pdflatex, xelatex, and lualatex

HW1

EM and Regression. For $X = \{(Z_i, Y_i) : i = 1, ..., n\}$, consider the model

$Y_i = \beta_1 + \beta_2 Z_i + \varepsilon_i$

where $\varepsilon_1, ..., \varepsilon_n$ are i.i.d. $N(0, \sigma^2)$, $Z_1, ..., Z_n$ are i.i.d. $N(\mu_1, \sigma_1^2)$ and independent of $\varepsilon_1, ..., \varepsilon_n$. Suppose that for $1 \le i \le m$ we observe both Z_i and Y_i and for $m + 1 \le i \le n$, we observe only Y_i. Complete the E- and M-steps of the EM algorithm for estimating $(\mu_1, \beta_1, \sigma_1^2, \sigma^2, \beta_2)$.

HW2. Due Jan. 29.

i) P76: Genetic Linkage (continued) (Missing Information Principle). Verify the result.

```
y <- c(125,18,20,34)
EM_Rao<- function(y, theta= 0.5, conv = 0.0001)
{
## This is an example of E-M algorithm for multinomial distribution
## "y" is the observed data represented by a vector
## use "theta"=0.5 is the starting value
## "conv" is the convergence criterion number
est<-NULL
repeat {
prob <- theta/(theta + 2)
num <- y[4] + y[1] * prob
den <- sum(y[2:4]) + y[ 1] * prob
thetahat <- num/den # This is an updated theta
est <- c( est, thetahat)
if(abs(thetahat - theta)< conv)
break
else theta <- thetahat
}
return(est)
}
EM_Rao(y, theta= 0.5, conv = 0.0001)
## [1] 0.60824742 0.62432105 0.62648888 0.62677732 0.62681563
```

For this model,

$$\frac{\partial \log p(\theta|Y,Z)}{\partial\theta} = \frac{x_2+x_5}{\theta} - \frac{x_3+x_4}{1-\theta}$$

while

$$\left.\frac{-\partial^2 Q(\theta,\hat{\theta})}{\partial\theta^2}\right|_{\hat{\theta}} = \frac{E(x_2|\hat{\theta},Y)+x_5}{\hat{\theta}^2} + \frac{x_3+x_4}{(1-\hat{\theta})^2}$$

For the data set listed in the sample of Section 4.1

$$-\left.\frac{\partial^2 Q(\theta,\hat{\theta})}{\partial\theta^2}\right|_{\hat{\theta}} = \frac{29.83+34}{0.6268^2} + \frac{38}{(1-0.6268)^2} = 435.3$$

Now

$$Var\left(\left.\frac{\partial \log p(\theta|Y,Z)}{\partial\theta}\right|_{\hat{\theta}}\right) = \frac{Var(x_2|\hat{\theta})}{\hat{\theta}^2} = \frac{125\left(\frac{\hat{\theta}}{2+\hat{\theta}}\right)\left(\frac{2}{2+\hat{\theta}}\right)}{\hat{\theta}^2} = 22.71/0.6268^2 = 57.8$$

Hence, it follows that

$$\left.\frac{-\partial^2 \log p(\theta|Y)}{\partial\theta^2}\right|_{\hat{\theta}} = 435.3 - 57.8 = 377.5$$

and the standard error of $\hat{\theta}$ is equal to $\sqrt{(1/377.5)} = 0.05$.

```
thetahat <- 0.6268156
## Missing Information Principle
## Observed Information = Complete Information - Missing Information
prob <- thetahat/(thetahat + 2)
Q_dd <- (y[1]*prob+y[4])/thetahat^2+(y[2]+y[3])/(1-thetahat)^2
H_dd <- y[1]*prob*(1-prob)/thetahat^2
(se <- sqrt(1/(Q_dd-H_dd)))
## [1] 0.051467821
```

The result is coordinated with the textbook.

ii) P78: Genetic Linkage (continued) (Monte Carlo Variance estimation). Verify the result by writing your own code.

Method I:

$$Var\left(\frac{\partial \log p(\theta|Y,z_j)}{\partial\theta}\bigg|_{\hat\theta}\right) = \frac{Var(x_{2j}|\hat\theta)}{\hat\theta^2} \approx 57.87179,\ j=1,..,10^6$$

Method II:

$$\frac{1}{10^6}\sum_{j=1}^{10^6}\left(\frac{\partial \log p(\theta|Y,Z)}{\partial\theta}\bigg|_{\hat\theta}\right)^2 = \frac{1}{10^6}\sum_{j=1}^{10^6}\left(\frac{x_{2j}+x_5}{\theta}-\frac{x_3+x_4}{1-\theta}\right)^2 \approx 57.87183$$

```
## Monte Carlo Variance Estimation
set.seed(123)
S <- 1e+6
z_draw <- rbinom(S, 125, prob)
var(z_draw)/thetahat^2 # Method I
## [1] 57.871789
mean(((z_draw+y[4])/thetahat-(y[2]+y[3])/(1-thetahat))^2) # Method II
## [1] 57.871828
```

Both of the results are coordinated with the textbook.

iii) P81: Genetic Linkage (continued) (Monte Carlo implementation of E-step). Verify the result by writing your own code.

```
MCEM_Rao<- function(y,S,theta,conv)
{
## This is an example of Monte Carlo implementation of E-step
## "conv" is the convergence criterion number
est<-NULL
repeat {
prob <- theta/(theta + 2)
set.seed(121)
z_draw <- rbinom(S, 125, prob)
num <- y[4] + mean(z_draw)
den <- sum(y[2:4]) + mean(z_draw)
thetahat <- num/den # This is an updated theta
```

```r
est <- c( est, thetahat)
if(abs(thetahat - theta)< conv)
break
else theta <- thetahat
}
return(est)
}
MCEM_Rao(y,1e+1, 0.4, conv = 1e-4) ## use "theta"=0.4 as the starting value
## [1] 0.59227468 0.62338949 0.62781587 0.62818004 0.62818004
MCEM_Rao(y,1e+2, 0.4, conv = 1e-5)
## [1] 0.59011973 0.62102324 0.62524655 0.62591061 0.62598425 0.62598425
MCEM_Rao(y,1e+3, 0.4, conv = 1e-6) ## try smaller convergence criterion number
## [1] 0.59106807 0.62227767 0.62651360 0.62715490 0.62723536 0.62724633 0.62724999
## [8] 0.62724999
MCEM_Rao(y,1e+4, 0.4, conv = 1e-4)
## [1] 0.59090650 0.62213794 0.62640528 0.62697885 0.62705244
MCEM_Rao(y,1e+5, 0.4, conv = 1e-4) ## try lager sample size
## [1] 0.59066346 0.62188430 0.62615866 0.62673798 0.62681393
```

The starting value is $\theta = 0.4$. All the setting of sample size and will converge

The results show that the sample size drawed from Binomial distribution decides the outcomes.

If the size is large enough ($S = 100000$), the value of $\theta \approx 0.6268$ after 5 iterations, which is same with textbook.

The smaller convergence criterion number will add more times of iteration but it doesn't give more precise results.

HW2 Extra

```r
x <- c(1,1,-1,-1,2,2,-2,-2)
y <- c(1,-1,1,-1,2,2,-2,-2)

EM_Bivariate_Normal<- function(x,y,n_xy,n_x,n_y,rho0,crit,itera)
{
## "x","y" is the data
## "theta" is the parameter vector: sigma_sq_x, sigma_sq_y, and rho.
## "thetastar" is the current parameter estimate.
## "n_xy" is the data size with both x and y
## "n_x"  is the data size with only x
## "n_y"  is the data size with only y
## "itera" is the upper limit of iterations.
############################################################
s <- 0 # iteration counter
n <- n_xy+n_x+n_y
i <- 1:n_xy
j <- (n_xy+1):(n_xy+n_x)
k <- (n_xy+1):(n_xy+n_y)

sigma_x <- sqrt(mean(x^2)) # initial parameter values
sigma_y <- sqrt(mean(y^2))
thetastar <- c(sigma_x,sigma_y,rho0)

repeat {
# Computing conditional mean
x_star <- thetastar[1]/thetastar[2]*thetastar[3]*y[k]
y_star <- thetastar[2]/thetastar[1]*thetastar[3]*x[j]
# set unconditional mean = conditional
sigma_x <- sqrt(mean(c(x^2,x_star^2+rep(thetastar[1]^2*(1-thetastar[3]^2),n_y))))
sigma_y <- sqrt(mean(c(y^2,y_star^2+rep(thetastar[2]^2*(1-thetastar[3]^2),n_x))))
rho <- mean(c(x[i]*y[i],x[j]*y_star,x_star*y[k]))/(sigma_x*sigma_y)
theta <- c(sigma_x,sigma_y,rho)
s <- s +1
if((sqrt(sum((thetastar-theta)^2)) < crit)| (s > itera))
break
thetastar <- theta
}
return(c(s,theta))
}
EM_Bivariate_Normal(x,y, n_xy=4,n_x=4,n_y=4,rho0=0,crit=1e-4,itera=10)
## [1] 1.0000000 1.5811388 1.5811388 0.0000000
EM_Bivariate_Normal(x,y, n_xy=4,n_x=4,n_y=4,rho0=0.25,crit=1e-6,itera=100)
## [1] 93.00000000  1.63299166  1.63299166  0.49999301
EM_Bivariate_Normal(x,y, n_xy=4,n_x=4,n_y=4,rho0=-0.8,crit=1e-6,itera=100)
## [1] 82.00000000  1.63299464  1.63299464 -0.50000688
```

HW3. Due Feb. 12.

i) Apply Louis' method to find an estimate of Fisher information matrix in the Schmee and Hahn (1979). Your assignment is to write a program to estimate $\beta 0, \beta 1$, and σ^2 by the EM algorithm. (We did this last term.) Then, you should find point-wise confidence intervals for the parameters by evaluating at the estimate of parameters provided by the EM algorithm $-\frac{\partial^2 \log p(\beta_0,\beta_1,\sigma^2|y)}{\partial^2(\beta_0,\beta_1,\sigma^2)}$,where y is the observed data . This evaluation should be done by Louis' method, as explained in class. That is, you have to simulate the right-censored values from the fitted normal distribution, conditional on being larger than ci.

The estimate of $\beta 0, \beta 1$, and σ by the EM algorithm. (detail omitted)

```
## [1] -6.01903378  4.31113761  0.25915984
```

- Pre-calculation

$\mu = \beta_0 + \beta_1 x$; $\vec{\theta} = (\beta_0, \beta_1, \sigma)$
$(y_j, x_j),\ j = 1, ..m$ is the observed data; $u = \frac{y-\mu}{\sigma}$.
$(c_i, x_i)\ i = m+1, ..n$ is the censored data; $v = \frac{c-\mu}{\sigma}$.
z_i is the expected values. $w = \frac{z-\mu}{\sigma}$

$$\phi(u;\theta) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{y-\mu}{\sigma})^2} = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{y-\beta_0-\beta_1 x}{\sigma})^2}$$

$$\frac{\partial \phi(u)}{\partial(\theta)} = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{y-\mu}{\sigma})^2} (\frac{y-\mu}{\sigma}) \begin{bmatrix} 1/\sigma \\ x/\sigma \\ u/\sigma \end{bmatrix} = \phi(u)\frac{u}{\sigma} \begin{bmatrix} 1 \\ x \\ u \end{bmatrix}$$

$$\frac{\partial}{\partial(\theta)}(1-\Phi(v)) = -\phi(v) \begin{bmatrix} -1/\sigma \\ -x/\sigma \\ -v/\sigma \end{bmatrix} = \phi(v)\frac{1}{\sigma} \begin{bmatrix} 1 \\ x \\ v \end{bmatrix}$$

$$\frac{\partial}{\partial\theta} H(v) = \frac{\partial}{\partial\theta}\frac{\phi(v)}{1-\Phi(v)} = \frac{(1-\Phi(v))\phi(v)\frac{v}{\sigma}\begin{bmatrix} 1 \\ x \\ v \end{bmatrix} - \phi(v)\phi(v)\frac{1}{\sigma}\begin{bmatrix} 1 \\ x \\ v \end{bmatrix}}{(1-\Phi(v))^2} = H(v)[v - H(v)]\frac{1}{\sigma}\begin{bmatrix} 1 \\ x \\ v \end{bmatrix}$$

$$\frac{\partial}{\partial\theta}(\frac{1}{\sigma}\begin{bmatrix} 1 \\ x \\ v \end{bmatrix}) = \frac{\partial}{\partial\theta}(\begin{bmatrix} \frac{1}{\sigma} \\ \frac{x}{\sigma} \\ \frac{c-\mu}{\sigma^2} \end{bmatrix}) = \frac{-1}{\sigma^2}\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & x \\ 1 & x & 2v \end{bmatrix}$$

$$\frac{\partial}{\partial\theta}(\frac{u}{\sigma}\begin{bmatrix} 1 \\ x \\ u \end{bmatrix}) = \frac{\partial}{\partial\theta}(\begin{bmatrix} \frac{u}{\sigma} \\ \frac{xu}{\sigma} \\ \frac{(y-\mu)^2}{\sigma^3} \end{bmatrix}) = \frac{-1}{\sigma^2}\begin{bmatrix} 1 & x & 2u \\ x & x^2 & 2xu \\ 2u & 2xu & 3u^2 \end{bmatrix}$$

$$
\begin{aligned}
E[\varepsilon_i|Z_i > c_i, \vec{\theta}^\star] &== \frac{\phi(\frac{c_i-\mu_i^\star}{\sigma^\star})}{1-\Phi(\frac{c_i-\mu_i^\star}{\sigma^\star})} = H(\frac{c_i-\mu_i^\star}{\sigma^\star}) & \square \\
E[\varepsilon_i^2|Z_i > c_i, \vec{\theta}^\star] &= \int_{\frac{c_i-\mu_i^\star}{\sigma^\star}}^{\infty} \frac{\varepsilon_i^2\phi(\varepsilon_i)}{1-\Phi(\frac{c_i-\mu_i^\star}{\sigma^\star})} d\varepsilon_i \\
&= \frac{1}{1-\Phi(\cdot)} \int_{\frac{c_i-\mu_i^\star}{\sigma^\star}}^{\infty} \varepsilon_i^2 \frac{1}{\sqrt{2\pi}} e^{\frac{-\varepsilon_i^2}{2}} d\varepsilon_i \\
&= \frac{-1}{1-\Phi(\cdot)} \left\{ \left[\varepsilon_i \frac{1}{\sqrt{2\pi}} e^{\frac{-\varepsilon_i^2}{2}}\right]_{\frac{c_i-\mu_i^\star}{\sigma^\star}}^{\infty} - \int_{\frac{c_i-\mu_i^\star}{\sigma^\star}}^{\infty} \frac{1}{\sqrt{2\pi}} e^{\frac{-\varepsilon_i^2}{2}} d\varepsilon_i \right\} \\
&= \frac{-1}{1-\Phi(\cdot)} \left\{ -\frac{c_i-\mu_i^\star}{\sigma^\star} \frac{1}{\sqrt{2\pi}} e^{\frac{-(\frac{c_i-\mu_i^\star}{\sigma^\star})^2}{2}} - (1-\Phi(\cdot)) \right\} \\
&= \frac{c_i-\mu_i^\star}{\sigma^\star} H(\cdot) + 1 & \square
\end{aligned}
$$

$$
\begin{aligned}
E[Z_i|Z_i > c_i, \vec{\theta}^\star] &= E[\mu_i^\star + \sigma^\star\varepsilon_i|.] = \mu_i^\star + \sigma^\star E[\varepsilon_i|.] = \mu_i^\star + \sigma^\star H(\frac{c_i-\mu_i^\star}{\sigma^\star}) & \square \\
E[Z_i^2|Z_i > c_i, \vec{\theta}^\star] &= \mu_i^{\star 2} + 2\mu_i^\star\sigma^\star E[\varepsilon_i|.] + \sigma^{\star 2}E[\varepsilon_i^2|.] \\
&= \mu_i^{\star 2} + 2\mu_i^\star\sigma^\star H(\cdot) + \sigma^{\star 2}[\frac{c_i-\mu_i^\star}{\sigma^\star}H(\cdot)+1] & \square
\end{aligned}
$$

$$
E[w|z > c, \vec{\theta}^\star] = \frac{E[z|z > c, \vec{\theta}^\star] - \mu}{\sigma} = \frac{\mu + \sigma H(v) - \mu}{\sigma} = H(v)
$$

$$
E[w^2|z > c, \vec{\theta}^\star] = \frac{1}{\sigma^2}(E[z^2|\cdot] - 2\mu E[z|\cdot] + \mu^2) = \frac{1}{\sigma^2}(\mu^2 + 2\mu\sigma H(\cdot) + \sigma^2[vH(\cdot)+1] - 2\mu[\mu + \sigma H(v)] + \mu^2) = vH(v) + 1
$$

- Complete Information:

$$\log(p(\beta_0,\beta_1,\sigma|y,z)) = -\frac{n}{2}\log(2\pi) - n\log(\sigma) - \frac{1}{2}\sum_{j=1}^{m}(\frac{y_j-\beta_0-\beta_1x_j}{\sigma})^2 - \frac{1}{2}\sum_{i=m+1}^{n}(\frac{z_i-\beta_0-\beta_1x_i}{\sigma})^2$$
$$= C - n\log(\sigma) + \sum_{j=1}^{m}\log\phi(u_j) + \sum_{i=m+1}^{n}\log\phi(w_i)$$

$$\frac{\partial}{\partial\theta}\log(p(\vec{\theta}|y,z)) = \begin{bmatrix}0\\0\\\frac{-n}{\sigma}\end{bmatrix} + \sum_{j=1}^{m}\frac{\phi(u_j)}{\phi(u_j)}\begin{bmatrix}u_j/\sigma\\x_ju_j/\sigma\\u_j^2/\sigma\end{bmatrix} + \sum_{i=m+1}^{n}\frac{\phi(w_i)}{\phi(w_i)}\begin{bmatrix}w_i/\sigma\\x_iw_i/\sigma\\w_i^2/\sigma\end{bmatrix} = \frac{1}{\sigma}\begin{bmatrix}\sum_1^m u_j + \sum_{m+1}^n w_i\\\sum_1^m u_jx_j + \sum_{m+1}^n w_ix_i\\\sum_1^m u_j^2 + \sum_{m+1}^n w_i^2 - n\end{bmatrix}$$

$$E[\frac{\partial}{\partial\theta}\log(p(\vec{\theta}|y,z))|\cdot] = \frac{1}{\sigma}\begin{bmatrix}\sum_1^m u_j + \sum_{m+1}^n H(v_i)\\\sum_1^m u_jx_j + \sum_{m+1}^n H(v_i)x_i\\\sum_1^m u_j^2 + \sum_{m+1}^n [v_iH(v_i)+1] - n\end{bmatrix}$$

$$\frac{\partial^2}{\partial\theta^2}\log(p(\vec{\theta}|y,z)) = \begin{bmatrix}0&0&0\\0&0&0\\0&0&\frac{n}{\sigma^2}\end{bmatrix} - \frac{1}{\sigma^2}\sum_{j=1}^{m}\begin{bmatrix}1&x_j&2u_j\\x_j&x_j^2&2x_ju_j\\2u_j&2x_ju_j&3u_j^2\end{bmatrix} - \frac{1}{\sigma^2}\sum_{i=m+1}^{n}\begin{bmatrix}1&x_i&2w_i\\x_i&x_i^2&2x_iw_i\\2w_i&2x_iw_i&3w_i^2\end{bmatrix}$$

$$= -\frac{1}{\sigma^2}\begin{bmatrix}n & \sum_1^n x_i & 2(\sum_1^m u_j + \sum_{m+1}^n w_i)\\\sum_1^n x_i & \sum_1^n x_i^2 & 2(\sum_1^m u_jx_j + \sum_{m+1}^n w_ix_i)\\2(\sum_1^m u_j + \sum_{m+1}^n w_i) & 2(\sum_1^m u_jx_j + \sum_{m+1}^n w_ix_i) & 3(\sum_1^m u_j^2 + \sum_{m+1}^n w_i^2) - n\end{bmatrix}$$

$$E[\frac{\partial^2}{\partial\theta^2}\log(p(\vec{\theta}|y,z))|\cdot] = -\frac{1}{\sigma^2}\begin{bmatrix}n & \sum_1^n x_i & 2(\sum_1^m u_j + \sum_{m+1}^n H(v_i))\\\sum_1^n x_i & \sum_1^n x_i^2 & 2(\sum_1^m u_jx_j + \sum_{m+1}^n H(v_i)x_i)\\2(\sum_1^m u_j + \sum_{m+1}^n H(v_i)) & 2(\sum_1^m u_jx_j + \sum_{m+1}^n H(v_i)x_i) & 3(\sum_1^m u_j^2 + \sum_{m+1}^n [v_iH(v_i)+1]) - n\end{bmatrix}$$

```r
Compy <- matrix(c(n, sum(D[,2])         , 2*(sum(u)+sum(H)),
          sum(D[,2]), sum(D[,2]^2)       , 2*(sum(u*exact[,2])+sum(H*censored[,2])),
            2*(sum(u)+sum(H)), 2*(sum(u*exact[,2])+sum(H*censored[,2])), 3*(sum(u^2)+sum(H*v+1))-n
                                     ),nrow = 3, ncol = 3,)/sigma^2  # Complete Information
```

- Missing Information:(By simulation method)

We can approximate

$$Var\left[\frac{\partial}{\partial\theta}\log(p(\vec{\theta}|y,z))\right] = \frac{1}{S}\sum_{j=1}^{S}\left(\frac{\partial}{\partial\theta}\log(p(\vec{\theta}|y,z))\Big|_{\hat{\theta}}\right)^2 = \frac{1}{S}\sum_{j=1}^{S}\frac{1}{\sigma^2}\begin{bmatrix}\sum_1^m u_j + \sum_{m+1}^n w_i\\\sum_1^m u_jx_j + \sum_{m+1}^n w_ix_i\\\sum_1^m u_j^2 + \sum_{m+1}^n w_i^2 - n\end{bmatrix}^2$$

where $w \sim$ Truncated Normal $(0,1)$ distribution given $z > c$.

```
set.seed(121) # MC simulation
S=1e+3
C1<-C2<-C3<-matrix(NA,nrow = S,ncol = n)
M <- matrix(NA,nrow = S,ncol = 3)
for (s in 1:S){
u_draw <- rnorm(n,mean=0,sd=1)
w_draw <- truncnorm::rtruncnorm(n-m,a=(censored[,1]-mu_i)/sigma,b=Inf,mean=0,sd=1)
C1[s,] <- u_draw
C2[s,] <- u_draw*exact[,2]
C3[s,] <- u_draw^2
M_1 <- sum(w_draw)
M_2 <- sum(w_draw*censored[,2])
M_3 <- sum(w_draw^2)
M[s,] <- c(M_1,M_2,M_3)
}
Compy_sim <- matrix(c(n,   sum(D[,2])        , 2*sum(colMeans(C1)),
               sum(D[,2]),   sum(D[,2]^2)      , 2*sum(colMeans(C2)),
     2*sum(colMeans(C1)), 2*sum(colMeans(C2)) , 3*sum(colMeans(C3))-n
          ),nrow = 3, ncol = 3)/sigma^2    # Complete Information
Miss <- var(M)/sigma^2 # Missing Information
#(t(M)%*%M)/S-c(mean(M[,1]),mean(M[,2]),mean(M[,3]))%*%t(c(mean(M[,1]),mean(M[,2]),mean(M[,3])))
```

```
I <- Compy-Miss # Apply the Louis' method
interval <- pnorm(0.975)*sqrt(diag(solve(I))) # Calculate Confidence Interval
CI <- matrix(c(theta-interval,theta+interval),3,2,dimnames =list(c("beta0","beta1","sigma"),c("CI-L","C
```

Table 1: Louis' Method: Var-Cov Matrix and Confidence intervals

	Beta0	Beta1	Sigma	CI-L	CI-U
Beta0	0.9402	-0.4335	-0.009532	-6.829	-5.209
Beta1	-0.4335	0.2006	0.004954	3.937	4.685
Sigma	-0.009532	0.004954	0.002127	0.2206	0.2977

We get the point-wise 95% confidence intervals for the parameters by Louis' method.

ii) Now compute the observed information matrix directly and obtain its inverse. Find out 95% confidence intervals for the parameters. Compare your results based on i) and ii).

Observed Information

$$\log(p(\vec{\theta}|y)) = C - m\log(\sigma) + \sum_{j=1}^{m} \log\phi(u_j) + \sum_{i=m+1}^{n} \log(1-\Phi(v_i))$$

$$\frac{\partial}{\partial\theta}\log(p(\vec{\theta}|y)) = \underbrace{\begin{bmatrix}0\\0\\\frac{-m}{\sigma}\end{bmatrix} + \sum_{j=1}^{m}\begin{bmatrix}u_j/\sigma\\x_j u_j/\sigma\\u_j^2/\sigma\end{bmatrix}}_{(U)} + \underbrace{\sum_{i=m+1}^{n} H(v_i)\frac{1}{\sigma}\begin{bmatrix}1\\x_i\\v_i\end{bmatrix}}_{(V)}$$

$$\frac{\partial}{\partial\theta}(U) = \begin{bmatrix}0&0&0\\0&0&0\\0&0&\frac{m}{\sigma^2}\end{bmatrix} - \frac{1}{\sigma^2}\sum_{j=1}^{m}\begin{bmatrix}1&x_j&2u_j\\x_j&x_j^2&2u_jx_j\\2u_j&2u_jx_j&3u_j^2\end{bmatrix} = -\frac{1}{\sigma^2}\sum_{j=1}^{m}\begin{bmatrix}1&x_j&2u_j\\x_j&x_j^2&2u_jx_j\\2u_j&2u_jx_j&3u_j^2-1\end{bmatrix}$$

$$\frac{\partial}{\partial\theta}(V)=\sum_{i=m+1}^{n}H(v_i)\frac{\partial}{\partial\theta}(\frac{1}{\sigma}\begin{bmatrix}1\\x_i\\v_i\end{bmatrix})+\sum_{i=m+1}^{n}\frac{1}{\sigma}\begin{bmatrix}1\\x_i\\v_i\end{bmatrix}\frac{\partial H(v_i)}{\partial\theta}$$

$$=\sum_{i=m+1}^{n}H(v_i)(\frac{-1}{\sigma^2})\begin{bmatrix}0&0&1\\0&0&x_i\\1&x_i&2v_i\end{bmatrix}+\sum_{i=m+1}^{n}\frac{1}{\sigma^2}H(v_i)[v_i-H(v_i)]\begin{bmatrix}1\\x_i\\v_i\end{bmatrix}\begin{bmatrix}1&x_i&v_i\end{bmatrix}$$

$$=\sum_{i=m+1}^{n}H(v_i)(\frac{-1}{\sigma^2})\begin{bmatrix}0&0&1\\0&0&x_i\\1&x_i&2v_i\end{bmatrix}+\sum_{i=m+1}^{n}\frac{1}{\sigma^2}H(v_i)[v_i-H(v_i)]\begin{bmatrix}1&x_i&v_i\\x_i&x_i^2&x_iv_i\\v_i&x_iv_i&v_i^2\end{bmatrix}$$

$$=\sum_{i=m+1}^{n}H\frac{1}{\sigma^2}\begin{bmatrix}v_i-H&x_i(v_i-H)&v_i(v_i-H)-1\\x_i(v_i-H)&x_i^2(v_i-H)&x_i[v_i(v_i-H)-1]\\v_i(v_i-H)-1&x_i[v_i(v_i-H)-1]&v_i[v_i(v_i-H)-2]\end{bmatrix}$$

$$\frac{\partial^2}{\partial\theta^2}\log(p(\vec{\theta}|y))=\frac{\partial}{\partial\theta}(U)+\frac{\partial}{\partial\theta}(V)$$

```r
I_u <- matrix(c(   m          , sum(exact[,2])     , 2*sum(u),
               sum(exact[,2]), sum(exact[,2]^2) , 2*sum(u*exact[,2]),
                2*sum(u)    , 2*sum(u*exact[,2]) , 3*sum(u^2)-m
),nrow = 3, ncol = 3)/(sigma^2) # Observed exact data
I_v <- matrix(c(sum((v-H)*H) , sum(H*(v-H)*censored[,2])       , sum(H*(v*(v-H)-1)),
   sum(censored[,2]*(v-H)*H) , sum(H*(v-H)*censored[,2]^2)     , sum(H*(v*(v-H)-1)*censored[,2]),
           sum(-H+v*(v-H)*H) , sum(H*(v*(v-H)-1)*censored[,2]) , sum(H*(v*(v-H)-2)*v)
                ),nrow = 3, ncol = 3)/(sigma^2) # Observed censored data
I_o <- I_u-I_v
```

```r
interval <- pnorm(0.975)*sqrt(diag(solve(I_o))) # Calculate Confidence Interval
CI_o <- matrix(c(theta-interval,theta+interval),3,2,dimnames =list(c("beta0","beta1","sigma"),c("CI-L",
```

Table 2: Direct Method: Var-Cov Matrix and Confidence intervals

	Beta0	Beta1	Sigma	CI-L	CI-U
Beta0	0.8962	-0.4126	-0.008243	-6.81	-5.228
Beta1	-0.4126	0.1906	0.004389	3.946	4.676
Sigma	-0.008243	0.004389	0.002241	0.2196	0.2987

which gives the 95% confidence intervals for $\beta 0, \beta 1$, and σ. Two methods give same result.

Detailed Steps

```r
exp<-expression(x^2+y^2+2*x*y-3*x+4*y+4)
dx<-D(exp,"x"); dx
dy<-D(exp,"y"); dy
dxy<-deriv(exp,c("x","y"));dxy;typeof(dxy)
x<-seq(-pi,pi,pi/4)
y<-pi
eval(dxy)
pp<-deriv(exp,c("x","y"),func=T)
pp(x,y)

integrate(f,0,1)

library(cubature) # load the package "cubature"
## Warning: package 'cubature' was built under R version 3.3.3
f <- function(x) { 2/3 * (x[1] + x[2] + x[3]) }
# "x" is vector x[1], x[2], x[3] are referring to x1, x2 and x3 respectively.
adaptIntegrate(f, lowerLimit = c(0, 0, 0), upperLimit = c(0.5, 0.5, 0.5))

x<-c(10,20,30,20,12,20,20)
diff(x)
cumsum(x)
```

$$\frac{-\partial^2 Q}{\partial\theta^2} - VarCov\left[\frac{\partial\log(p(\theta|y,z)}{\partial\theta}\right] = \begin{cases} \frac{n}{\sigma^2} - \frac{n-m}{\sigma^2} = \frac{m}{\sigma^2} \\ \frac{1}{\sigma^2}\sum_{i=1}^n x_i^2 - \frac{1}{\sigma^2}\sum_{i=m+1}^n x_i^2 = \frac{1}{\sigma^2}\sum_{i=1}^m x_i^2 \\ -\frac{n}{2\sigma^4} + \frac{1}{\sigma^6}\sum_{j=1}^m [y_j-\mu_j]^2 + \frac{n-m}{\sigma^4} - \frac{n-m}{2\sigma^4} = -\frac{m}{2\sigma^4} + \frac{1}{\sigma^6}\sum_{j=1}^m [y_j-\mu_j]^2 \end{cases}$$

$$\frac{\partial Q}{\partial\beta_0} = \frac{1}{\sigma^2}\sum_{j=1}^m [y_j - \beta_0 - \beta_1 x_j] + \frac{1}{\sigma^2}\sum_{i=m+1}^n [z_i - \beta_0 - \beta_1 x_i]$$

$$\frac{\partial^2 Q}{\partial\beta_0^2} = -\frac{m}{\sigma^2} - \frac{n-m}{\sigma^2} = -\frac{n}{\sigma^2} \qquad \square$$

$$\frac{\partial^2 Q}{\partial\beta_0\partial\beta_1} = -\frac{1}{\sigma^2}\sum_{j=1}^m x_j - \frac{1}{\sigma^2}\sum_{i=m+1}^n x_i = -\frac{1}{\sigma^2}\sum_{i=1}^n x_i \qquad \square$$

$$\frac{\partial^2 Q}{\partial\beta_0\partial\sigma^2} = -\frac{1}{\sigma^4}\sum_{j=1}^m [y_j - \mu_j] - \frac{1}{\sigma^4}\sum_{i=m+1}^n [z_i - \mu_i] \qquad \square$$

....

$$
\begin{aligned}
\frac{\partial Q}{\partial \beta_1} &= \frac{1}{\sigma^2}\sum_{j=1}^{m}[y_j-\beta_0-\beta_1 x_j]x_j + \frac{1}{\sigma^2}\sum_{i=m+1}^{n}[c_i-\beta_0-\beta_1 x_i]x_i \\
\frac{\partial^2 Q}{\partial \beta_1 \partial \beta_0} &= -\frac{1}{\sigma^2}\sum_{j=1}^{m}x_j - \frac{1}{\sigma^2}\sum_{i=m+1}^{n}x_i = -\frac{1}{\sigma^2}\sum_{i=1}^{n}x_i && \square \\
\frac{\partial^2 Q}{\partial \beta_1^2} &= -\frac{1}{\sigma^2}\sum_{j=1}^{m}x_j^2 - \frac{1}{\sigma^2}\sum_{i=m+1}^{n}x_i^2 = -\frac{1}{\sigma^2}\sum_{i=1}^{n}x_i^2 && \square \\
\frac{\partial^2 Q}{\partial \beta_1 \partial \sigma^2} &= -\frac{1}{\sigma^4}\sum_{j=1}^{m}[y_j-\mu_j]x_j - \frac{1}{\sigma^4}\sum_{i=m+1}^{n}[z_i-\mu_i]x_i && \square
\end{aligned}
$$

. . . .

$$
\begin{aligned}
\frac{\partial Q}{\partial \sigma^2} &= -\frac{n}{2\sigma^2} + \frac{1}{2\sigma^4}\sum_{j=1}^{m}[y_j-\mu_j]^2 + \frac{1}{2\sigma^4}\sum_{i=m+1}^{n}[(\mu_i^\star-\mu_i)^2+\sigma^{\star 2}+\sigma^\star H(\cdot)(z_i+\mu_i^\star-2\mu_i)] \\
&= -\frac{n}{2\sigma^2} + \frac{1}{2\sigma^4}\sum_{j=1}^{m}[y_j-\mu_j]^2 + \frac{1}{2\sigma^4}\sum_{i=m+1}^{n}[(z_i-\mu_i)^2] \\
\frac{\partial^2 Q}{\partial \sigma^2 \partial \beta_0} &= -\frac{1}{\sigma^4}\sum_{j=1}^{m}[y_j-\mu_j] - \frac{1}{\sigma^4}\sum_{i=m+1}^{n}[Z_i-\mu_i] && \square \\
\frac{\partial^2 Q}{\partial \sigma^2 \partial \beta_1} &= -\frac{1}{\sigma^4}\sum_{j=1}^{m}[y_j-\mu_j]x_j - \frac{1}{\sigma^4}\sum_{i=m+1}^{n}[Z_i-\mu_i]x_i && \square \\
\frac{\partial^2 Q}{\partial (\sigma^2)^2} &= \frac{n}{2\sigma^4} - \frac{1}{\sigma^6}\sum_{j=1}^{m}[y_j-\mu_j]^2 - \frac{1}{\sigma^6}\sum_{i=m+1}^{n}E[(z_i-\mu_i)^2] && \square
\end{aligned}
$$

Missing Information: H function

$$
\log(p(\beta_0,\beta_1,\sigma|y,z)) = -\frac{n}{2}\ln(2\pi) - \frac{n}{2}\ln(\sigma^2) - \frac{1}{2\sigma^2}\sum_{j=1}^{m}(y_j-\beta_0-\beta_1 x_j)^2 - \frac{1}{2\sigma^2}\sum_{i=m+1}^{n}(Z_i-\beta_0-\beta_1 x_i)^2
$$

$$
\begin{aligned}
\frac{\partial \log(p(\beta_0,\beta_1,\sigma|y,z)}{\partial \beta_0} &= \frac{1}{\sigma^2}\sum_{j=1}^{m}[y_j-\beta_0-\beta_1 x_j] + \frac{1}{\sigma^2}\sum_{i=m+1}^{n}[Z_i-\beta_0-\beta_1 x_i] \\
\frac{\partial \log(p(\vec{\theta}|y,z)}{\partial \beta_1} &= \frac{1}{\sigma^2}\sum_{j=1}^{m}[y_j-\beta_0-\beta_1 x_j]x_j + \frac{1}{\sigma^2}\sum_{i=m+1}^{n}[Z_i-\beta_0-\beta_1 x_i]x_i \\
\frac{\partial \log(p(\vec{\theta}|y,z)}{\partial \sigma^2} &= -\frac{n}{2\sigma^2} + \frac{1}{2\sigma^4}\sum_{j=1}^{m}[y_j-\beta_0-\beta_1 x_j]^2 + \frac{1}{2\sigma^4}\sum_{i=m+1}^{n}[Z_i-\beta_0-\beta_1 x_i]^2
\end{aligned}
$$

$$Var\left[\frac{\partial \log(p(\vec{\theta}|y,z)}{\partial\beta_0}\right] = \frac{1}{\sigma^4}\sum_{i=m+1}^{n} Var[Z_i] = \frac{n-m}{\sigma^2} \qquad \text{details down below}$$

$$Cov\left[\frac{\partial \log(p(\vec{\theta}|y,z)}{\partial\beta_0}, \frac{\partial \log(p(\vec{\theta}|y,z)}{\partial\beta_1}\right] = \frac{1}{\sigma^2}\sum_{j=1}^{m} x_j + \frac{1}{\sigma^2}\sum_{i=m+1}^{n} x_i = \frac{1}{\sigma^2}\sum_{i=m+1}^{n} x_i \qquad \square$$

$$Cov\left[\frac{\partial \log(p(\vec{\theta}|y,z)}{\partial\beta_0}, \frac{\partial \log(p(\vec{\theta}|y,z)}{\partial\sigma^2}\right] = \frac{1}{\sigma^4}\sum_{i=m+1}^{n} [z_i-\mu_i] \qquad \square$$

. . . .

$$Cov\left[\frac{\partial \log(p(\vec{\theta}|y,z)}{\partial\beta_1}, \frac{\partial \log(p(\vec{\theta}|y,z)}{\partial\beta_0}\right] = \frac{1}{\sigma^2}\sum_{j=1}^{m} x_j + \frac{1}{\sigma^2}\sum_{i=m+1}^{n} x_i = \frac{1}{\sigma^2}\sum_{i=m+1}^{n} x_i \qquad \square$$

$$Var\left[\frac{\partial \log(p(\vec{\theta}|y,z)}{\partial\beta_1}\right] = \frac{1}{\sigma^4}\sum_{i=m+1}^{n} Var[x_iZ_i] = \frac{1}{\sigma^4}\sum_{i=m+1}^{n} x_i^2 Var[Z_i] = \frac{1}{\sigma^2}\sum_{i=m+1}^{n} x_i^2 \qquad \square$$

$$Cov\left[\frac{\partial \log(p(\vec{\theta}|y,z)}{\partial\beta_1}, \frac{\partial \log(p(\vec{\theta}|y,z)}{\partial\sigma^2}\right] = \frac{1}{\sigma^4}\sum_{i=m+1}^{n} [z_i-\mu_i]x_i \qquad \square$$

. . . .

$$Cov\left[\frac{\partial \log(p(\vec{\theta}|y,z)}{\partial\sigma^2}, \frac{\partial \log(p(\vec{\theta}|y,z)}{\partial\beta_0}\right] = \frac{1}{\sigma^4}\sum_{i=m+1}^{n} [z_i-\mu_i] \qquad \square$$

$$Cov\left[\frac{\partial \log(p(\vec{\theta}|y,z)}{\partial\sigma^2}, \frac{\partial \log(p(\vec{\theta}|y,z)}{\partial\beta_1}\right] = \frac{1}{\sigma^4}\sum_{i=m+1}^{n} [z_i-\mu_i]x_i \qquad \square$$

$$\begin{aligned} Var\left[\frac{\partial \log(p(\vec{\theta}|y,z)}{\partial\sigma^2}\right] &= \frac{1}{4\sigma^8}\sum_{j=m+1}^{n} Var[(z_i-\mu_i)^2] = \frac{1}{4\sigma^8}\sum_{j=m+1}^{n}\{4\sigma^2 E[(z_i-\mu_i)^2] - 2\sigma^4\} \\ &= -\frac{n-m}{2\sigma^4} + \frac{1}{\sigma^6}\sum_{j=m+1}^{n} E[(z_i-\mu_i)^2] \qquad \square \end{aligned}$$

$$\frac{\partial \log(p(\vec{\theta}|y)}{\partial\beta_0} = \frac{1}{\sigma^2}\sum_{j=1}^{m}[y_j-\beta_0-\beta_1x_j] \qquad \frac{\partial^2 \log(p(\vec{\theta}|y)}{\partial\beta_0^2} = -\frac{m}{\sigma^2}$$

$$\frac{\partial \log(p(\vec{\theta}|y)}{\partial\beta_1} = \frac{1}{\sigma^2}\sum_{j=1}^{m}[y_j-\beta_0-\beta_1x_j]x_j \qquad \frac{\partial^2 \log(f(\beta_0,\beta_1,\sigma|y)}{\partial\beta_1^2} = -\frac{1}{\sigma^2}\sum_{j=1}^{m} x_j^2$$

$$\frac{\partial \log(p(\vec{\theta}|y)}{\partial\sigma^2} = -\frac{m}{2\sigma^2} + \frac{1}{2\sigma^4}\sum_{j=1}^{m}[y_j-\beta_0-\beta_1x_j]^2 \qquad \frac{\partial^2 \log(p(\vec{\theta}|y)}{\partial(\sigma^2)^2} = \frac{m}{2\sigma^4} - \frac{1}{\sigma^6}\sum_{j=1}^{m}[y_j-\mu_j]^2$$

$$E[\varepsilon_i|Z_i > c_i, \vec{\theta}^\star] == \frac{\phi(\frac{c_i-\mu_i^\star}{\sigma^\star})}{1-\Phi(\frac{c_i-\mu_i^\star}{\sigma^\star})} = H(\frac{c_i-\mu_i^\star}{\sigma^\star}) \qquad \square$$

$$\begin{aligned}
E[\varepsilon_i^2|Z_i > c_i, \vec{\theta}^\star] &= \int_{\frac{c_i-\mu_i^\star}{\sigma^\star}}^{\infty} \frac{\varepsilon_i^2\phi(\varepsilon_i)}{1-\Phi(\frac{c_i-\mu_i^\star}{\sigma^\star})} d\varepsilon_i \\
&= \frac{1}{1-\Phi(\cdot)} \int_{\frac{c_i-\mu_i^\star}{\sigma^\star}}^{\infty} \varepsilon_i^2 \frac{1}{\sqrt{2\pi}} e^{\frac{-\varepsilon_i^2}{2}} d\varepsilon_i \\
&= \frac{-1}{1-\Phi(\cdot)} \left\{ \left[\varepsilon_i \frac{1}{\sqrt{2\pi}} e^{\frac{-\varepsilon_i^2}{2}} \right]_{\frac{c_i-\mu_i^\star}{\sigma^\star}}^{\infty} - \int_{\frac{c_i-\mu_i^\star}{\sigma^\star}}^{\infty} \frac{1}{\sqrt{2\pi}} e^{\frac{-\varepsilon_i^2}{2}} d\varepsilon_i \right\} \\
&= \frac{-1}{1-\Phi(\cdot)} \left\{ -\frac{c_i-\mu_i^\star}{\sigma^\star} \frac{1}{\sqrt{2\pi}} e^{\frac{-(\frac{c_i-\mu_i^\star}{\sigma^\star})^2}{2}} - (1-\Phi(\cdot)) \right\} \\
&= \frac{c_i-\mu_i^\star}{\sigma^\star} H(\cdot) + 1 \qquad \square
\end{aligned}$$

$$\begin{aligned}
E[\varepsilon_i^3|Z_i > c_i, \vec{\theta}^\star] &= \frac{1}{1-\Phi(\cdot)} \int_{\frac{c_i-\mu_i^\star}{\sigma^\star}}^{\infty} \varepsilon_i^3 \frac{1}{\sqrt{2\pi}} e^{\frac{-\varepsilon_i^2}{2}} d\varepsilon_i \\
&= \frac{2}{1-\Phi(\cdot)} \int_{\frac{c_i-\mu_i^\star}{\sigma^\star}}^{\infty} \frac{-\varepsilon_i^2}{2} \frac{1}{\sqrt{2\pi}} e^{\frac{-\varepsilon_i^2}{2}} d(\frac{-\varepsilon_i^2}{2}) \\
&= \frac{2}{1-\Phi(\cdot)} \left\{ \left[\frac{-\varepsilon_i^2}{2} \frac{1}{\sqrt{2\pi}} e^{\frac{-\varepsilon_i^2}{2}} \right]_{\frac{c_i-\mu_i^\star}{\sigma^\star}}^{\infty} - \int_{\frac{c_i-\mu_i^\star}{\sigma^\star}}^{\infty} \frac{1}{\sqrt{2\pi}} e^{\frac{-\varepsilon_i^2}{2}} d(\frac{-\varepsilon_i^2}{2}) \right\} \\
&= \frac{2}{1-\Phi(\cdot)} \left\{ -(-\frac{1}{2})(\frac{c_i-\mu_i^\star}{\sigma^\star})^2 \frac{1}{\sqrt{2\pi}} e^{\frac{-(\frac{c_i-\mu_i^\star}{\sigma^\star})^2}{2}} - \left[\frac{1}{\sqrt{2\pi}} e^{\frac{-\varepsilon_i^2}{2}} \right]_{\frac{c_i-\mu_i^\star}{\sigma^\star}}^{\infty} \right\} \\
&= \frac{1}{1-\Phi(\cdot)} \left[(\frac{c_i-\mu_i^\star}{\sigma^\star})^2 + 2 \right] \frac{1}{\sqrt{2\pi}} e^{-\frac{(\frac{c_i-\mu_i^\star}{\sigma^\star})^2}{2}} \\
&= \left[(\frac{c_i-\mu_i^\star}{\sigma^\star})^2 + 2 \right] H(\cdot) \qquad \square
\end{aligned}$$

$$\begin{aligned}
E[\varepsilon_i^4|Z_i > c_i, \vec{\theta}^\star] &= \frac{-1}{1-\Phi(\cdot)} \int_{\frac{c_i-\mu_i^\star}{\sigma^\star}}^{\infty} \varepsilon_i^3 \frac{-\varepsilon_i}{\sqrt{2\pi}} e^{\frac{-\varepsilon_i^2}{2}} d\varepsilon_i \\
&= \frac{-1}{1-\Phi(\cdot)} \left\{ \left[\varepsilon_i^3 \frac{1}{\sqrt{2\pi}} e^{\frac{-\varepsilon_i^2}{2}} \right]_{\frac{c_i-\mu_i^\star}{\sigma^\star}}^{\infty} - \int_{\frac{c_i-\mu_i^\star}{\sigma^\star}}^{\infty} \frac{3\varepsilon_i^2}{\sqrt{2\pi}} e^{\frac{-\varepsilon_i^2}{2}} d\varepsilon_i \right\} \\
&= \frac{-1}{1-\Phi(\cdot)} \left\{ -(\frac{c_i-\mu_i^\star}{\sigma^\star})^3 \frac{1}{\sqrt{2\pi}} e^{\frac{-(\frac{c_i-\mu_i^\star}{\sigma^\star})^2}{2}} \right\} + 3[\frac{c_i-\mu_i^\star}{\sigma^\star} H(\cdot) + 1] \\
&= (\frac{c_i-\mu_i^\star}{\sigma^\star})^3 H(\cdot) + 3[\frac{c_i-\mu_i^\star}{\sigma^\star} H(\cdot) + 1] \\
&= [(\frac{c_i-\mu_i^\star}{\sigma^\star})^3 + 3\frac{c_i-\mu_i^\star}{\sigma^\star}] H(\cdot) + 3 \qquad \square
\end{aligned}$$

....

$$\begin{aligned}
E[Z_i|Z_i > c_i, \vec{\theta}^\star] &= E[\mu_i^\star + \sigma^\star\varepsilon_i|.] = \mu_i^\star + \sigma^\star E[\varepsilon_i|.] = \mu_i^\star + \sigma^\star H(\frac{c_i - \mu_i^\star}{\sigma^\star}) \quad \square \\
E[Z_i^2|Z_i > c_i, \vec{\theta}^\star] &= \mu_i^{\star 2} + 2\mu_i^\star\sigma^\star E[\varepsilon_i|.] + \sigma^{\star 2}E[\varepsilon_i^2|.] \\
&= \mu_i^{\star 2} + 2\mu_i^\star\sigma^\star H(\cdot) + \sigma^{\star 2}[\frac{c_i - \mu_i^\star}{\sigma^\star}H(\cdot) + 1] \\
&= \mu_i^{\star 2} + \sigma^\star(c_i + \mu_i^\star)H(\frac{c_i - \mu_i^\star}{\sigma^\star}) + \sigma^{\star 2} \quad \square \\
E[Z_i^4|Z_i > c_i, \vec{\theta}^\star] &= \mu_i^{\star 4} + 4\mu_i^{\star 3}\sigma^\star E[\varepsilon_i|.] + 6\mu_i^{\star 2}\sigma^{\star 2}E[\varepsilon_i^2|.] + 4\mu_i^\star\sigma^{\star 3}E[\varepsilon_i^3|.] + \sigma^{\star 4}E[\varepsilon_i^4|.] \\
&= \mu_i^{\star 4} + 4\mu_i^{\star 3}\sigma^\star H(\cdot) + 6\mu_i^{\star 2}\sigma^{\star 2}\left[\frac{c_i - \mu_i^\star}{\sigma^\star}H(\cdot) + 1\right] + 4\mu_i^\star\sigma^{\star 3}\left[(\frac{c_i - \mu_i^\star}{\sigma^\star})^2 + 2\right]H(\cdot) \\
&\quad + \sigma^{\star 4}\left\{[(\frac{c_i - \mu_i^\star}{\sigma^\star})^3 + 3\frac{c_i - \mu_i^\star}{\sigma^\star}]H(\cdot) + 3\right\} \\
&= \mu_i^{\star 4} + 6\mu_i^{\star 2}\sigma^{\star 2} + 3\sigma^{\star 4} + H(\cdot)\left[3\mu_i^{\star 3}\sigma^\star - 5\mu_i^{\star 2}\sigma^\star c_i + \mu_i^\star\sigma^\star c_i^2 + 5\mu_i^\star\sigma^{\star 3} + 3\sigma^{\star 3}c_i + \sigma^\star c_i^3\right] \quad \square
\end{aligned}$$

$$\begin{aligned}
Var[Z_i|Z_i > c_i, \vec{\theta}^\star] &= E[Z^2|.] - (E[Z|.])^2 \\
&= \mu_i^{\star 2} + \sigma^{\star 2} + \sigma^\star(c_i + \mu_i^\star)H(\frac{c_i - \mu_i^\star}{\sigma^\star}) - \left[\mu_i^\star + \sigma^\star H(\frac{c_i - \mu_i^\star}{\sigma^\star})\right]^2 \\
&= \sigma^{\star 2}[1 - H^2(\frac{c_i - \mu_i^\star}{\sigma^\star})] + \sigma^\star(c_i - \mu_i^\star)H(\frac{c_i - \mu_i^\star}{\sigma^\star}) \\
Var[Z_i] &= \sigma^2 \quad \square
\end{aligned}$$

$$\begin{aligned}
Var[Z_i^2|Z_i > c_i, \vec{\theta}^\star] &= E[Z^4|.] - (E[Z^2|.])^2 \\
&= \mu_i^{\star 4} + 6\mu_i^{\star 2}\sigma^{\star 2} + 3\sigma^{\star 4} + H(\cdot)\left[3\mu_i^{\star 3}\sigma^\star - 5\mu_i^{\star 2}\sigma^\star c_i + \mu_i^\star\sigma^\star c_i^2 + 5\mu_i^\star\sigma^{\star 3} + 3\sigma^{\star 3}c_i + \sigma^\star c_i^3\right] \\
&\quad - \left[\mu_i^{\star 2} + \sigma^\star(c_i + \mu_i^\star)H(\cdot) + \sigma^{\star 2}\right]^2 \\
&= \mu_i^{\star 4} + 6\mu_i^{\star 2}\sigma^{\star 2} + 3\sigma^{\star 4} + H(\cdot)\left[3\mu_i^{\star 3}\sigma^\star - 5\mu_i^{\star 2}\sigma^\star c_i + \mu_i^\star\sigma^\star c_i^2 + 5\mu_i^\star\sigma^{\star 3} + 3\sigma^{\star 3}c_i + \sigma^\star c_i^3\right] \\
&\quad - \{\mu_i^{\star 4} + \sigma^{\star 4} + \sigma^{\star 2}(c_i + \mu_i^\star)^2H^2(\cdot) + H(\cdot)[2\mu_i^{\star 2}\sigma^\star c_i + 2\mu_i^{\star 3}\sigma^\star + 2\sigma^{\star 3}c_i + 2\mu_i^\star\sigma^{\star 3}] + 2\mu_i^{\star 2}\sigma^{\star 2}\} \\
&= 2\sigma^{\star 4} + 4\mu_i^{\star 2}\sigma^{\star 2} + \sigma^{\star 2}(c_i + \mu_i^\star)^2H^2(\cdot) + H(\cdot)[\mu_i^{\star 3}\sigma^\star - 7\mu_i^{\star 2}\sigma^\star c_i + \mu_i^\star\sigma^\star c_i^2 + 3\mu_i^\star\sigma^{\star 3} + \sigma^{\star 3}c_i + \sigma^\star c_i^3] \\
Var[Z_i^2] &= 2\sigma^4 + 4\mu_i^2\sigma^2 \quad \square
\end{aligned}$$

....

$$\begin{aligned}
E[Z_i - \beta_0 - \beta_1x_i|Z_i > c_i, \vec{\theta}^\star] &= \mu_i^\star + \sigma^\star H(\frac{c_i - \mu_i^\star}{\sigma^\star}) - (\beta_0 + \beta_1x_i) \quad \square \\
E[(Z_i - \beta_0 - \beta_1x_i)^2|Z_i > c_i, \vec{\theta}^\star] &= \mu_i^{\star 2} + \sigma^{\star 2} + \sigma^\star(c_i + \mu_i^\star)H(\cdot) - 2(\beta_0 + \beta_1x_i)[\mu_i^\star + \sigma^\star H(\cdot)] + (\beta_0 + \beta_1x_i)^2 \quad \square
\end{aligned}$$

HW3 Extra

$$\begin{aligned}
f(\mathbf{y},\alpha|\mu,\Sigma) &= \exp\frac{-1}{2}\left\{(\begin{bmatrix}\mathbf{y}\\ \alpha\end{bmatrix}-\mu)'\Sigma^{-1}(\begin{bmatrix}\mathbf{y}\\ \alpha\end{bmatrix}-\mu)+(N+a)\log(2\pi)+\log(|\Sigma|)\right\}\\
&= \exp\frac{-1}{2}\left\{\begin{bmatrix}\mathbf{y}-1\beta\\ \alpha\end{bmatrix}'\Sigma^{-1}\begin{bmatrix}\mathbf{y}-1\beta\\ \alpha\end{bmatrix}+C+\log(|\Sigma|)\right\}\\
&= \exp\frac{-1}{2}\left\{\begin{bmatrix}\mathbf{y}-1\beta\\ \alpha\end{bmatrix}'\begin{bmatrix}I & 0\\ -\Sigma_{22}^{-1}\Sigma_{21} & I\end{bmatrix}\begin{bmatrix}(\frac{\Sigma}{\Sigma_{22}})^{-1} & 0\\ 0 & \Sigma_{22}^{-1}\end{bmatrix}\begin{bmatrix}I & -\Sigma_{12}\Sigma_{22}^{-1}\\ 0 & I\end{bmatrix}\begin{bmatrix}\mathbf{y}-1\beta\\ \alpha\end{bmatrix}+C+\log(|\Sigma|)\right\}\\
&= \exp\frac{-1}{2}\left\{(\mathbf{y}-1\beta-\Sigma_{12}\Sigma_{22}^{-1}\alpha)'(\frac{\Sigma}{\Sigma_{22}})^{-1}(\mathbf{y}-1\beta-\Sigma_{12}\Sigma_{22}^{-1}\alpha)+\alpha^{'}\Sigma_{22}^{-1}\alpha+C+\log(|\Sigma|)\right\}\\
&= \exp\frac{-1}{2}\left\{(\mathbf{y}-Z\alpha-1\beta)'\frac{I_3}{\sigma_\varepsilon^2}(\mathbf{y}-Z\alpha-1\beta)+\alpha^{'}\frac{I_a}{\sigma_\alpha^2}\alpha+\log(\sigma_\varepsilon^{2N}\sigma_\alpha^{2a})+C\right\}\\
&= \exp\frac{-1}{2}\left\{(\frac{\mathbf{y}-Z\alpha)'I_N(\mathbf{y}-Z\alpha)}{\sigma_\varepsilon^2}-\frac{2(\mathbf{y}-Z\alpha)'I_N1\beta}{\sigma_\varepsilon^2}+\frac{1^{'}\beta I_N1\beta}{\sigma_\varepsilon^2}+\frac{\alpha^{'}I_a\alpha}{\sigma_\alpha^2}+N\log(\sigma_\varepsilon^2)+a\log(\sigma_\alpha^2)+C\right\}\\
&= \exp\frac{-1}{2}\left\{\frac{1}{\sigma_\varepsilon^2}(\mathbf{y}-Z\alpha)'(\mathbf{y}-Z\alpha)'-\frac{2\beta}{\sigma_\varepsilon^2}(\mathbf{y}-Z\alpha)'1+\frac{\beta^2}{\sigma_\varepsilon^2}+N\log(\sigma_\varepsilon^2)+\frac{1}{\sigma_\alpha^2}\alpha^{'}\alpha+a\log(\sigma_\alpha^2)+C\right\}
\end{aligned}$$

Therefore, the sufficient statistics are $\frac{1}{N}(\mathbf{y}-Z\alpha)'(\mathbf{y}-Z\alpha)$, $\frac{1}{a}\alpha^{'}\alpha$, and $\frac{1}{N}(\mathbf{y}-Z\alpha)'1$.

Let $\frac{\partial f(\mathbf{y},\alpha|\mu,\Sigma)}{\partial\beta}=0$

$$\frac{2}{\sigma_\varepsilon^2}(\mathbf{y}-Z\alpha-1\beta)=0 \implies \hat{\beta}=\frac{1}{N}(\mathbf{y}-Z\alpha)'1$$

Let $\eta_\varepsilon=1/\sigma_\varepsilon^2$; $\eta_\alpha=1/\sigma_\alpha^2$.

$\sigma_\varepsilon^2=1/\eta_\varepsilon$; $\sigma_\alpha^2=1/\eta_\alpha$.

$B(\sigma_\varepsilon^2,\sigma_\alpha^2,\beta)=-N\log(\sigma_\varepsilon^2)-a\log(\sigma_\alpha^2)$

$$E[(\mathbf{y}-Z\alpha)'(\mathbf{y}-Z\alpha)]=A'(\eta_\varepsilon)=N/\eta_\varepsilon=N\sigma_\varepsilon^2$$
$$E[\alpha^{'}\alpha]=A'(\eta_\alpha)=a/\eta_\alpha=a\sigma_\alpha^2$$

Therefore, $E[\frac{1}{N}(\mathbf{y}-Z\alpha)'(\mathbf{y}-Z\alpha)]=\sigma_\varepsilon^2$, $E[\frac{1}{a}\alpha^{'}\alpha]=\sigma_\alpha^2$, and $E[\frac{1}{N}(\mathbf{y}-Z\alpha)'1]=\beta$

....

Details of Matrix operation

$$\Sigma=\begin{bmatrix}\Sigma_{11}=V & \Sigma_{12}=\sigma_\alpha^2Z\\ \Sigma_{21}=\sigma_\alpha^2Z' & \Sigma_{22}=\sigma_\alpha^2I_a\end{bmatrix},\qquad |\Sigma|=\sigma_\varepsilon^{2N}\sigma_\alpha^{2a}\qquad\square$$

$$\Sigma_{11}=\begin{bmatrix}\sigma_\varepsilon^2I_1+\sigma_\alpha^2J_1 & 0\\ 0 & \sigma_\varepsilon^2I_2+\sigma_\alpha^2J_2\end{bmatrix}$$

By $(aI_k+bJ_k)^{-1}=\frac{I_k}{a}-\frac{bJ_k}{a(a+kb)}$,

$$\Sigma_{11}^{-1} = \begin{bmatrix} (\sigma_\varepsilon^2 I_1 + \sigma_\alpha^2 J_1)^{-1} & 0 \\ 0 & (\sigma_\varepsilon^2 I_2 + \sigma_\alpha^2 J_2)^{-1} \end{bmatrix} = \begin{bmatrix} \frac{I_1}{\sigma_\varepsilon^2} - \frac{\sigma_\alpha^2 J_1}{\sigma_\varepsilon^2+\sigma_\alpha^2} & 0 \\ 0 & \frac{I_2}{\sigma_\varepsilon^2} - \frac{\sigma_\alpha^2 J_2}{\sigma_\varepsilon^2+2\sigma_\alpha^2} \end{bmatrix} = \begin{bmatrix} \frac{1}{\sigma_\varepsilon^2+\sigma_\alpha^2} & 0 & 0 \\ 0 & \frac{\sigma_\varepsilon^2+\sigma_\alpha^2}{\sigma_\varepsilon^2(\sigma_\varepsilon^2+2\sigma_\alpha^2)} & \frac{-\sigma_\alpha^2}{\sigma_\varepsilon^2(\sigma_\varepsilon^2+2\sigma_\alpha^2)} \\ 0 & \frac{-\sigma_\alpha^2}{\sigma_\varepsilon^2(\sigma_\varepsilon^2+2\sigma_\alpha^2)} & \frac{\sigma_\varepsilon^2+\sigma_\alpha^2}{\sigma_\varepsilon^2(\sigma_\varepsilon^2+2\sigma_\alpha^2)} \end{bmatrix}$$

$$\Sigma_{22}^{-1} = \frac{I_a}{\sigma_\alpha^2}\Sigma_{12}\Sigma_{22}^{-1}\alpha = Z\alpha \qquad \square$$

The partitional matrices

$$\underbrace{\begin{bmatrix} I & -\Sigma_{12}\Sigma_{22}^{-1} \\ 0 & I \end{bmatrix}}_{X} \underbrace{\begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} I & 0 \\ -\Sigma_{22}^{-1}\Sigma_{21} & I \end{bmatrix}}_{Y} = \begin{bmatrix} \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21} & 0 \\ 0 & \Sigma_{22} \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{\Sigma}{\Sigma_{22}} & 0 \\ 0 & \Sigma_{22} \end{bmatrix}}_{Z}$$

By Schur decomplement of Σ w.r.t Σ_{22} is $\frac{\Sigma}{\Sigma_{22}} = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}$ and it is invertible.

$$\frac{\Sigma}{\Sigma_{11}} = \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}$$

$$\begin{aligned} \Sigma^{-1} = YZ^{-1}X &= \begin{bmatrix} I & 0 \\ -\Sigma_{22}^{-1}\Sigma_{21} & I \end{bmatrix} \begin{bmatrix} (\frac{\Sigma}{\Sigma_{22}})^{-1} & 0 \\ 0 & \Sigma_{22}^{-1} \end{bmatrix} \begin{bmatrix} I & -\Sigma_{12}\Sigma_{22}^{-1} \\ 0 & I \end{bmatrix} \\ &= \begin{bmatrix} (\frac{\Sigma}{\Sigma_{22}})^{-1} & -(\frac{\Sigma}{\Sigma_{22}})^{-1}\Sigma_{12}\Sigma_{22}^{-1} \\ -\Sigma_{22}^{-1}\Sigma_{21}(\frac{\Sigma}{\Sigma_{22}})^{-1} & (\frac{\Sigma}{\Sigma_{22}})^{-1} + \Sigma_{22}^{-1}\Sigma_{21}(\frac{\Sigma}{\Sigma_{22}})^{-1}\Sigma_{12}\Sigma_{22}^{-1} \end{bmatrix} \end{aligned}$$

Alternatively,

$$\Sigma^{-1} = \begin{bmatrix} (\frac{\Sigma}{\Sigma_{11}})^{-1} + \Sigma_{11}^{-1}\Sigma_{12}(\frac{\Sigma}{\Sigma_{11}})^{-1}\Sigma_{21}\Sigma_{11}^{-1} & -\Sigma_{11}^{-1}\Sigma_{12}(\frac{\Sigma}{\Sigma_{11}})^{-1} \\ -(\frac{\Sigma}{\Sigma_{11}})^{-1}\Sigma_{21}\Sigma_{11}^{-1} & (\frac{\Sigma}{\Sigma_{11}})^{-1} \end{bmatrix}$$

$$\begin{aligned} C_{11} = (\frac{\Sigma}{\Sigma_{22}})^{-1} \quad &= (\Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21})^{-1} = \begin{bmatrix} \sigma_\varepsilon^2 I_1 + \sigma_\alpha^2 J_1 & 0 \\ 0 & \sigma_\varepsilon^2 I_2 + \sigma_\alpha^2 J_2 \end{bmatrix} - \sigma_\alpha^2 Z \frac{I_a}{\sigma_\alpha^2}\sigma_\alpha^2 Z' \quad = \frac{I_3}{\sigma_\varepsilon^2} \\ C_{12} = -(\frac{\Sigma}{\Sigma_{22}})^{-1}\Sigma_{12}\Sigma_{22}^{-1} \quad &= (\Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21} - \Sigma_{11})^{-1}\Sigma_{12}\Sigma_{22}^{-1} = -\frac{1}{\sigma_\varepsilon^2}Z \\ C_{21} = -(\frac{\Sigma}{\Sigma_{11}})^{-1}\Sigma_{21}\Sigma_{11}^{-1} \quad &= (\Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12} - \Sigma_{22})^{-1}\Sigma_{21}\Sigma_{11}^{-1} = -\frac{1}{\sigma_\varepsilon^2}Z' \\ C_{22} = (\frac{\Sigma}{\Sigma_{11}})^{-1} \quad &= (\Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12})^{-1} = \begin{bmatrix} \frac{\sigma_\varepsilon^2+\sigma_\alpha^2}{\sigma_\varepsilon^2\sigma_\alpha^2} & 0 \\ 0 & \frac{\sigma_\varepsilon^2+2\sigma_\alpha^2}{\sigma_\varepsilon^2\sigma_\alpha^2} \end{bmatrix} \end{aligned}$$

$$\Sigma^{-1} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} 1/\sigma_\varepsilon^2 & 0 & 0 & -1/\sigma_\varepsilon^2 & 0 \\ 0 & 1/\sigma_\varepsilon^2 & 0 & 0 & -1/\sigma_\varepsilon^2 \\ 0 & 0 & 1/\sigma_\varepsilon^2 & 0 & -1/\sigma_\varepsilon^2 \\ -1/\sigma_\varepsilon^2 & 0 & 0 & \frac{\sigma_\varepsilon^2+\sigma_\alpha^2}{\sigma_\varepsilon^2\sigma_\alpha^2} & 0 \\ 0 & -1/\sigma_\varepsilon^2 & -1/\sigma_\varepsilon^2 & 0 & \frac{\sigma_\varepsilon^2+2\sigma_\alpha^2}{\sigma_\varepsilon^2\sigma_\alpha^2} \end{bmatrix}$$

Midterm Project

- Pre-calculation

$(y_i, p_i),\ i = 1, ..m$ is the data with $z = 1$; $m = \sum\limits_{z_i=1} z_i$ is the number of $z = 1$;

(y_j, p_j) $j = m+1, ..n$ is the data with $z = 0$; $n - m$ is the number of $z = 0$.

Mixing proportions $p_i = Pr(z = 1)$; $u = \frac{y-\mu_2}{\sigma}$; $1 - p_j = Pr(z = 0)$; $v = \frac{y-\mu_1}{\sigma}$. $\vec{\theta} = (\mu_1, \mu_2, \sigma)$

$$\phi(u;\theta) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{y-\mu_2}{\sigma})^2}; \quad \phi(v;\theta) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{y-\mu_1}{\sigma})^2}$$

$$\frac{\partial \phi(u)}{\partial(\mu_1, \mu_2, \sigma)} = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{y-\mu_2}{\sigma})^2} (\frac{y-\mu_2}{\sigma}) \begin{bmatrix} 0 \\ 1/\sigma \\ u/\sigma \end{bmatrix} = \phi(u)\frac{u}{\sigma} \begin{bmatrix} 0 \\ 1 \\ u \end{bmatrix}$$

$$\frac{\partial \phi(v)}{\partial(\mu_1, \mu_2, \sigma)} = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{y-\mu_1}{\sigma})^2} (\frac{y-\mu_1}{\sigma}) \begin{bmatrix} 1/\sigma \\ 0 \\ v/\sigma \end{bmatrix} = \phi(v)\frac{v}{\sigma} \begin{bmatrix} 1 \\ 0 \\ v \end{bmatrix}$$

i) EM Algorithm:

E-Step

$$Pr(\vec{\theta}|\vec{y},\vec{z}) = \prod_{i=1}^{n} \left(\frac{z_i}{\sigma} \phi(\frac{y_i - \mu_2}{\sigma})^{z_i} p_i^{z_i} + \frac{1-z_i}{\sigma} \phi(\frac{y_j - \mu_1}{\sigma})^{1-z_i} (1-p_i)^{1-z_i} \right) = \prod_{i=1}^{m} \frac{1}{\sigma} \phi(u_i)^1 p_i^1 \cdot \prod_{j=m+1}^{n} \frac{1}{\sigma} \phi(v_j)^1 (1-p_j)^1$$

$$\log(Pr(\vec{\theta}|\vec{y},\vec{z}^\star)) = \sum_{i=1}^{m} \left[\log Pr(\vec{\theta}|y_i, z_i^\star = 1)\right] + \sum_{j=m+1}^{n} \left[\log Pr(\vec{\theta}|y_j, z_j^\star = 0)\right]$$

$$= C - n\log(\sigma) + \sum_{i=1}^{m} \log\phi(u_i) + \sum_{j=m+1}^{n} \log\phi(v_j) + \sum_{i=1}^{m} \log p_i + \sum_{j=m+1}^{n} \log(1-p_j)$$

$$Q(\vec{\theta},\vec{\theta}^\star) = \frac{1}{S}\sum_{s=1}^{S} \log(Pr(\vec{\theta}|\vec{y},\vec{z}^{(s)})) = C - n\log(\sigma) + \sum_{i=1}^{\bar{m}} \log\phi(u_i) + \sum_{j=\bar{m}+1}^{n} \log\phi(v_j) + \sum_{i=1}^{\bar{m}} \log p_i + \sum_{j=\bar{m}+1}^{n} \log(1-p_j)$$

where $\bar{m} = \frac{1}{S}\sum_{s=1}^{S}\sum_{z_i=1} z_i^{(s)}$

M-Step

$$\frac{\partial Q(\vec{\theta},\vec{\theta}^\star)}{\partial\theta} = \begin{bmatrix} 0 \\ 0 \\ \frac{-n}{\sigma} \end{bmatrix} + \sum_{i=1}^{m} \frac{\phi(u_i)}{\phi(u_i)} \begin{bmatrix} 0 \\ u_i/\sigma \\ u_i^2/\sigma \end{bmatrix} + \sum_{j=m+1}^{n} \frac{\phi(v_j)}{\phi(v_j)} \begin{bmatrix} v_j/\sigma \\ 0 \\ v_j^2/\sigma \end{bmatrix} = \frac{1}{\sigma} \begin{bmatrix} \sum_{j=m+1}^{n} v_j \\ \sum_{i=1}^{m} u_i \\ \sum_{i=1}^{m} u_i^2 + \sum_{j=m+1}^{n} v_j^2 - n \end{bmatrix} \overset{set}{=} 0$$

$$\begin{bmatrix} \hat{\mu}_1 = & \frac{1}{n-m}\sum_{j=m+1}^{n} y_j \\ \hat{\mu}_2 = & \frac{1}{m}\sum_{i=1}^{m} y_j \\ \hat{\sigma}^2 = & \frac{1}{n}\left(\sum_{i=1}^{m}(y_i - \mu_2)^2 + \sum_{j=m+1}^{n}(y_j - \mu_1)^2\right) \end{bmatrix}$$

$$r = Pr(z = 1|\vec{y}, \vec{\theta}^\star) = \frac{\phi(u^\star)p}{\phi(u^\star)p + \phi(v^\star)(1-p)}$$

```r
EM_Mix_Normal<- function(y,S,crit,itera)
{
## "y" (w,p) is the data; "n" is the data size
## "theta" is the parameter vector: mu1,mu2, sigma.
## "thetastar" is the current parameter estimate.
## "itera" is the upper limit of iterations.
##########################################################
s <- 0 ; set.seed(121) # iteration counter
n <- nrow(y)
z <- ifelse (p>=median(p),1,0)
V<- w[which(z==0)]
U<- w[which(z==1)]
mu <- c(mean(V),mean(U))
sigma <- sd(w)
thetastar <- c(mu,sigma)# initial parameter values
repeat {
v <- (w-mu[1])/sigma
u <- (w-mu[2])/sigma
r <- dnorm(u)*p/(dnorm(v)*(1-p)+dnorm(u)*p)
Z <- replicate(S,rbinom(20, 1, r))
W <- replicate(S,w)
U <- W*Z
V <- W*(1-Z)
V <- V[which(V!=0)]
U <- U[which(U!=0)]
mu <- c(mean(V),mean(U))
sigma <- sqrt(sum(c(V-mu[1],U-mu[2])^2)/n/S)
theta <- c(mu,sigma)
s <- s +1
if( (abs(thetastar[1]-theta[1])< crit)| (s > itera)) #  # (sum(thetastar-theta)^2< crit)|
break
thetastar <- theta
}
return(list(s,theta,cbind(y,z,r)))
}
Theta<- EM_Mix_Normal(y,S,crit=1e-4,itera=100)
```

In previous approach, if responsibilities $r \geq p_i$, this observation is more likely $z = 1$ given y and θ values. Else, let $z_j = 0$.

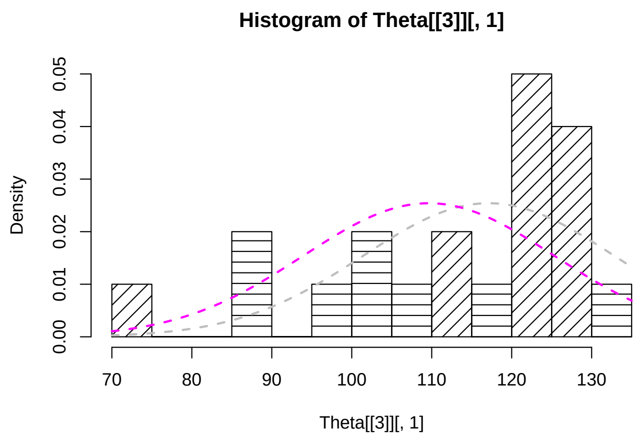
Then update the θ with preivous $\vec{z}$.Repeat the iteration until converge.

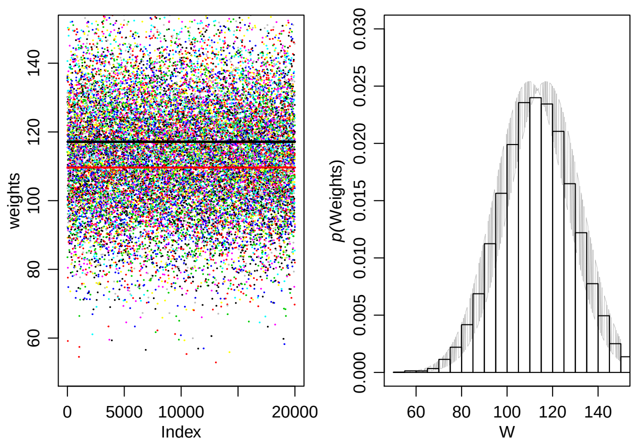
In current approach, we draw many times of $\vec{z}$ from Bernoulli($\vec{r}$) (or, equivalently, Binomial $(1, \vec{r})$) and get the mean values. The convergent result is $\theta = (\mu_1, \mu_2, \sigma) = 117.13086257,\ 109.63384189,\ 15.69912702$ respectively.

	x
Iterations	101

	x
mu1	117.130863
mu2	109.633842
sigma	15.699127

w	p	z	r
101.76	0.79	1	0.83918644
124.76	0.59	1	0.51081746
85.96	0.70	1	0.83475011
72.29	0.40	0	0.67961839
112.69	0.32	0	0.32422392
98.99	0.52	0	0.61900812
127.63	0.26	0	0.19038503
125.54	0.44	0	0.35805298
126.32	0.62	1	0.53118733
121.04	0.17	0	0.14166649
123.06	0.91	1	0.88501859
127.73	0.69	1	0.59766862
117.89	0.24	0	0.21757946
109.12	0.51	0	0.53989575
123.65	0.64	1	0.57100336
123.75	0.44	0	0.36972486
134.77	0.73	1	0.59674828
88.54	0.87	1	0.93090349
103.99	0.23	0	0.28011880
110.73	0.95	1	0.95342514





ii) Louis'Method:

By simulation method, We draw S times of $\vec{z} \sim$ Binomial $(1, \vec{r})$.

Then we can approximate the complete information and missing information.

- Complete Information:

$$\frac{\partial}{\partial\theta}\log(p(\vec{\theta}|y,z)) = \frac{1}{\sigma}\begin{bmatrix} \sum_{j=m+1}^{n} v_j \\ \sum_{i=1}^{m} u_i \\ \sum_{i=1}^{m} u_i^2 + \sum_{j=m+1}^{n} v_j^2 - n \end{bmatrix}$$

$$\frac{\partial^2}{\partial\theta^2}\log(p(\vec{\theta}|y,z)) = \frac{-1}{\sigma^2}\begin{bmatrix} n-m & 0 & 2\sum_{j=m+1}^{n} v_j \\ 0 & m & 2\sum_{i=1}^{m} u_i \\ 2\sum_{j=m+1}^{n} v_j & 2\sum_{i=1}^{m} u_i & 3(\sum_{i=1}^{m} u_i^2 + \sum_{j=m+1}^{n} v_j^2) - n \end{bmatrix}$$

$$E[\frac{\partial^2}{\partial\theta^2}\log(p(\vec{\theta}|y,z))] = \frac{1}{S}\sum_{1}^{S}\frac{-1}{\sigma^2}\begin{bmatrix} n-m & 0 & 2\sum_{j=m+1}^{n} v_j \\ 0 & m & 2\sum_{i=1}^{m} u_i \\ 2\sum_{j=m+1}^{n} v_j & 2\sum_{i=1}^{m} u_i & 3(\sum_{i=1}^{m} u_i^2 + \sum_{j=m+1}^{n} v_j^2) - n \end{bmatrix}$$

```
set.seed(121)
# r <- dnorm(u)*p/(dnorm(v)*(1-p)+dnorm(u)*p)
Z <- replicate(S,rbinom(20, 1, r))
W <- replicate(S,w)
V <- W*(1-Z)
U <- W*Z
V <- V[which(V!=0)]
U <- U[which(U!=0)]
v <- (V-mu[1])/sigma
u <- (U-mu[2])/sigma
(Compy <- matrix(c(sum(Z==0),   0      , 2*sum(v),
                      0,   sum(Z)     , 2*sum(u),
               2*sum(v), 2*sum(u) , 3*(sum(u^2)+sum(v^2))-n*S
),3,3,)/S/sigma^2)  # Complete Information
##               [,1]         [,2]          [,3]
## [1,]  0.0368615782 0.0000000000 -0.0015843734
## [2,]  0.0000000000 0.0442866402  0.0012859843
## [3,] -0.0015843734 0.0012859843  0.1643525191
```

- Missing Information:

$$Var\left[\frac{\partial}{\partial\theta}\log(p(\vec{\theta}|y,z))\right] = \frac{1}{S}\sum_{1}^{S}\left(\frac{\partial}{\partial\theta}\log(p(\vec{\theta}|y,z))\Big|_{\hat{\theta}}\right)^2 = \frac{1}{S}\sum_{1}^{S}\frac{1}{\sigma^2}\begin{bmatrix}\sum_{j=m+1}^{n} v_j \\ \sum_{i=1}^{m} u_i \\ \sum_{i=1}^{m} u_i^2 + \sum_{j=m+1}^{n} v_j^2 - n\end{bmatrix}^2$$

```
set.seed(121)  # MC simulation
M1<-M2<-M3<-matrix(NA,S,1)
M3 <- (colSums(u^2)+colSums(v^2))-n
V <- W*(1-Z)
U <- W*Z
M1 <- (colMeans(V)-mu[1])/sigma
M2 <- (colMeans(U)-mu[2])/sigma
M <- matrix(c(M1,M2,M3),S,3)
(Miss <- var(M)/sigma^2)
```

```
set.seed(121)  # MC simulation
M1<-M2<-M3<-matrix(NA,S,1)
M <- matrix(NA,S,3)
for (s in 1:S){
z_sim<-rbinom(20,1,r)
u_sim <- (w[which(z_sim==1)]-mu[2])/sigma
v_sim <- (w[which(z_sim==0)]-mu[1])/sigma
M1[s,] <- sum(v_sim)
M2[s,] <- sum(u_sim)
M3[s,] <- sum(u_sim^2)+sum(v_sim^2)
M[s,] <- c(M1[s],M2[s],M3[s]-n)
}
(Miss <- var(M)/sigma^2) # Missing Information
##              [,1]         [,2]         [,3]
## [1,]  0.015154881 -0.014296175 -0.014064169
## [2,] -0.014296175  0.016904994  0.014899924
## [3,] -0.014064169  0.014899924  0.013831623
```

```
(I <- Compy-Miss) # Apply the Louis' method
##             [,1]         [,2]         [,3]
## [1,] 0.021706698  0.014296175  0.012479795
## [2,] 0.014296175  0.027381646 -0.013613940
## [3,] 0.012479795 -0.013613940  0.150520896
interval <- pnorm(0.975)*sqrt(diag(solve(I)/n)) # Calculate Confidence Interval
CI <- matrix(c(theta-interval,theta+interval),3,2,dimnames =list(c("mu1","mu2","sigma"),c("CI-L","CI-U"
```

Table 3: Louis' Method: Var-Cov Matrix and Confidence intervals

	mu1	mu2	Sigma	CI-L	CI-U
mu1	86.4	-50.97	-11.77	115.4	118.9
mu2	-50.97	68.3	10.4	108.1	111.2
Sigma	-11.77	10.4	8.561	15.15	16.25

We get the point-wise 95% confidence intervals for the parameters by Louis' method.

iii) Compute the observed information matrix directly and obtain its inverse. Find out 95% confidence intervals for the parameters.

$$r_i = \frac{p_i\phi(u_i)}{p_i\phi(u_i) + (1-p_i)\phi(v_i)}$$

$$\frac{\partial}{\partial\theta}r_i = \frac{[p_i\phi(u_i)+(1-p_i)\phi(v_i)]p_i\phi(u_i)\frac{u_i}{\sigma}\begin{bmatrix}0\\1\\u_i\end{bmatrix} - p_i\phi(u_i)\left(p_i\phi(u_i)\frac{u_i}{\sigma}\begin{bmatrix}0\\1\\u_i\end{bmatrix} + (1-p_i)\phi(v_i)\frac{v_i}{\sigma}\begin{bmatrix}1\\0\\v_i\end{bmatrix}\right)}{[p_i\phi(u_i)+(1-p_i)\phi(v_i)]^2} = r_i[1-r_i]\frac{1}{\sigma}\begin{bmatrix}-v_i\\u_i\\u_i^2-v_i^2\end{bmatrix}$$

$$\log(Pr(\vec{\theta}|\vec{y})) = \sum_{i=1}^{n}\log\left(\frac{1-p_i}{\sigma}\phi(v_i) + \frac{p_i}{\sigma}\phi(u_i)\right) = C - n\log(\sigma) + \sum_{i=1}^{n}\log\left((1-p_i)\phi(v_i) + p_i\phi(u_i)\right)$$

$$\frac{\partial}{\partial\theta}\log(p(\vec{\theta}|y)) = \frac{1}{\sigma}\sum_{i=1}^{n}\begin{bmatrix}\frac{(1-p_i)\phi(v_i)}{p_i\phi(u_i)+(1-p_i)\phi(v_i)}v_i\\ \frac{p_i\phi(u_i)}{p_i\phi(u_i)+(1-p_i)\phi(v_i)}u_i\\ \frac{p_i\phi(u_i)u_i^2+(1-p_j)\phi(v_i)v_i^2}{p_i\phi(u_i)+(1-p_i)\phi(v_i)}-1\end{bmatrix} = \frac{1}{\sigma}\sum_{i=1}^{n}\begin{bmatrix}(1-r_i)v_i\\ r_iu_i\\ r_iu_i^2+(1-r_i)v_i^2-1\end{bmatrix}$$

$$\frac{\partial^2\log(p(\vec{\theta}|y))}{\partial\theta^2} = \frac{1}{\sigma^2}\sum_{i=1}^{n}\begin{bmatrix}(1-r_i)(r_iv_i^2-1) & -r_i(1-r_i)u_iv_i & -(1-r_i)v_i[r_i(u_i^2-v_i^2)+2]\\ -r_i(1-r_i)u_iv_i & r_i((1-r_i)u_i^2-1) & r_iu_i[(1-r_i)(u_i^2-v_i^2)-2]\\ -(1-r_i)v_i[r_i(u_i^2-v_i^2)+2] & r_iu_i[(1-r_i)(u_i^2-v_i^2)-2] & r_i(1-r_i)(u_i^2-v_i^2)^2-3(r_iu_i^2+(1-r_i)v_i^2)+1\end{bmatrix}$$

```
(I_o <- matrix(c(sum((1-r)*(r*v^2-1)), sum(-r*(1-r)*u*v)  , sum(-(1-r)*v*(r*(u^2-v^2)+2)) ,
            sum(-r*(1-r)*u*v),       sum(r*((1-r)*u^2-1)) ,  sum(r*u*((1-r)*(u^2-v^2)-2)) ,
sum(-(1-r)*v*(r*(u^2-v^2)+2)),sum(r*u*((1-r)*(u^2-v^2)-2)),sum(r*(1-r)*(u^2-v^2)^2-3*(r*u^2+(1-r)*v^2)+
),nrow = 3, ncol = 3)/(sigma^2)) # Observed exact data
##              [,1]         [,2]         [,3]
## [1,] -0.020806019 -0.014809553 -0.014503680
## [2,] -0.014809553 -0.027282100  0.015255249
## [3,] -0.014503680  0.015255249 -0.148126585
```

```
interval <- pnorm(0.975)*sqrt(diag(solve(-I_o))) # Calculate Confidence Interval
CI_o <- matrix(c(theta-interval,theta+interval),3,2,dimnames =list(c("mu1","mu2","sigma"),c("CI-L","CI-
```

Table 4: Direct Method: Var-Cov Matrix and Confidence intervals which gives the 95% confidence intervals for μ_1, μ_2, and σ. Two methods give same result.

	mu1	mu2	Sigma	CI-L	CI-U
mu1	110.5	-70.08	-18.04	108.4	125.9
mu2	-70.08	83.33	15.44	102	117.3
Sigma	-18.04	15.44	10.11	13.04	18.35

$$\log(p(\vec{\theta}|\vec{y})) = \int_z \log(Pr(\vec{\theta}|\vec{y},\vec{z}))\cdot Pr(\vec{z}|\vec{y},\vec{\theta})dz$$

$$= \sum_{i=1}^{m}\left[Pr(z=1|y_i,\vec{\theta})\log Pr(y_i|z=1,\vec{\theta})\right] + \sum_{j=m+1}^{n}\left[Pr(z=0|y_j,\vec{\theta})\log Pr(y_j|z=0,\vec{\theta})\right]$$

$$= C - n\log(\sigma) + \sum_{i=1}^{m} p_i \log[\phi(u_i)] + \sum_{j=m+1}^{n}(1-p_j)\log[\phi(v_j)]$$

$$\frac{\partial}{\partial\theta}\log(p(\vec{\theta}|y)) = \frac{1}{\sigma}\begin{bmatrix} \sum_{j=m+1}^{n}(1-p_j)v_j \\ \sum_{i=1}^{m} p_i u_i \\ \sum_{i=1}^{m} p_i u_i^2 + \sum_{j=m+1}^{n}(1-p_j)v_j^2 - n \end{bmatrix}$$

$$\frac{\partial^2}{\partial\theta^2}\log(p(\vec{\theta}|y)) = \frac{-1}{\sigma^2}\begin{bmatrix} \sum_{j=m+1}^{n}(1-p_j) & 0 & 2\sum_{j=m+1}^{n}(1-p_j)v_j \\ 0 & \sum_{i=1}^{m} p_i & 2\sum_{i=1}^{m} p_i u_i \\ 2\sum_{j=m+1}^{n}(1-p_j)v_j & 2\sum_{i=1}^{m} p_i u_i & 3(\sum_{i=1}^{m} p_i u_i^2 + \sum_{j=m+1}^{n}(1-p_j)v_j^2) - n \end{bmatrix}$$

```
(I_o <- matrix(c(sum(1-V[,2]),     0          , 2*sum(v*(1-V[,2])),
                            0 , sum(U[,2])     , 2*sum(u*U[,2]),
           2*sum(v*(1-V[,2])), 2*sum(u*U[,2]) , 3*(sum(u^2*U[,2])+sum(v^2*(1-V[,2])))-n
),nrow = 3, ncol = 3)/(sigma^2)) # Observed exact data
##              [,1]        [,2]         [,3]
## [1,]  0.026251449 0.000000000 -0.016886478
## [2,]  0.000000000 0.030390008  0.016122734
## [3,] -0.016886478 0.016122734  0.107273448
```

```
interval <- pnorm(0.975)*sqrt(diag(solve(I_o))) # Calculate Confidence Interval
CI_o <- matrix(c(theta-interval,theta+interval),3,2,dimnames =list(c("mu1","mu2","sigma"),c("CI-L","CI-
```

Table 5: Direct Method: Var-Cov Matrix and Confidence intervals

	mu1	mu2	Sigma	CI-L	CI-U
mu1	42.8	-3.884	7.322	111.7	122.6
mu2	-3.884	36.11	-6.038	104.6	114.7
Sigma	7.322	-6.038	11.38	12.88	18.52

HW4. Due Mar. 04.

1. Suppose (X1,...,Xk) follows a multinomial distribution with parameters (n,p1,...,pk). We wish to simulate this distribution by Gibbs sampling. What are the conditional distribution of any Xj given the rest of the variables? Does Gibbs sampling work in this case? Justify your answer.

$$
\begin{aligned}
f(x_{1:k}) &= \frac{n!}{x_1!x_2!\cdots x_k!}p_1^{x_1}p_2^{x_2}\cdots p_k^{x_k}\\
&= \left(\frac{n!}{x_1!\cdot x_2!\cdots x_{(j-1)}!\cdot x_{(j+1)}!\cdots x_k!}p_1^{x_1}\cdot p_2^{x_2}\cdots p_{(j-1)}^{x_{(j-1)}}\cdot p_{(j+1)}^{x_{(j+1)}}\cdots p_k^{x_k}\right)\cdot\frac{p_j^{x_j}}{x_j!} \qquad \text{or}
\end{aligned}
$$

$$
Pr(p_j|p_{i\neq j},x_{1:k}) = \frac{Pr(p_{1:k},x_{1:k})}{Pr(p_{1:(j-1)},p_{(j+1):k},x_{1:k})} = \frac{\frac{n!}{\prod_{i=1}^k x_i!}\prod_{i=1}^k p_i^{x_i}}{\frac{(n-1)!}{\prod_{i=1}^{j-1}x_i!\prod_{i=j+1}^k x_i!}\prod_{i=1}^{j-1}p_i^{x_i}\prod_{i=j+1}^k p_i^{x_i}} = \frac{np_j^{x_j}}{x_j!}
$$

Given the rest of the variables, $x_j = n-\sum_{i\neq j}x_i$ is a constant, all the variable values are known. It doesn't has a strictly positive density on E_j (Asymptotic Behavior of Gibbs Sampler). After estimating the parameter values, the variable values would not change anymore. Thus, Gibbs sampling doesn't work in this case.

I simulate a multinomial data ($n=50, p_1=0.1,, p_1=0.2, p_1=0.3, p_1=0.4$) data set. $x_{1:4}=(9,27,28,36)$ and put it into a Gibbs sampler. The result shows the estimated parameters will not change with iterations. It confirm that Gibbs sampling doesn't work for this case.

2. Reconsider the example with the pump failures at the nuclear plants. Write your own program to carry out the Gibbs sampling algorithm for the double gamma prior, and report the posterior estimates and 95% Bayesian percentile confidence intervals. Discuss the simulation in detail.

Let d_i denote the failures for the i^{th} pump, $d_i|\lambda_i,t_i\sim Poisson(\lambda_i t_i)$.

And $\lambda_i|\alpha,\beta\sim Gamma(\alpha=1.802,\beta)$; $\beta\sim Gamma(\gamma=0.01,\delta=1)$.

$\theta=(\lambda_1,..,\lambda_{10},\alpha=1.802,\beta,\gamma=0.01,\delta=1)$. The Posterior Distribution is

$$
\begin{aligned}
\pi(\lambda_{1:n},\alpha,\beta,\gamma,\delta|d_{1:n},t_{1:n}) =& Pr(d_{1:n},t_{1:n}|\lambda_{1:n},\alpha,\beta)\cdot Pr(\lambda_{1:n},\alpha,\beta) = Pr(d_{1:n},t_{1:n}|\lambda_{1:n})\cdot Pr(\lambda_{1:n}|\alpha,\beta,\gamma,\delta)\cdot Pr(\alpha,\beta|\gamma,\delta)\\
=& Pr(d_{1:n},t_{1:n}|\lambda_{1:n})\cdot Pr(\lambda_1|\alpha,\beta,\gamma,\delta)\cdots Pr(\lambda_n|\alpha,\beta,\gamma,\delta)\cdot Pr(\beta|\gamma,\delta)\\
=& \left(\prod_{i=1}^n\frac{(\lambda_i t_i)^{d_i}\exp(-\lambda_i t_i)}{d_i!}\right)\left(\prod_{i=1}^n\frac{\beta^\alpha}{\Gamma(\alpha)}\lambda_i^{\alpha-1}\exp(-\beta\lambda_i)\mathbf{1}_{\theta>0}\right)\beta^{\gamma-1}\exp(-\delta\beta)\\
=& \left(\frac{t_i^{d_i}}{\Gamma^n(\alpha)d_i!}\cdot\prod_{i=1}^n\lambda_i^{d_i+\alpha-1}\exp[-\lambda_i t_i]\right)\beta^{n\alpha+\gamma-1}\exp[-(\delta+\sum\lambda_i)\beta] \qquad (1)\\
=& \left(\frac{t_i^{d_i}}{\Gamma^n(\alpha)d_i!}\beta^{n\alpha+\gamma-1}\cdot\prod_{i\neq j}\lambda_i^{d_i+\alpha-1}\exp[-(\delta+\sum_{i\neq j}\lambda_i)\beta]\right)\lambda_j^{d_j+\alpha-1}\exp[-(\beta+t_j)\lambda_j] \qquad (2)
\end{aligned}
$$

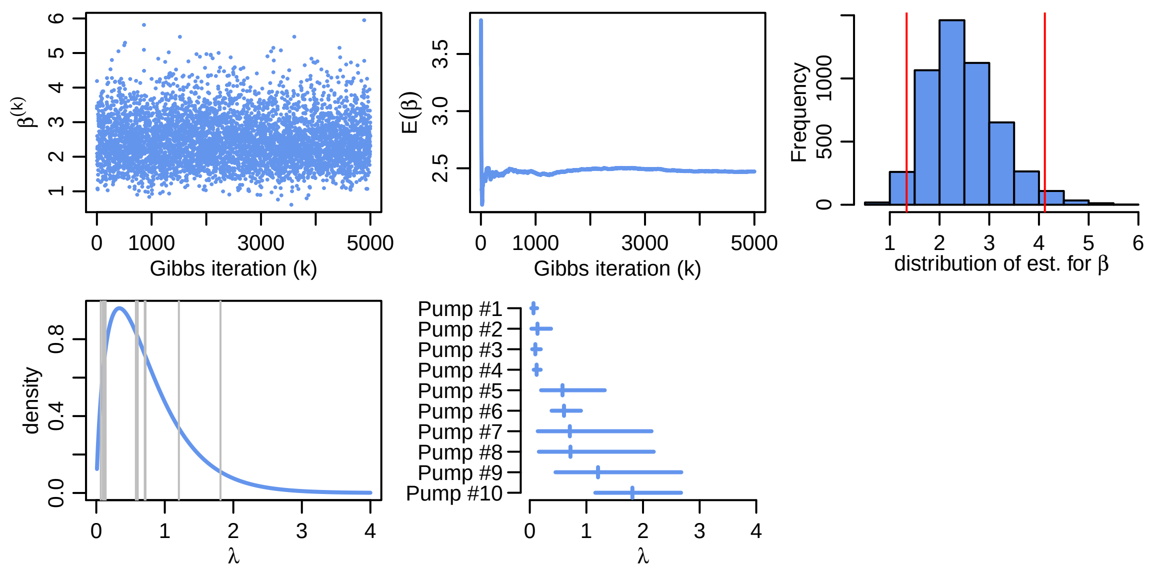
$$(1)\implies \beta|\lambda_{1:10},\alpha\sim Gamma(n\alpha+\gamma,\delta+\sum\lambda_i)$$

$$(2)\implies (\lambda_j|\Lambda_1=\lambda_1,..,\Lambda_{j-1}=\lambda_{j-1},\Lambda_{j+1}=\lambda_{j+1},..,\lambda_{10},B=\beta)\sim Gamma(\alpha+d_j,\beta+t_j)$$

```r
pump <- matrix(c(5, 94.320,1, 15.720,5, 62.880,14, 125.760,3, 5.240,
                 19, 31.440,1, 1.048,1, 1.048,4, 2.096,22, 10.480),2,10)
pump <-t(pump)
colnames(pump) <- c("Failures", "Times")
d <- pump[,1]
t <- pump[,2]
n <- length(d)
```

```r
alpha<-1.802; gamma <- 0.01; delta<- 1 #def hyperparameters
beta <- gamma/delta #initialize lambda and beta
lambda <- d/t
S <- 10000; set.seed(121)
gibbs.mat <- matrix(NA,ncol=(n+1),nrow=S) #Gibbs Sampelr variables
for(k in 1:S){  #Gibbs Sampler
for(j in 1:n){
lambda[j]<- rgamma(1,shape=(d[j]+alpha),rate=(beta+t[j]))  # sample lambda
}
beta <-  rgamma(1,shape=(gamma+n*alpha),rate=(delta+sum(lambda)))  # sample beta
gibbs.mat[k,] <- c(lambda,beta)   # store draws
}
```

	mean	2.5%	50%	97.5%
Lambda1	0.0693	0.0272	0.0658	0.1297
Lambda2	0.1526	0.0291	0.1364	0.3757
Lambda3	0.1033	0.0423	0.0982	0.1946
Lambda4	0.1238	0.0699	0.1211	0.1934
Lambda5	0.6262	0.1996	0.5779	1.324
Lambda6	0.6162	0.3857	0.6046	0.9039
Lambda7	0.8242	0.1408	0.7075	2.151
Lambda8	0.8355	0.1601	0.7182	2.19
Lambda9	1.302	0.4543	1.205	2.677
Lambda10	1.841	1.157	1.812	2.671
Beta	2.471	1.337	2.384	4.119



The table gives mean, median, and 95% credible intervals for the individual failure rates.

The figure of Gibbs iterations shows that the estimate will be stable after 1000 iterations and converge after 2000 iterations.

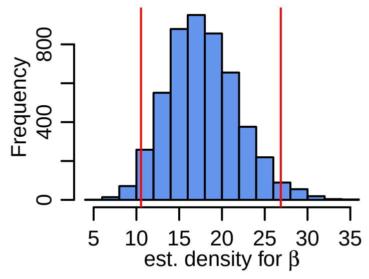
The histogram and density of estimate of β show a Gamma distribution. The same is for $\lambda_{1:10}$.

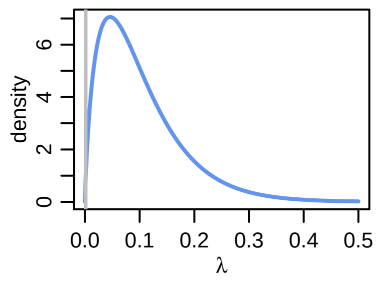
When using thousand hours as the unit of observed times, the range of median of $\lambda_{1:10}$ is 0.0658 - 1.812 and they evenly locate in the density plot. From the box-plot, pump 1-4 have a similar performance, which have significant difference with pump 5-10. The 95% Bayesian percentile confidence intervals of pump 5-10 are overlapping except between pump 6 and 10.

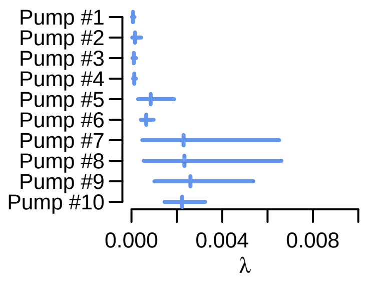
When using hours as the unit of observed times, the range of median of $\lambda_{1:10}$ is 0.000067 - 0.002602. It is reasonable because $\lambda_i t_i$ will not change for the observation. The shape of density of λ keep the same but be scaled a lot. The shape of density of β changed with $\sum \lambda_i$ in the Gibbs sampling. An important change is that the medians of estimate of λ locate in the extremely low region in this case. Therefore, Choosing thousand hours as the unit of observed times is more reliable.

	mean	2.5%	50%	97.5%
Lambda1	0.000071	0.000028	0.000067	0.000133
Lambda2	0.000176	0.000034	0.000157	0.000426
Lambda3	0.000107	0.000044	0.000102	0.000202
Lambda4	0.000126	0.000071	0.000123	0.000197
Lambda5	0.000911	0.000293	0.000846	0.001882
Lambda6	0.000664	0.000415	0.000653	0.000978
Lambda7	0.002609	0.000477	0.002299	0.00651
Lambda8	0.002654	0.000539	0.002335	0.006615
Lambda9	0.002746	0.001011	0.002602	0.005377
Lambda10	0.002266	0.001459	0.002236	0.003248
Beta	17.81	10.54	17.5	26.87

The table gives mean, median, and 95% credible intervals for the individual failure rates.







HW4 Extra

```r
rm(list =ls() )
## generate data with 100 sample size
n_i <- c(10,10,10,10,20,20,20,30,30,30)
a <- length(n_i)
N <- sum(n_i)
sigma_a<-0.1 # give the ture parameter values
sigma_e<-0.3
beta<-2
set.seed(123)
I_a<-diag(a)
I_N<-diag(N)
SIGMA_a_sq<-(sigma_a^2)*I_a
SIGMA_e_sq<-(sigma_e^2)*I_N
Alpha<-MASS::mvrnorm(n = 1, rep(0,a), SIGMA_a_sq) # generate alpha
Epsilon<-MASS::mvrnorm(n = 1, rep(0,N), SIGMA_e_sq) # generate epsilon
Z<-matrix(0,nrow=N,ncol=a)
for (i in 1:a){
Z[(sum(n_i[1:i])-n_i[i]+1):sum(n_i[1:i]),i] <-1 # Z%*%t(Z)
}
N1<-rep(1,N)
Y<-N1*beta+Z%*%Alpha+Epsilon # generate Y
sigma_a<-sigma_e<-beta<-NULL
```

```r
## find MLE
EM_oneway<- function(Y, n_i, crit, itera)
{
i <- 0 # iteration counter
a <- length(n_i)
N <- sum(n_i)
beta <- y_bar <- mean(Y)
Y_ibar <- rep(NA,a)
for (i in 1:a){
Y_ibar[i] <- mean(Y[(sum(n_i[1:i])-n_i[i]+1):sum(n_i[1:i])])
}

sigma_a_sq <- sum((Y_ibar-y_bar)^2)/(a-1)
sigma_e_sq <- sum((Y-Y_ibar)^2)/(N-a)
theta <- c(sigma_a_sq,sigma_e_sq,beta)
repeat {
tau<-theta[1]/theta[2]
sigma_a_sq <- (tau^2*sum(((Y_ibar-theta[3])/(tau+1/n_i))^2)-
               theta[1]*sum(tau/(tau+1/n_i))+
               a*theta[1])/a
sigma_e_sq <- ( -tau*sum(((Y_ibar-theta[3])/(tau+1/n_i))^2*(2+n_i*tau))+
               theta[2]*sum(tau/(tau+1/n_i))+
               sum((Y-theta[3])^2))/N
beta       <- sum((Y_ibar+n_i*tau*theta[3])/(tau+1/n_i))/N
theta_new <- c(sigma_a_sq,sigma_e_sq,beta) # new estimate of parameters
if((abs(theta[1]-theta_new[1]) < crit)| (i > itera)) # (sqrt(sum((theta-theta_new)^2))
break
else {
```

```r
i <- i +1
theta <- theta_new
}
}
return(c(sqrt(theta[1:2]),theta[3],i))
}
EM_oneway(Y, n_i, crit=1e-8, itera = 50)
```

```
## [1]  0.082789265  0.285368749  2.008143535 35.000000000
```

$\hat{\sigma}_\alpha = 0.082789227 \approx 0.1$, $\hat{\sigma}_\varepsilon = 0.285368751 \approx 0.3$, and $\hat{\beta} = 2.008143570 \approx 2$.

After 35 iterations, the results are very close to the true parameter values.

HW5. Due March 11.

Do the Genetic Linkage example in page 177 of the textbook. Write your own code using candidate values sampled from either unif(0,1) or normal distribution centered at the current value with standard deviation of your choice, and analyze your result. Draw a density plot of the θ values.

Algorithm 1: The Metropolis algorithm

```
Data: $Y \sim p(\theta)$
Result: generates $\theta_k^{(1)}, ..., \theta_k^{(s)} \sim$ iid $p(\theta|y)$
Initialization;
Choose $\sigma$ to make the approximation algorithm run efficiently;
for the number of chains $k \leftarrow 1$ to $K$ do
    for a chain $s \leftarrow 1$ to $S$ do
        1.Select a symmetric $q(\theta_b|\theta_a) = q(\theta_a|\theta_b)$ ;
        Let $q(\theta^\star|\theta^{(s)}) = g(\theta^\star) = 1$ ;
        2.Sample $\theta^\star \sim$ uniform$(0,1)$ or normal$(\theta^{(s)}, \sigma^2)$ ;
        3.Compute the acceptance ratio ;
            $$r = \frac{\pi(\theta^\star|y)g(\theta^{(s)})}{\pi(\theta^{(s)}|y)g(\theta^\star)} = \frac{\pi(\theta^\star|y)}{\pi(\theta^{(s)}|y)}$$
        3. Sampling $u \sim$ uniform$(0,1)$;
        if the ratio $r > 1, (u)$ then
            $\theta^{(s+1)} \longleftarrow \theta^\star$ with probability $\min(r,1)$;
        if the ratio $r < 1, (u)$ then
            $\theta^{(s+1)} \longleftarrow \theta^{(s)}$ with probability $1-\min(r,1)$;
        generates a value $\theta^{(s+1)}$ given $\theta^{(s)}$;
```

The Metropolis algorithm works with the observed posterior:

$$\pi(\theta) = (2+\theta)^{125}(1-\theta)^{38}\theta^{34} \propto p(\theta|Y), \quad \theta \in [0,1]$$

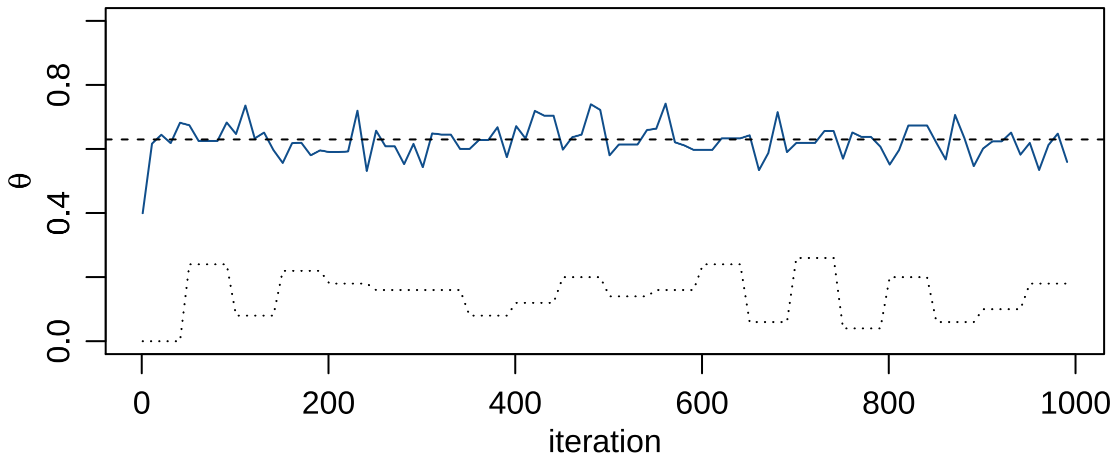
The starting value of $\theta = 0.1$. For this example, we can set $q(\theta^\star|\theta^{(s)}) = 1$. That is, candidate values were generated from Uniform(0,1).

When using the uniform to drive the Metropolis algorithm, it doesn't need $\theta^{(s)}$ and σ in general case $q(\theta^\star|\theta^{(s)}) = \text{uniform}(\theta^{(s)} - \sigma, \theta^{(s)} + \sigma)$.

Uniform distribution itself covers the whole parameter space. Thus, there isn't starting point to "forget". The acceptance rate for this chain is very low and the chain quickly move into the region where the posterior mass is located.

```r
pi <- function(y,theta){(2+theta)^y[1]*(1-theta)^y[2]*theta^y[3]}
# generate from Uniform
ratio<-function(y,theta,theta.star){pi(y,theta.star)/pi(y,theta)}

Metro_Multi_unif <- function(y,S,theta){
THETA<-matrix(NA,S,2) ; acr <- acs<-0;  set.seed(121)
for(s in 1:S) {
  theta.star<-runif(1)
  r<-ratio(y,theta,theta.star)
  if((runif(1))<r) { theta<-theta.star; acs<-acs+1 }
  if(s%%50==0) {acr <- acs/50; acs<-0}
  THETA[s,]<-c(theta,acr)
}
return(THETA)
}
```

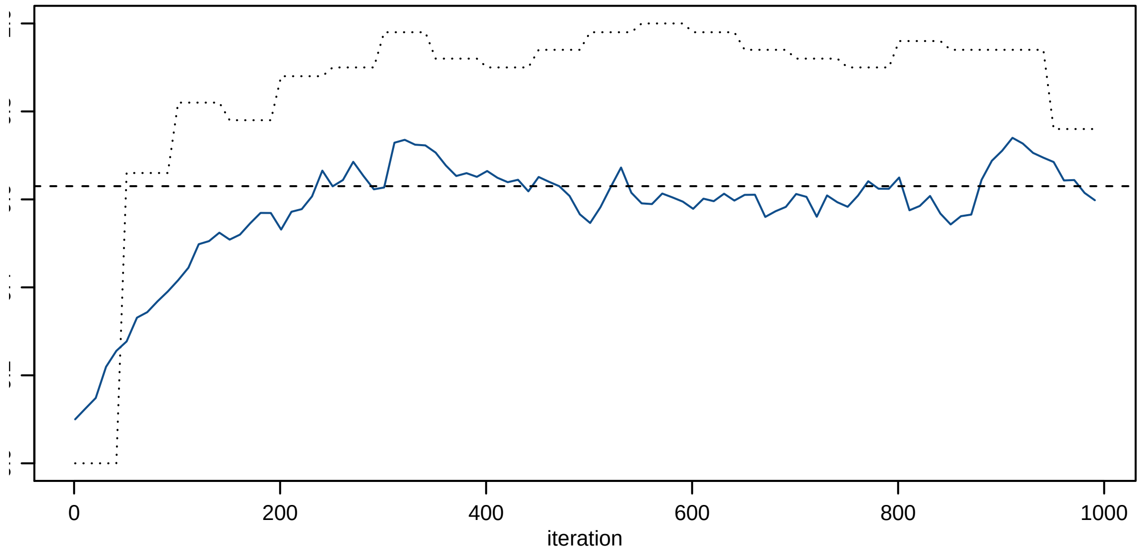


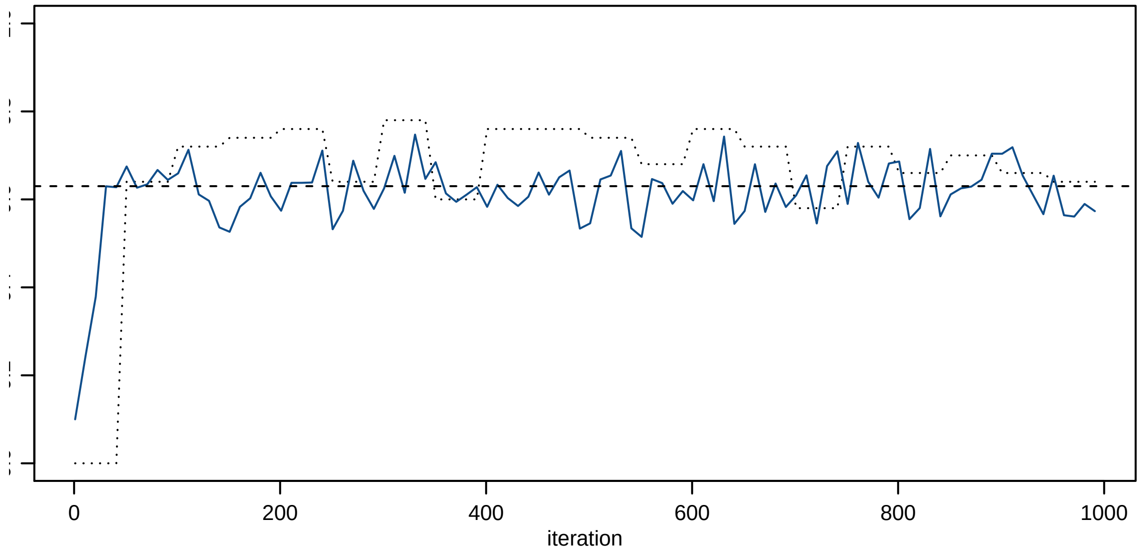
One chain of Metropolis algorithm for the multinomial model from Uniform

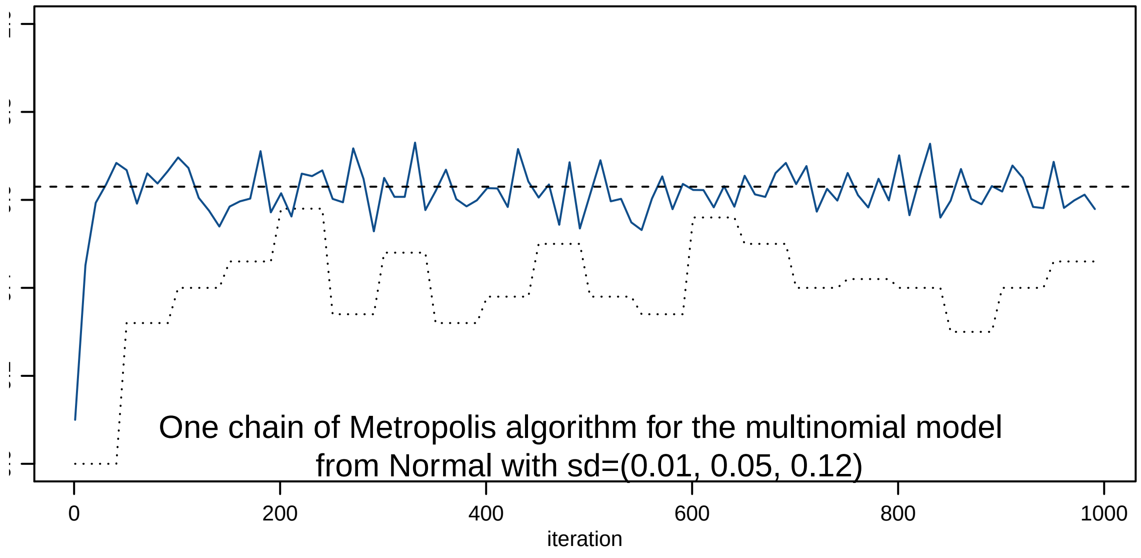
When using the normal to drive the Metropolis algorithm, the candidate values were generated from three symmetric transition functions: Normal $(\theta^\star, \sigma^2)$ with $\sigma = (0.01, 0.05, 0.12)$. It leads to $\theta^\star$ values outside the range [0,1]. The candidate is accepted only when $\theta^\star \leq 1$.

The figures show that the variance of the first driver is grossly under specified, leading to a highly correlated sequence of values that require many iterations to "forget" the starting point and reach the equilibrium distribution. The acceptance rate for this chain tends to be very high, suggesting that the chain is moving up the hill of the posterior density rather than quickly exploring the parameter space. The two other chains quickly move into the region where the posterior mass is located.

```
# generate from Normal
ratio<-function(y,theta,theta.star){pi(y,theta.star)/pi(y,theta)}
Metro_Multi_norm <- function(y,S,theta,sd){
THETA<-matrix(NA,S,2) ; acr <- acs<-0;  set.seed(121)
for(s in 1:S)
{
  theta.star<-rnorm(1,theta,sd)
  r<-ratio(y,theta,theta.star)
  if((runif(1))<r & theta.star<=1) { theta<-theta.star; acs<-acs+1 }
  if(s%%50==0) {acr <- acs/50; acs<-0}
  THETA[s,]<-c(theta,acr)
}
return(THETA)
}
```

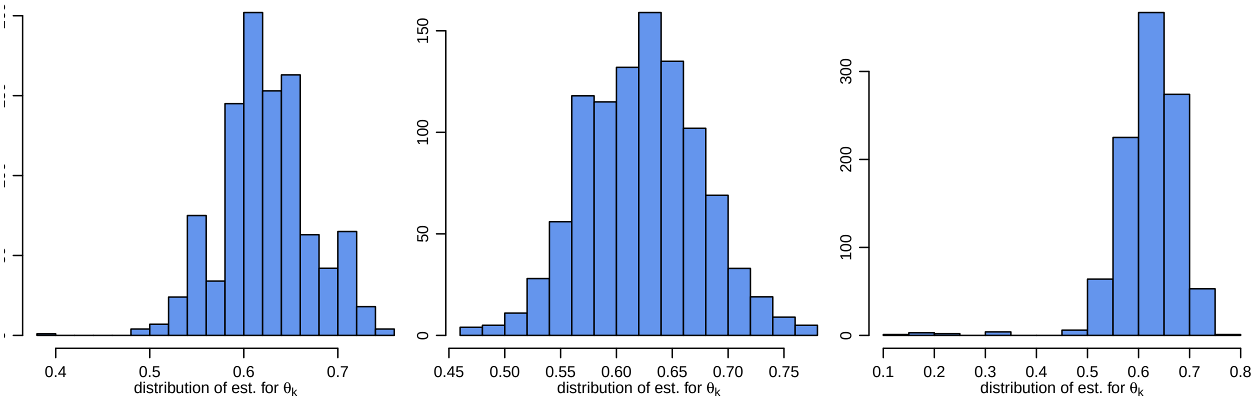

0
200
400
600
800
1000
iteration


0
200
400
600
800
1000
iteration


One chain of Metropolis algorithm for the multinomial model
from Normal with sd=(0.01, 0.05, 0.12)
0
200
400
600
800
1000
iteration

```
THETA_u <- THETA_n<-matrix(NA,S,2); K <- 1000
ptm <- proc.time()
for(k in 1:K) { # Run K Number of chains by drawing from Uniform
THETA_u[k,] <- Metro_Multi_unif(y,S,0.1)[k,]
}
ptm_u <- proc.time() - ptm
ptm <- proc.time()
for(k in 1:K) { # Run K Number of chains by drawing from Normal
THETA_n[k,] <- Metro_Multi_norm(y,S,0.1,0.12)[k,]
}
ptm_n <- proc.time() - ptm
```

```
logpi <- function(theta){125*log(2+theta) + 38*log(1-theta) + 34*log(theta)}
alpha <- function(theta,r) {min(exp(logpi(r)-logpi(theta)),1)}
ptm <- proc.time()
THETA_l<-matrix(NA,S,2)
for(k in 1:K) {
  theta<-runif(1); acs<-0
for(s in 1:S) {
  r <- runif(1)
  i <- rbinom(1,1,alpha(theta,r)) # Using the log of path weight
  theta <- theta + i*(r-theta)
  acs <- acs+i
}
  acr <- acs/S
  THETA_l[k,]<-c(theta,acr)
}
ptm_l <- proc.time() - ptm
```



	mean	2.5%	median	97.5%	95% HPD U	95% HPD L	Acceptance Rate	running time
Uniform	0.6253	0.5346	0.6240	0.7197	0.5316	0.7152	0.1442	13.748
Log_Unif	0.6229	0.5246	0.6240	0.7289	0.5168	0.7164	0.1642	10.932
Normal	0.6217	0.5139	0.6222	0.7172	0.5305	0.7302	0.3974	15.306

In this example, the three setting of the Metropolis Algorithm have similar results with EM Algorithm ($\theta = 0.6268$).

To drive the Metropolis algorithm, drawing θ from Uniform is the most accurate; Using the log of path weight and drawing θ from Uniform is the most efficient; Drawing θ from Normal to drive the Metropolis algorithm is the most Robust.

Final Project

$$\lambda_i \sim Gamma(\alpha + d_i, \tilde{\beta} + t_i)$$

where $\tilde{\beta} = 2.459$ in HW-4

$$\beta \sim Gamma(\gamma + n\alpha, \delta + \sum \lambda)$$

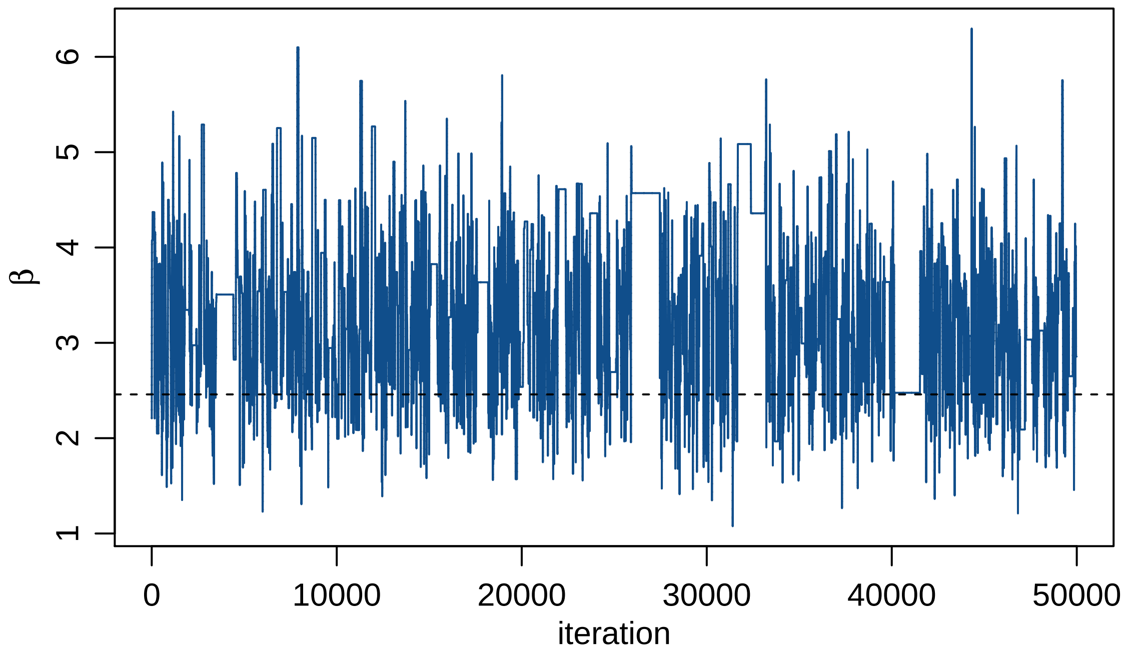
$$\frac{\pi(\lambda', \beta')g(\lambda, \beta)}{\pi(\lambda, \beta)g(\lambda', \beta')} = \prod_{i=1}^{10} \left\{ (\frac{\beta' + t_i}{\beta + t_i})^{\alpha + d_i} (\frac{\sum \lambda + \delta}{\sum \lambda' + \delta})^{\gamma + 10\alpha} \right\} \cdot \exp \left\{ (\beta - \tilde{\beta})\lambda + (\tilde{\beta} - \beta')\lambda' \right\}$$

```
pump <- matrix(c(5, 94.320,1, 15.720,5, 62.880,14, 125.760,3, 5.240,
                 19, 31.440,1, 1.048,1, 1.048,4, 2.096,22, 10.480),2,10)
pump <-t(pump)
colnames(pump) <- c("Failures", "Times")
d <- pump[,1]
t <- pump[,2]
n <- length(d)
```

```
alpha<-1.802; gamma <- 0.01; delta<- 1 #def hyperparameters
beta <- beta0 <- 2.459 #initialize lambda and beta
lambda <- d/t; lambda.star <- rep(NA,n)
S <- 50000; set.seed(121)

ratio<- function(lambda,lambda.star,beta,beta.star){
for(i in 1:n){
lik <- ((beta.star+t[i])/(beta+t[i]))^(alpha+d[i])*((sum(lambda)+delta)/(sum(lambda.star)+delta))^(gamma
expo <- exp((beta-beta0)*lambda+(beta0-beta.star)*lambda.star)
return(prod(lik)*expo)
}}

MH_GG <- function(beta0,S){
THETA<-matrix(NA,S,12) ; acr <- acs<-0;  set.seed(121)
for(s in 1:S)
{
 for(j in 1:n){
lambda.star[j] <- rgamma(1,shape=(d[j]+alpha),rate=(beta0+t[j])) # sample lambda
 }
beta.star <- rgamma(1,shape=(gamma+n*alpha),rate=(delta+sum(lambda.star)))  # sample beta
  r<-ratio(lambda,lambda.star,beta,beta.star)
  if((runif(1))<r) { beta<-beta.star; lambda<-lambda.star;acs<-acs+1 }
  if(s%%50==0) {acr <- acs/50; acs<-0}
  THETA[s,]<-c(beta,lambda,acr)
}
return(THETA)
}
THETA <- MH_GG(beta0,S)
```



the Metropolis−Hasting for the Double Gamma model from Uniform

	mean	2.5%	50%	97.5%
Beta	3.298	1.933	3.128	5.085
Lambda1	0.0677	0.0265	0.0621	0.1303
Lambda2	0.1166	0.0228	0.0995	0.2946
Lambda3	0.0971	0.037	0.0909	0.1873
Lambda4	0.1195	0.0699	0.1179	0.1839
Lambda5	0.445	0.0999	0.413	0.9523
Lambda6	0.5654	0.3622	0.5482	0.8477
Lambda7	0.3994	0.1073	0.3301	1.036
Lambda8	0.4149	0.1148	0.3421	1.083
Lambda9	0.7609	0.2925	0.7436	1.542
Lambda10	1.464	0.9444	1.464	2.139

```
alpha<-1.802; gamma <- 0.01; delta<- 1 #def hyperparameters
beta <- beta0 <- 2.459 #initialize lambda and beta
lambda <- d/t; lambda.star <- rep(NA,n)
K <- S <- 50000; set.seed(121)

ratio<- function(lambda,lambda.star,beta,beta.star){
for(i in 1:n){
lik <- ((beta.star+t[i])/(beta+t[i]))^(alpha+d[i])*((sum(lambda)+delta)/(sum(lambda.star)+delta))^(gamma
expo <- exp((beta-beta0)*lambda+(beta0-beta.star)*lambda.star)
return(prod(lik)*expo)
}}

MH_GG <- function(beta,S){
Theta<-matrix(NA,S,11) ;  set.seed(121)
for(s in 1:S) {
 for(j in 1:n){
```

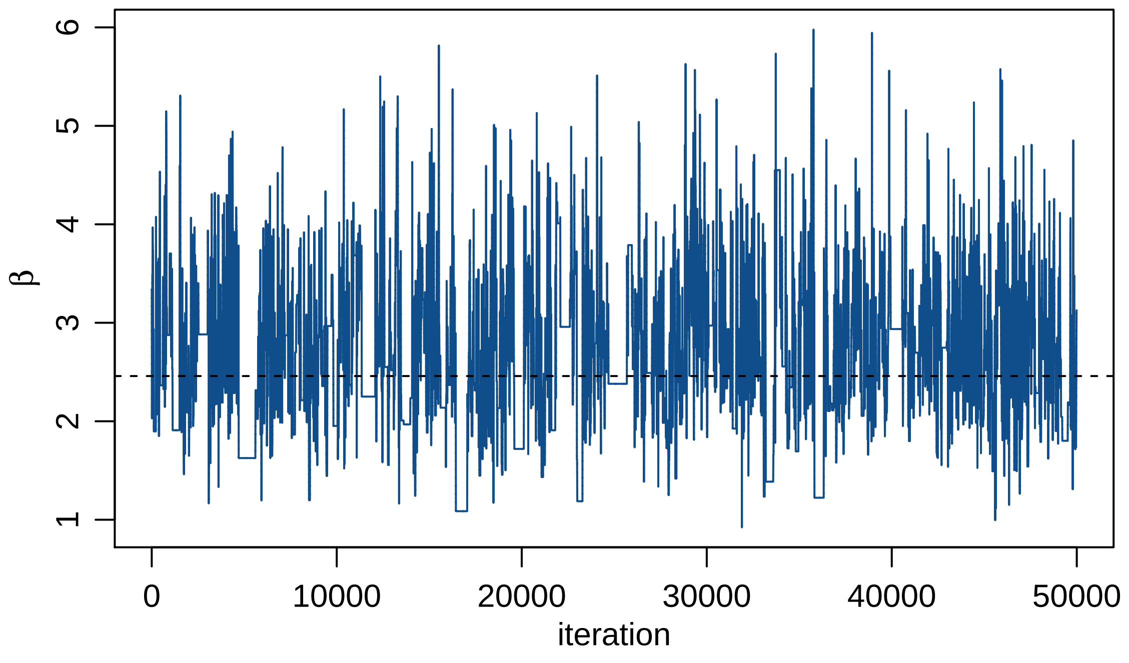
```
lambda.star[j] <- rgamma(1,shape=(d[j]+alpha),rate=(beta+t[j])) # sample lambda
 }

  r<-ratio(lambda,lambda.star,beta,beta)
  if((runif(1))<r) { lambda<-lambda.star }

beta.star <- rgamma(1,shape=(gamma+n*alpha),rate=(delta+sum(lambda.star)))  # sample beta

  r<-ratio(lambda,lambda.star,beta,beta.star)
  if((runif(1))<r) { beta<-beta.star }

  Theta[s,]<-c(beta,lambda)
}
return(Theta)
}
#THETA<-matrix(NA,S,11)
#for(k in 1:K) {
#THETA[k,] <- MH_GG(beta,S)[S,]
#}
THETA <- MH_GG(beta,S)
```



the Metropolis−Hasting for the Double Gamma model from Uniform

	mean	2.5%	50%	97.5%
Beta	2.699	1.25	2.634	4.465
Lambda1	0.0638	0.0286	0.0587	0.1263
Lambda2	0.1258	0.0127	0.1151	0.2906
Lambda3	0.0969	0.0298	0.0949	0.1757
Lambda4	0.1225	0.0715	0.1217	0.1871
Lambda5	0.4096	0.133	0.3914	0.8577
Lambda6	0.5513	0.2982	0.5498	0.7851

	mean	2.5%	50%	97.5%
Lambda7	0.4095	0.0671	0.3685	1.043
Lambda8	0.3899	0.0862	0.3486	0.9824
Lambda9	0.6951	0.2554	0.6446	1.397
Lambda10	1.42	0.9246	1.397	2.064

```
d.lambda[j]<- dgamma(lambda,shape=(d[j]+alpha),rate=(beta+t[j]))  # sample lambda
d.beta <-dgamma(beta,shape=(gamma+n*alpha),rate=(delta+sum(lambda)))

lik_lambda[j] <- lambda[j]^(d[j]+alpha-1)*exp(-lambda[j]*t[j])
beta.star <-  rgamma(1,shape=gamma,rate=delta)  # sample beta
gibbs.mat[k,] <- c(lambda,beta)   # store draws


ratio<-function(d,t,t.star,beta,beta.star){ # Metropolis-Hastings
pi(d,t.star,beta.star)/pi(d,t.star,beta.star)*dnorm(beta.star,0,10)/dnorm(beta,0,10)}

Metro_Multi_norm <- function(y,S,theta,sd,mu,t2){
THETA<-matrix(NA,S,2) ; acr <- acs<-0;  set.seed(121)
for(s in 1:S)
{
  theta.star<-rnorm(1,theta,sd)
  r<-ratio(y,theta,theta.star,mu,t2)
  if((runif(1))<r & theta.star<=1) { theta<-theta.star; acs<-acs+1 }
  if(s%%50==0) {acr <- acs/50; acs<-0}
  THETA[s,]<-c(theta,acr)
}
return(THETA)
}

Metro_Multi_unif <- function(y,S,theta,mu,t2){
THETA<-matrix(NA,S,2) ; acr <- acs<-0;  set.seed(121)
for(s in 1:S)
{
  theta.star<-runif(1)
  r<-ratio(y,theta,theta.star,mu,t2)
  if((runif(1))<r) { theta<-theta.star; acs<-acs+1 }
  if(s%%50==0) {acr <- acs/50; acs<-0}
  THETA[s,]<-c(theta,acr)
}
return(THETA)
}
```

```
par(mfrow=c(3,1),mar=c(5,3,1,1),mgp=c(1.75,.75,0))
for(sd in c(0.01,0.05,0.12) ) { # under three different proposal distributions
THETA <- Metro_Multi_norm(y,S,0.1,sd,0,10)
plot(skeep,THETA[skeep,2],type="l",ylim = c(0,1),xlab="iteration",ylab=expression(theta),lty=3,col=alph
lines(skeep,THETA[skeep,1],col="dodgerblue4" )
abline(h=0.63,lty=2)
}
mtext("the Metropolis algorithm for the multinomial model \n from Normal with sd=(0.01, 0.05, 0.12)", s
```

(a) Introduction, main findings, discussion, and references

(b) Comparison between this and the Gibbs sampler analysis

(c) Convergence rate comparison between close and far away transition probability kernels

(d) Other properties of the posterior distribution such as median, mode, or quantiles?

(e) What challenges did you encounter?

(f) Any suggestion for an appropriate q(.)?

Algorithm 2: The Metropolis Hastings algorithm

```
Data: $Y \sim p(\theta)$
Result: generates $\theta^{(1)}, ..., \theta^{(s)} \sim$ iid $p(\theta|y)$
initialization;
Choose $\sigma$ to make the approximation algorithm run efficiently;
while not convergentcy do
    for Symmetric: $J(\theta_b|\theta_a) = J(\theta_a|\theta_b)$ do
        1. Sample $\theta^\star \sim J(\theta|\theta^{(s)})$; such as ;
               $$J(\theta^\star|\theta^{(s)}) = \text{normal}(\theta^{(s)}, \delta^2)$$
         OR
               $$J(\theta^\star|\theta^{(s)}) = \text{uniform}(0, 1)$$
         2. Compute the acceptance ratio
               $$r = \frac{\pi(y|\theta^\star)q(\theta^\star)}{\pi(y|\theta^{(s)})q(\theta^{(s)})}$$
         3. Sampling $u \sim \text{uniform}(0, 1)$;
        if the ratio $r > 1, (u)$ then
            $\theta^{(s+1)} \longleftarrow \theta^\star$ with probability $\min(r, 1)$;
        if the ratio $r < 1, (u)$ then
            $\theta^{(s+1)} \longleftarrow \theta^{(s)}$ with probability $1 - \min(r, 1)$;
        generates a value $\theta^{(s+1)}$ given $\theta^{(s)}$;
```

Note: the normalizing constant for $p(\theta|Y)$ is not required for implementing the Metropolis algorithm.

In such instances, we recall that $\pi(\theta) = 0$ for values of θ outside the interval $[0, 1]$.

One long chain, starting from $\theta = 0.1$, was created for each driver, and the results are presented in Figures 6.18, 6.19 and 6.20.

In each figure, the solid line represents the path of the chain, while the dotted/dashed line represents the proportion of accepted jumps for the Metropolis algorithm (computed over contiguous blocks of 50 iterations).

The horizontal bar is located at $Y = 0.63$, the posterior mean.

Note that the variance of the first driver is grossly underspecified, leading to a highly correlated sequence of values that require many iterations to “forget” the starting point and reach the equilibrium distribution.

The acceptance rate for this chain tends to be quite high, suggesting that the chain is moving up the hill of the posterior density rather than quickly exploring the parameter space.

The two other chains quickly move into the region where the posterior mass is located.

Gelman et al. (1995) suggest that the optimal variance on the normal driver is $c^2\Sigma$;, where $c \approx 2.4\sqrt{d}$ (d is the dimension of the parameter vector) and Σ is the variance-covariance matrix (i.e. based on the curvature of the posterior at the mode).

In the present case $\Sigma = 0.052$ and $c = 2.4$, implying that the optimal standard deviation for the normal driver is 0.12.

Gelman et al. (1995) also suggest that the optimal jumping rule has an acceptance rate of about 0.44 for one dimensional problems, decreasing to 0.23 for problems where the dimension of the parameter vector exceeds 5.

For the present example, the driver standard deviation of 0.12 does (approximately) yield the optimal rate.

Backup

```r
# Backup
motorette <-  matrix(
c(1764,2772,3444,3542,3780,4860,5196,408,408,1344,1344,1440,408,408,504,504,504,
  rep(8064,10),rep(5448,3),rep(1680,5),rep(528,5),
  rep(170,7),rep(190,5),rep(220,5),
  rep(150,10),rep(170,3),rep(190,5),rep(220,5)),
nrow=40,
ncol=2,
byrow = F)
```

```r
# Backup

EM<- function(D, n, m, crit = 0.000001, itera = 50)
{
##"D" is the data, which have exact part and right censored part.
##This algorithm computes the MLE of parameters in the simple linear
##regression with right censored data.
##"theta" is the parameter vector, that is, beta0, beta I, and sigma.
##"thetastar" is the current parameter estimate.
##In this special project temperatures and failure times
##had specific transformations. ·
##
##"itera" is the upper limit of iterations. That is, if the algorithm
##doesn't converge until this upper limit, we have to stop at this
##moment and treat our current estimate as the limit point we are
##trying to obtain.
##"n" is the sample size
##"m" is the exact data size.
##########################################################
thetastar <- rep(0,3)
theta<- rep(0,3)
i <- 0 # iteration counter
D[,1] = log10(D[,1]) # transformed time
D[,2] = 1000/(D[,2]+273.2) # transformed temperature
exact<- D[1:m,] # exact data (uncensored data)
censored<- D[(m+1):n,] # censored data
out<- lm(D[,1]~ D[,2])
# the least squares method based on the data where we assume
# the censored are exact
thetastar[1:2] <- out$coefficients
thetastar[3] <- sqrt(sum(out$residuals^2)/(m - 2))
repeat {
```

```r
############ E Step ##############
# Computing Q function
mustar <- thetastar[1] + thetastar[2]*censored[, 2]
sigmastar <- thetastar[3]
zscore <- (censored[,1]-mustar)/sigmastar # standardized score
num <- dnorm(zscore)
denom <- 1-pnorm(zscore)
H <- num/denom # hazard rate function of standard normal dist'n
tstar <- mustar + sigmastar*H # estimated failure times
############ M Step ##############
cmpy <- c( exact[, 1], tstar) # complete data augmented by
# estimated failure times
out<- lm( cmpy~ D[, 2])
theta[1:2] <- out$coefficients # new estimates of reg. coeff.
mu<- theta[1]+ theta[2]*D[, 2] # new mean
S1 <- sum((exact[,1]-mu[1 :m])^2) # exact part
# S2, S3, S4 are for the censored part
S2 <- sum(sigmastar^2+mustar^2+sigmastar*( censored[, 1 ]+mustar-
2*mu[(m+ 1):n])*H)
S3 <- sum(mu[(m+1):n]*mustar)
S4 <- sum(mu[(m+1):n]^2)
rssq <- S1 + S2 - 2*S3 + S4 # total residul sum of squares
theta[3] <- sqrt(rssq/n) # new estimate of residual std
if((sqrt(sum((thetastar-theta)^2)) < crit)| (i > itera))
break
else {
i <- i +1
thetastar <- theta
}
}
return(list(theta,i))
}
EM(motorette,n=40,m=17,crit=0.0001)
```

```
## [[1]]
## [1] -6.01903378  4.31113761  0.25915984
##
## [[2]]
## [1] 24
```

```r
I_u <- c(11:17,21:25,31:35)
I_c <- c(1:10,18:20,26:30,36:40)
temp <- c(150,170,190,220) #temperature levels
trec <- 1000/(temp+273.2) #reciprocal of the absolute temperature T
nu <- c(rep(trec[1],10),rep(trec[2],10),rep(trec[3],10),rep(trec[4],10))
t<- t_0<-log10(c(                         rep(8064,10),
         1764,2772,3444,3542,3780,4860,5196,rep(5448,3),
                     408,408,1344,1344,1440,rep(1680,5),
                        408,408,504,504,504,rep(528,5)))
# initial value
m <- length(I_u)
n<-length(I_c)+m
w<- unique(t_0[I_c])
```

```
nu_i <- nu[I_c]
nu_j <- nu[I_u]
nu_bar <- mean(nu)

t_i<- t[I_c]
t_j<- t[I_u]

fit0 <- lm(t~nu)
fit_c <- lm(t_i~nu_i)
fit_u <- lm(t_j~nu_j)

sigma <- sigma(fit0) #standard error of residuals
beta0 <- coef(fit0)[1] #intercept
beta1 <- coef(fit0)[2] #slope

mu_j <- beta0 + beta1 * nu_j
SS_nu <- sum(nu^2)-n*nu_bar^2
mu_i <- beta0 + beta1 * nu_i
z_i=(t_i-mu_i)/sigma
H_i <- dnorm(z_i)/(1-pnorm(z_i))
# H_i <-dnorm(0)/(1-pnorm(0))
ET_i  <- mu_i+ sigma*H_i

S <- 60
THETA<-matrix(nrow=S,ncol=8,dimnames=list(NULL,
     c('Iteration','Intercept','Slope','Sigma','mu150','mu170','mu190','mu220')))
THETA[1,]<-theta<-c(1, beta0, beta1,sigma,unique(ET_i))
Q <- 0
k=1
delta = 1e-4
```

```
repeat {
beta0<- THETA[k,2]
beta1<- THETA[k,3]
sigma<- THETA[k,4]

# M step

mu_i <- beta0 + beta1 * nu_i
z_i=(t_i-mu_i)/sigma
H_i <- dnorm(z_i)/(1-pnorm(z_i))
ET_i <- mu_i+ sigma*H_i

beta0_star <- sum(t_j)/n+sum(ET_i)/n-
              nu_bar/SS_nu*(sum(t_j*(nu_j-nu_bar))+sum(ET_i*(nu_i-nu_bar)))

beta1_star <- (sum(t_j*(nu_j-nu_bar))+sum(ET_i*(nu_i-nu_bar)))/SS_nu

mu_i_star <- beta0_star+beta1_star*nu_i
mu_j_star <- beta0_star+beta1_star*nu_j

sigma_star <- sqrt((sum((t_j-mu_j_star)^2)+
                 sum(sigma^2+sigma*H_i*(ET_i+mu_i-2*mu_i_star)+(mu_i-mu_i_star)^2)
```

```
                )/n)

ET_i_star <- mu_i_star+ sigma_star*H_i

# E step
# Get Q
k <-  k+1
Q[k] <-
 -n*log(2*pi)/2-
  n*log(sigma_star)-
 (1/(2*sigma_star^2))*sum((t_j-mu_j_star)^2)-# 1/2sigma^2*sum(j uncensored)-
 (1/(2*sigma_star^2))*sum(                    # 1/2sigma^2*sum(i censored
                      sigma^2+        # sigma^2+
                      sigma*H_i*(     # sigma*H*(
          ET_i+mu_i-2*mu_i_star)+      # w_i+mu_i_star-2mu_i)
             (mu_i-mu_i_star)^2)      # (mu_i-mu_i_star)^2)

# Update THETA
 THETA[k,2] <- beta0_star
 THETA[k,3] <- beta1_star
 THETA[k,4] <- sigma_star
THETA[k,5:8]<-unique(ET_i_star)
THETA[k,1] <- k

 if(abs(Q[k]-Q[k-1])<=delta) break
}
```

Using the EM method, the results of $\hat\beta_0$,$\hat\beta_1$,$\hat\sigma$ are r THETA[max(THETA[,1],na.rm =T),2:4] after 58th iterations. Although the estimates are smaller than SL method, Thay shows a same trend: $\hat\beta_1$ and $\hat\mu$ grow larger and converges.

```
plot(nu,t_0,main="EM Method with full data",xlab="reciprocal of the absolute temperature",ylab="log10 o
     xlim=c(2,2.5),ylim=c(2.5,5),
     panel.first=abline(h=c(3.577492,3.906551),v=c(2.027575,2.158895,2.256318,2.362949),lty=3,col="gray
for (i in (1:max(THETA[,1],na.rm =T)))
abline(THETA[i,2:3],lwd=0.1,lty=1,col=gray(i/max(THETA[,1],na.rm =T)))
abline(fit_c,lwd=0.5,lty=1,col="red")
abline(fit_u,lwd=0.5,lty=1,col="blue")
text(2.48,4.6, labels = "58th");text(2.45,4.35, labels = "1st")
legend(2.2,3.2,legend=c("regression line after iterations","initial censored data","initial uncensored
```

```
kable(THETA[c(1:10,48:58),])%>%footnote(general = "The first and last 10 rows")
EM_full<- THETA[max(THETA[,1],na.rm =T),]
```

http://statisticalrecipes.blogspot.com/2012/03/em-algorithm-and-confidence-intervals.html

Say you have some sample data $X_1, X_2, \dots, X_n$ that are iid and follow some distribution $f(X|\theta)$ (e.g. Binomial(n, p)). Finding the maximum likelihood estimate (MLE) and confidence interval of θ is fairly straightforward. To find the MLEs, just compute the gradient of the log likelihood, set equal to 0 and solve for θ. Let $\hat\theta$ be the MLE of θ and let the Fisher Information matrix of the sample be given by

$$I(\theta) = E_\theta((\frac{\partial}{\partial\theta}\log f(\mathbf{X}|\theta))^2) = -E_\theta(\frac{\partial^2}{\partial\theta^2}\log f(\mathbf{X}|\theta))$$

An important property of MLEs is the distribution of the estimators is asymptotically normal with mean θ and the Var(θ) being approximated by the inverse of the Fisher Information matrix $I(\theta)$.

To calculate a $100(1-\alpha)\%$ confidence interval for θ, compute the Fisher Information matrix from the sample and

$$[\hat{\theta} - Z_{\alpha/2}(\frac{1}{\sqrt{I(\theta)}}), \hat{\theta} + Z_{\alpha/2}(\frac{1}{\sqrt{I(\theta)}})]$$

The confidence intervals calculated above are when your data is complete and does not contain any missing data. When there is missing information, the Expectation-Maximization Algorithm is commonly used to estimate the parameters. There are several good guides out there including one of my favorites (here). A question I recently came across was, how do we calculate the confidence intervals for MLEs of incomplete data out of the EM algorithm? How do we compute the observed information matrix of the incomplete data? Louis (1982), Meilijson (1989), Lange (1995), Oakes (1999) were a few of the references I found on the topic.

First, we must introduce a little notation. Let y be the observed data, z be the missing data and $x = (y, z)$ be the complete data. Define $L(\theta|y)$ as the observed (or incomplete) likelihood and $L_0(\theta|x)$ as the complete likelihood with the full data. In the EM algorithm, we want to maximize $L(\theta|y)$ in θ, but we do this using a conditional expectation

$$Q(\theta|\theta^{(t)}) = E_{X|Y,\theta^{(t)}}[\log L_0(\theta|X)]$$

where $\theta^{(t)}$ is the parameter estimate for θ at the tth iteration. The EM algorithm moves in iterations between two steps:

1) Expectation Step: take the expectation of the complete data X conditional on the observed data y and the current parameter estimates $\theta^{(t)}$.
2) Maximization Step: find the new $\theta^{(t+1)}$ that maximizes $Q(\theta|\theta^{(t)})$. Louis (1982) defines the notation of the gradient and the negative of the second derivatives of the complete likelihood,

$$S(X,\theta) = \frac{\partial \log L_0(\theta|X)}{\partial\theta} \quad \text{and} \qquad B(X,\theta) = -\frac{\partial^2 \log L_0(\theta|X)}{\partial\theta^2}$$

   and the gradient of the observed likelihood

$$S^*(Y,\theta) = \frac{\partial \log L(\theta|Y)}{\partial\theta}$$

where $S^*(y,\theta) = E_{X|Y,\theta}[S(X,\theta)]$ and $S^*(y,\hat{\theta}) = 0$. Then, the observed information matrix of the incomplete data can be obtained using

$$I_Y(\theta) = E_{X|Y,\theta}[B(X,\theta)] - E_{X|Y,\theta}[S(X,\theta)S^T(X,\theta)] + S^*(y,\theta)S^{*T}(y,\theta)$$

or another way to think about it is

$$I_Y = I(\hat{\theta}) = I_X(\theta) - I_{X|Y}$$

The authors note Efron and Hinkley (1978) define I_Y as the observed information and say it is "a more appropriate measure of information than the a priori expectation $E_\theta[B^*(Y,\theta)]$".

Oakes (1999) shows the function $Q(\theta|\theta^{(t)})$ can be used in the maximization of the observed likelihood $L(\theta|y)$. Therefore, when calculating the observed information matrix of the incomplete data, it is sufficient to use

$$I(\theta) = -\frac{\partial^2 Q}{\partial \theta^2}|_{\theta=\hat{\theta}}$$

To calculate a $100(1-\alpha)\%$ confidence interval for θ, we then use the same formula as above

$$\hat{\theta} - Z_{\alpha/2}(\frac{1}{\sqrt{I(\theta)}}),\ \hat{\theta} + Z_{\alpha/2}(\frac{1}{\sqrt{I(\theta)}})]$$