
  s<-s0
  for (k in 1:K)
  {
    Z<-pasoS(x,teta,M)
    s<-pasoSA(x,Z,s,gamma[k])
    teta<-pasoM(s,n)
    teta.est[k+1,] <- c(k, teta$p, teta$mu, teta$sigma)
  }
  df.saem <- as.data.frame(teta.est)
  names(df.saem) <- col.names
  return(df.saem)
}

K <- 250
K1 <- 250
M <- 1

##  SAEM
df.saem <- mixt1.saem(x, teta0, K, K1, M)
```

```
df.saem <- logLikelihood(x,df.saem)
print(round(df.saem[nrow(df.saem),],3))
```

```
##     iteration   p1   p2   mu1   mu2 sigma1 sigma2 deviance
## 251       250 0.71 0.29 0.011 4.236  1.053  0.814  788.594
```

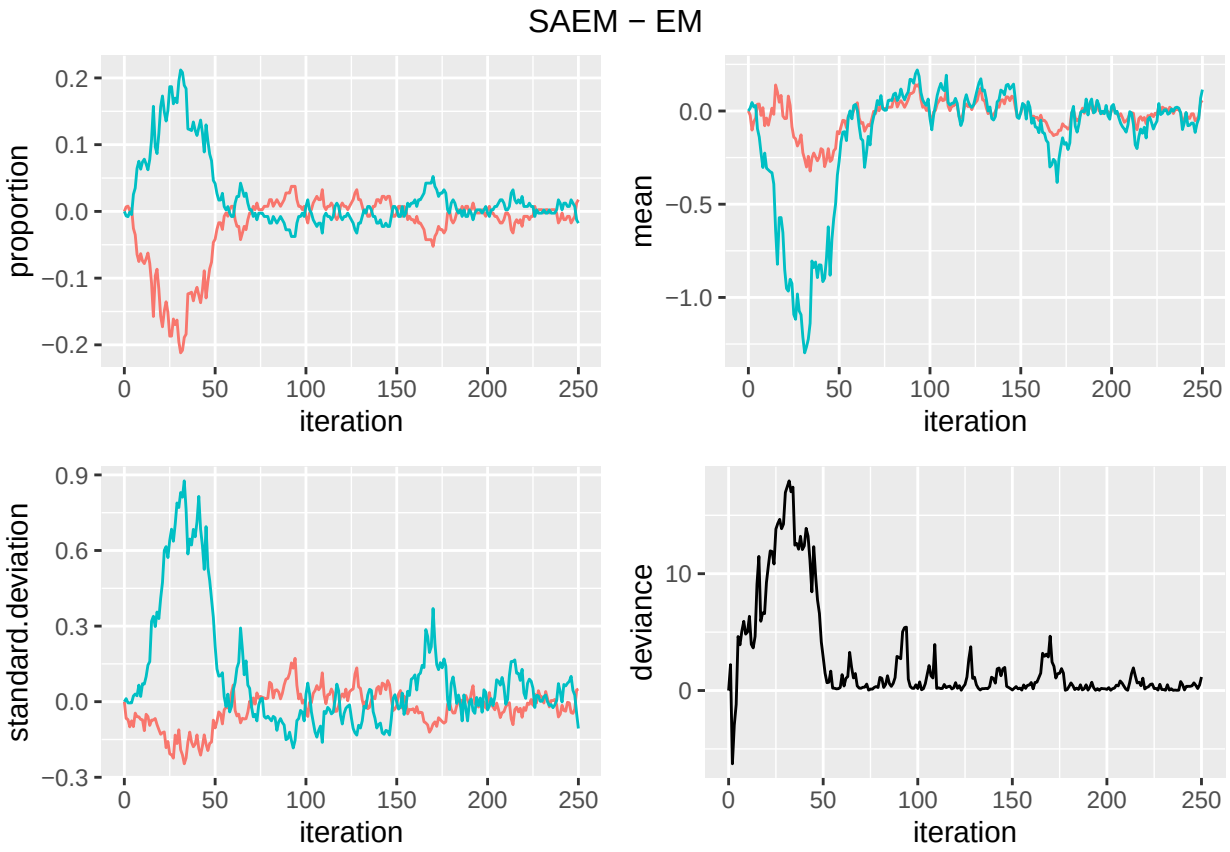
```
graphConvergence(df.saem, title="SAEM")
```

Haciendo la diferencia entre el método SAEM y EM se tiene:

```
## SAEM - EM
df.diff <- df.saem - df.em
df.diff$iteration <- df.em$iteration
print(round(df.diff[nrow(df.saem),],3))
```

```
##     iteration    p1     p2   mu1   mu2 sigma1 sigma2 deviance
## 251       250 0.018 -0.018 0.057 0.115   0.05 -0.106    1.159
```

```
graphConvergence(df.diff, title="SAEM - EM")
```

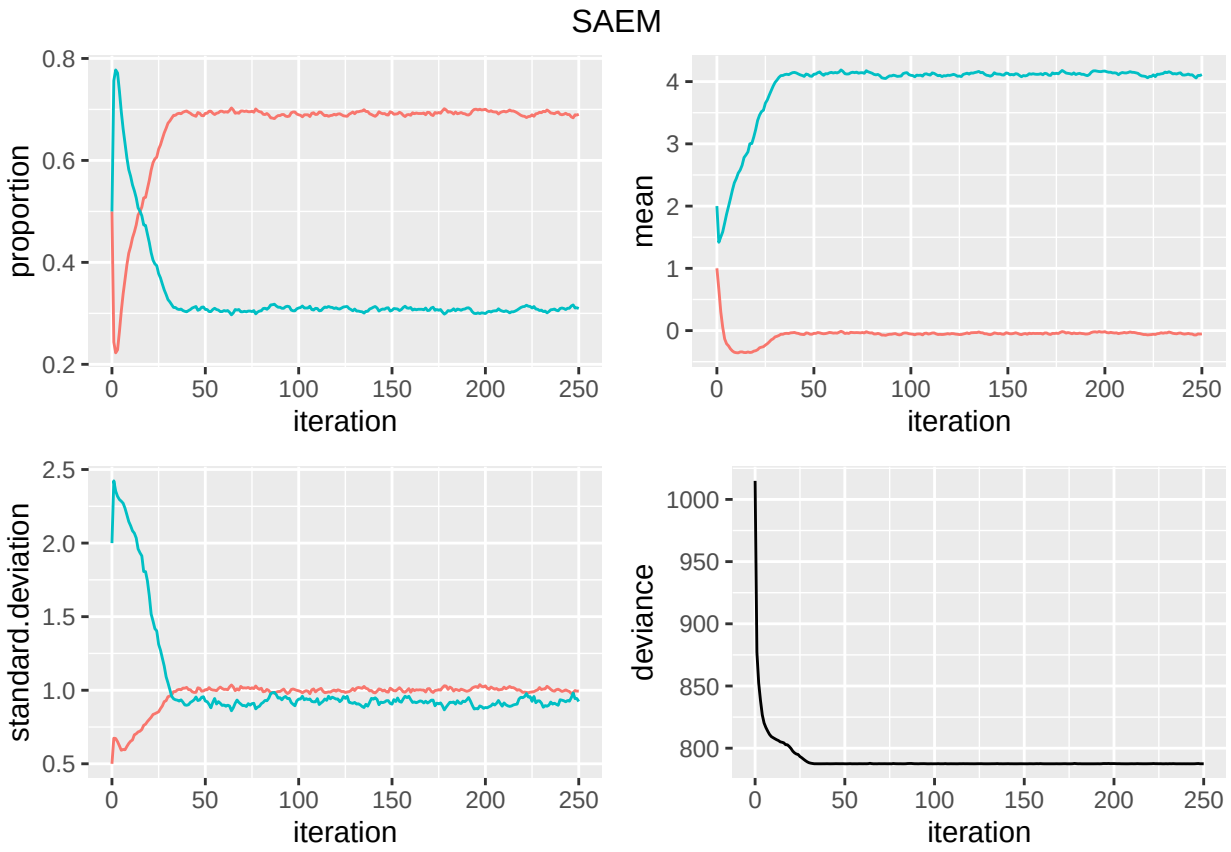


Si ahora $M = 10$, se puede ver que los valores del algortimo SAEM, el cual sigue siendo un algoritmo MCEM puro, se aproximan a los del algoritmo EM (las gráficas de SAEM se suavizan):

```
K <- 250
M <- 10

##  EM
df.em <- mixt1.em(x, teta0, K)
df.em <- logLikelihood(x,df.em)

K <- 250
K1 <- 250
M <- 10

##  SAEM
df.saem <- mixt1.saem(x, teta0, K, K1, M)
df.saem <- logLikelihood(x,df.saem)
print(round(df.saem[nrow(df.saem),],3))
```

```
##     iteration    p1   p2    mu1   mu2 sigma1 sigma2 deviance
## 251       250 0.691 0.31 -0.053 4.112  0.994  0.923  787.453
```

```
graphConvergence(df.saem, title="SAEM")
```

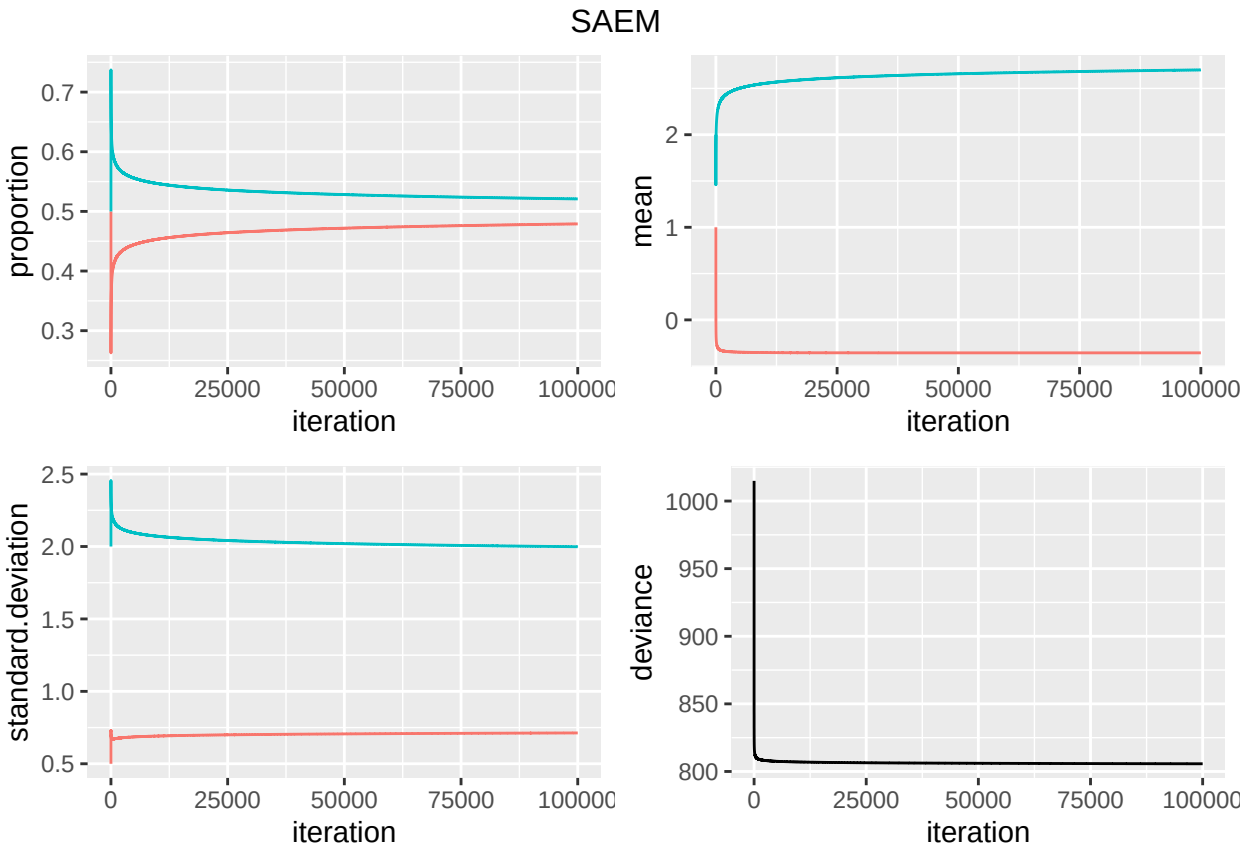


Si ahora se toma la sucesión decreciente $\{\gamma_k\}_{k=1}^{\infty} = \{\frac{1}{k}\}_{k=1}^{\infty}$, la gráfica de los valores del algoritmo SAEM se suaviza aún más.

```
K <- 250
M <- 10

##  EM
df.em <- mixt1.em(x, teta0, K)
df.em <- logLikelihood(x,df.em)

K <- 100000
K1 <- 0
M <- 10

##  SAEM
df.saem <- mixt1.saem(x, teta0, K, K1, M)
df.saem <- logLikelihood(x,df.saem)
print(round(df.saem[nrow(df.saem),],3))
```

```
##        iteration    p1    p2    mu1 mu2 sigma1 sigma2 deviance
## 100001     1e+05 0.479 0.521 -0.356 2.7  0.713  1.999  805.696
```

```
graphConvergence(df.saem, title="SAEM")
```



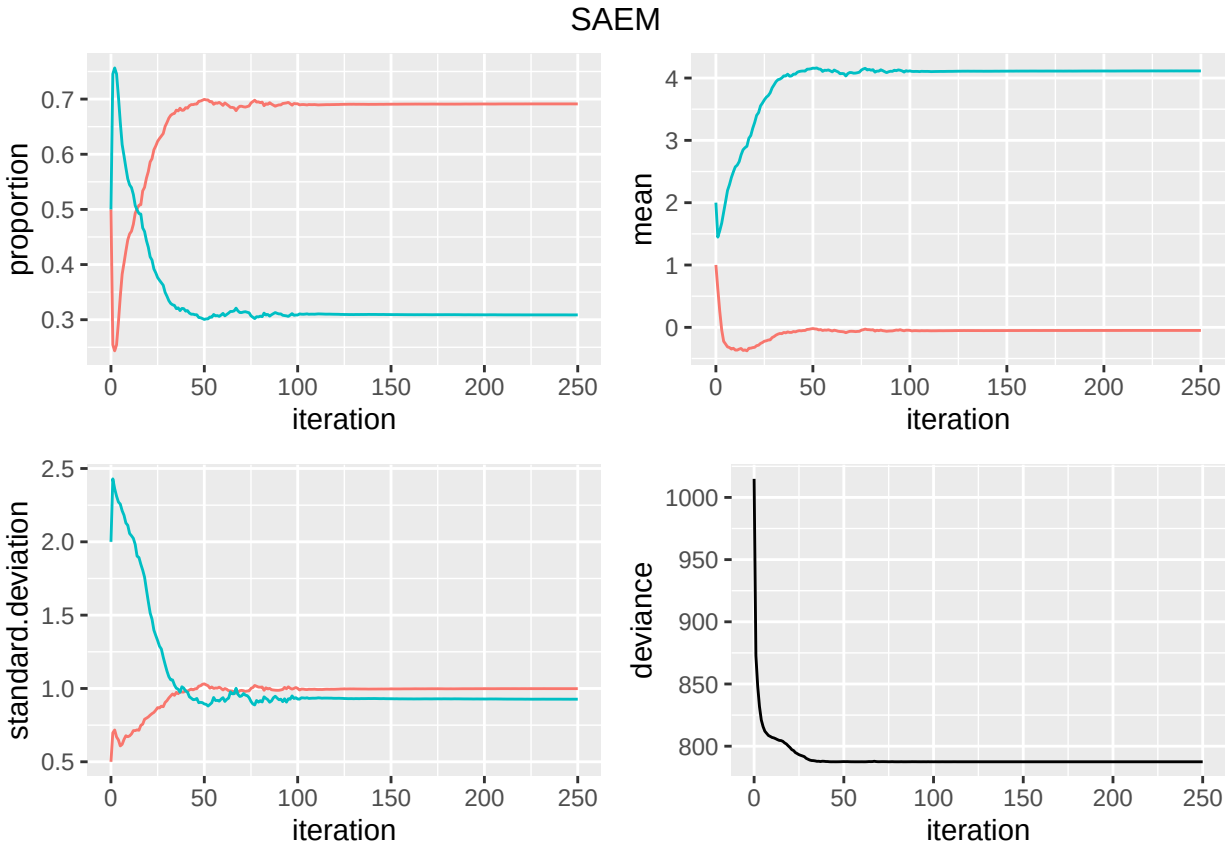
Lo cierto es que los valores convergen muy lentamente, además de que no convergen a los valores del algoritmo EM. En este caso entonces, se puede formar una sucesión para tener una combinación de los algoritmos MCEM puro y SAEM, para poder encontrar una mejor aproximación al valor verdadero de los parámetros. Es decir, los primeros k_1 valores de la sucesión podrían ser todos 1 y a partir del valor del valor $k_1 + 1$ podría tenerse propiamente una sucesión decreciente de números reales.

En el siguiente ejemplo se hace uso de ambos de ambos algortimos estocásticos con k_1=100 y $\{\gamma_k\}_{k=1}^{\infty} = \{\frac{1}{k}\}_{k=101}^{\infty}$.

```
K <- 250
K1 <- 100
M <- 10

##  SAEM
df.saem <- mixt1.saem(x, teta0, K, K1, M)
df.saem <- logLikelihood(x,df.saem)
print(round(df.saem[nrow(df.saem),],3))
```

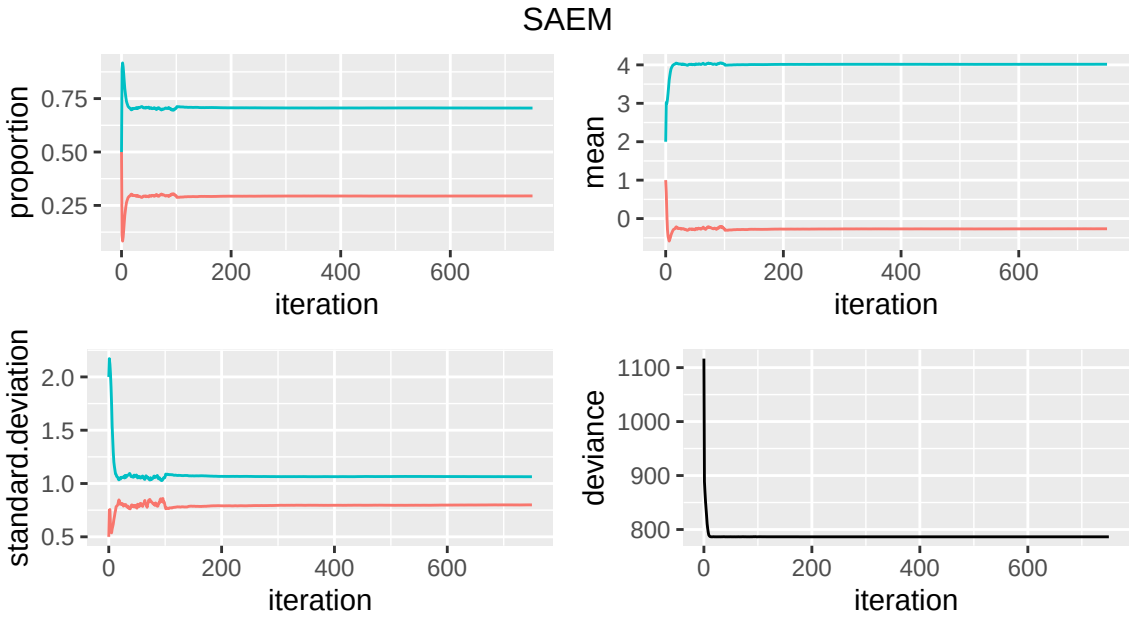
```
##     iteration    p1    p2    mu1   mu2 sigma1 sigma2 deviance
## 251       250 0.691 0.309 -0.049 4.114  0.999  0.927  787.439
```

```
graphConvergence(df.saem, title="SAEM")
```

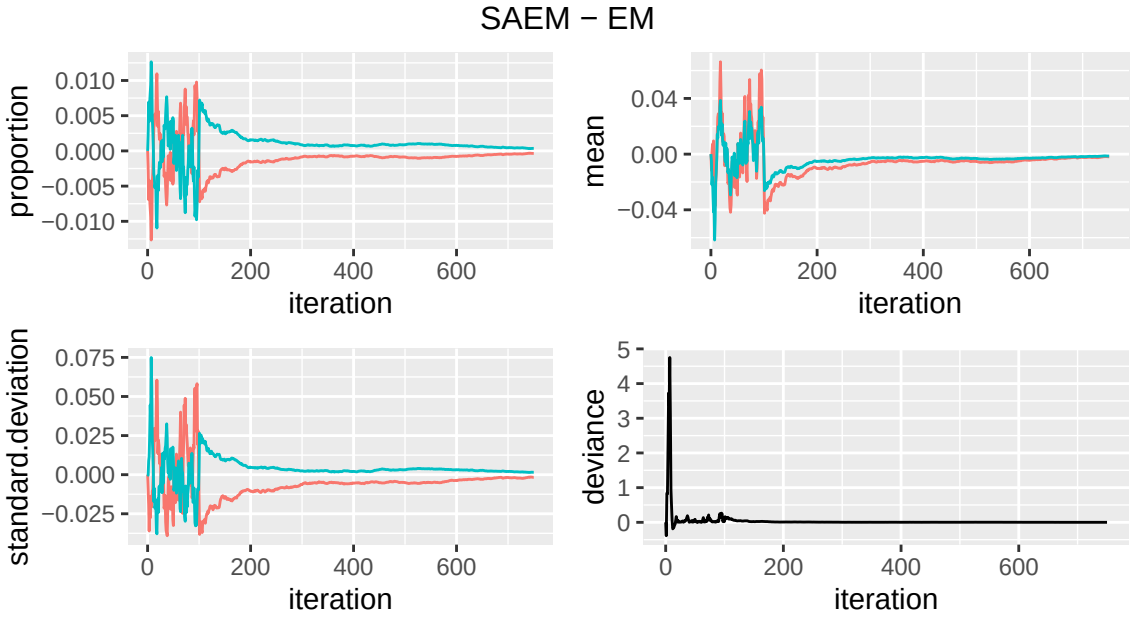


Por último, se puede modificar el algoritmo SAEM para que en vez de tomar la sucesión $\{\gamma_k\}_{k=1}^{\infty}$ se tome la sucesión $\{\gamma_k^{\alpha}\}_{k=1}^{\infty}$, donde $0.5 < \alpha < 1$. Con $\gamma = .8$ se observa que la aproximación a los valores de los parámetros es aún mejor, además de que las oscilaciones se vuelven más pequeñas a medida que se tienen más iteraciones.

```
##     iteration    p1    p2    mu1   mu2 sigma1 sigma2 deviance
## 751       750 0.294 0.706 -0.266 4.016    0.8  1.064  786.444
```



```
##     iteration p1 p2    mu1    mu2 sigma1 sigma2 deviance
## 751       750  0  0 -0.002 -0.001 -0.002  0.002        0
```



References

[1] Lavielle, M. (2014). Mixed effects models for the population approach: models, tasks, methods and tools. CRC press.

[2] McLachlan, G. J., & Peel, D. (2000). Finite Mixture Models. Wiley series in probability and statistics.

[3] McLachlan, G. J., & Krishnan, T. (1997). The EM algorithm and extensions. Wiley series in probability and statistics.