

Stat 590 Statistical Computing Homework 8
Final Project — Data augmentation/EM
application for a censored regression model

Erik Erhardt

May 5, 2005

Tanner p. 67 *EM application for a censored regression model.*

An implementation of the EM algorithm under the normal regression model with right-censored data is given in the appendix. The estimates are the same as what are arrived at in the text,

$$\underline{\beta} = (-6.01925, 4.311247)', \quad \sigma = 0.2591827$$

with 54 iterations. The plot in Figure 1 on page 2 gives the data and the best fit line based on the EM algorithm MLEs.

Table 1 on page 4 provides the parameter iterations. Note that beyond iteration 16 or so, very little improvement occurs in the estimates. The plot in Figure 2 on page 3 gives trace plots of the parameters. Convergence is quick early, then little is gained after 16 iterations.

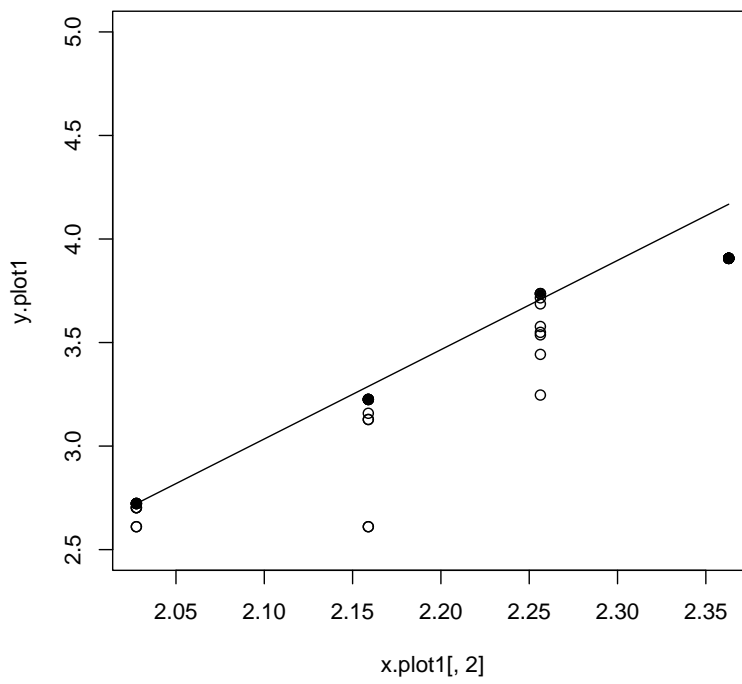


Figure 1: 8.1: Data (censored = solid) and the best fit line based on the EM algorithm MLEs.

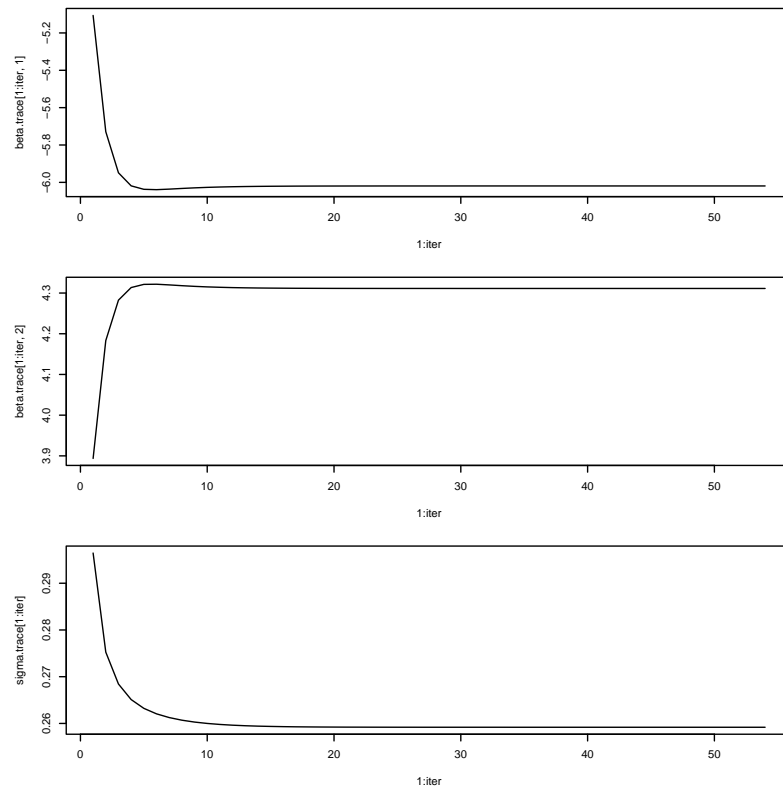


Figure 2: 8.1: Trace plots of the parameters: β_0, β_1, σ .

iter	β_0	β_1	σ	convergence
1	-5.106339	3.893641	0.2964629	1.021581e+01
2	-5.728805	4.183253	0.2752571	9.862945e-01
3	-5.948759	4.282514	0.2684350	3.443119e-01
4	-6.018715	4.313492	0.2651081	1.134050e-01
5	-6.037225	4.321268	0.2632300	3.339566e-02
6	-6.039072	4.321627	0.2620516	6.693658e-03
7	-6.036214	4.319970	0.2612566	7.554061e-03
8	-6.032596	4.318060	0.2606966	7.673758e-03
9	-6.029387	4.316406	0.2602932	6.411866e-03
10	-6.026828	4.315098	0.2599991	4.997370e-03
11	-6.024873	4.314102	0.2597835	3.780773e-03
12	-6.023406	4.313357	0.2596251	2.821445e-03
13	-6.022317	4.312804	0.2595085	2.091798e-03
14	-6.021511	4.312395	0.2594226	1.545883e-03
15	-6.020916	4.312093	0.2593594	1.140614e-03
16	-6.020477	4.311870	0.2593129	8.409039e-04
17	-6.020154	4.311706	0.2592786	6.196804e-04
18	-6.019916	4.311585	0.2592533	4.565498e-04
19	-6.019740	4.311496	0.2592347	3.363190e-04
20	-6.019611	4.311431	0.2592210	2.477314e-04
21	-6.019516	4.311382	0.2592109	1.824694e-04
22	-6.019446	4.311347	0.2592035	1.343957e-04
23	-6.019394	4.311320	0.2591980	9.898561e-05
24	-6.019356	4.311301	0.2591940	7.290420e-05
25	-6.019328	4.311287	0.2591910	5.369437e-05
26	-6.019307	4.311276	0.2591888	3.954594e-05
27	-6.019292	4.311269	0.2591872	2.912546e-05
28	-6.019281	4.311263	0.2591860	2.145074e-05
29	-6.019273	4.311259	0.2591852	1.579830e-05
30	-6.019267	4.311256	0.2591845	1.163530e-05
31	-6.019262	4.311253	0.2591840	8.569282e-06
32	-6.019259	4.311252	0.2591837	6.311182e-06
33	-6.019256	4.311251	0.2591834	4.648112e-06
34	-6.019255	4.311250	0.2591832	3.423279e-06
35	-6.019253	4.311249	0.2591831	2.521203e-06
36	-6.019252	4.311249	0.2591830	1.856834e-06
37	-6.019252	4.311248	0.2591829	1.367535e-06
38	-6.019251	4.311248	0.2591829	1.007172e-06
39	-6.019251	4.311248	0.2591828	7.417690e-07
40	-6.019250	4.311248	0.2591828	5.463032e-07
41	-6.019250	4.311247	0.2591828	4.023452e-07
42	-6.019250	4.311247	0.2591828	2.963220e-07
43	-6.019250	4.311247	0.2591827	2.182372e-07

Table 1: 8.1: Parameter iterations.

Tanner p. 96 *Data Augmentation application for a censored regression model.*

An implementation of the Data Augmentation algorithm under the normal regression model with right-censored data is given in the appendix. The estimates,

$$\underline{\beta} = (-6.181908, 4.397502)', \quad \sigma = 0.2961307$$

with 10000 iterations, are similar to what was given by the EM algorithm, indicating the imputed censored values make little difference to the fit of the regression,

The plot in Figure 3 on page 5 summarizes all the results, giving trace plots and histograms of β_0, β_1, σ , a plot of β_0, β_1 bivariate draws, and the fitted line. These plots are detailed in Figures 4 through 8.

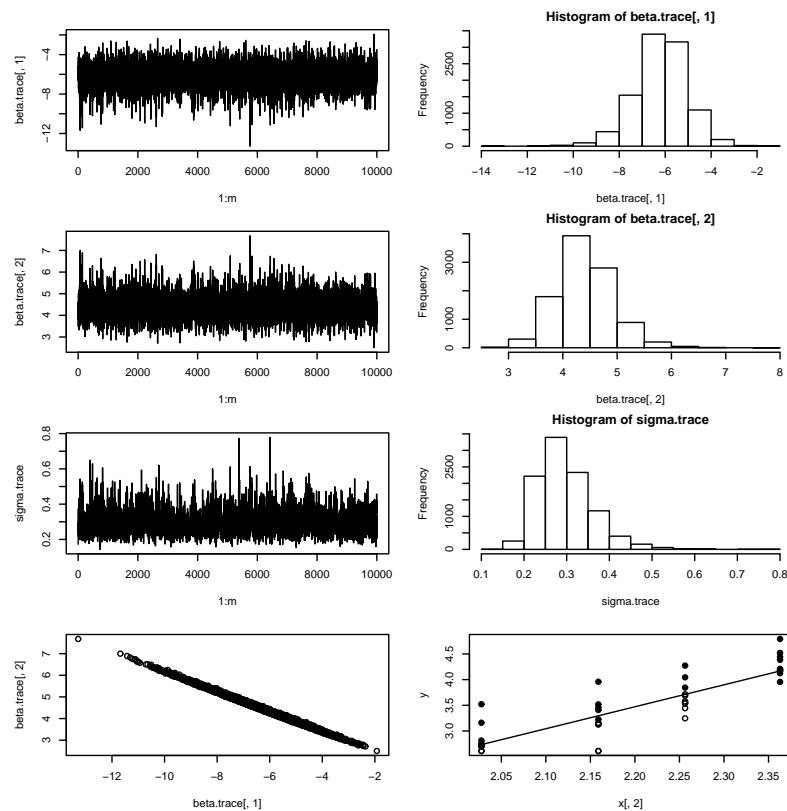


Figure 3: 8.2: Trace plots and histograms of $\beta_0, \beta_1, \sigma, \beta_0, \beta_1$ bivariate draws, and fitted line.

The plot in Figure 4 on page 6 gives the trace plot and histogram for β_0 . The chain appears to be mixing well, and the distribution of β_0 is roughly symmetric.

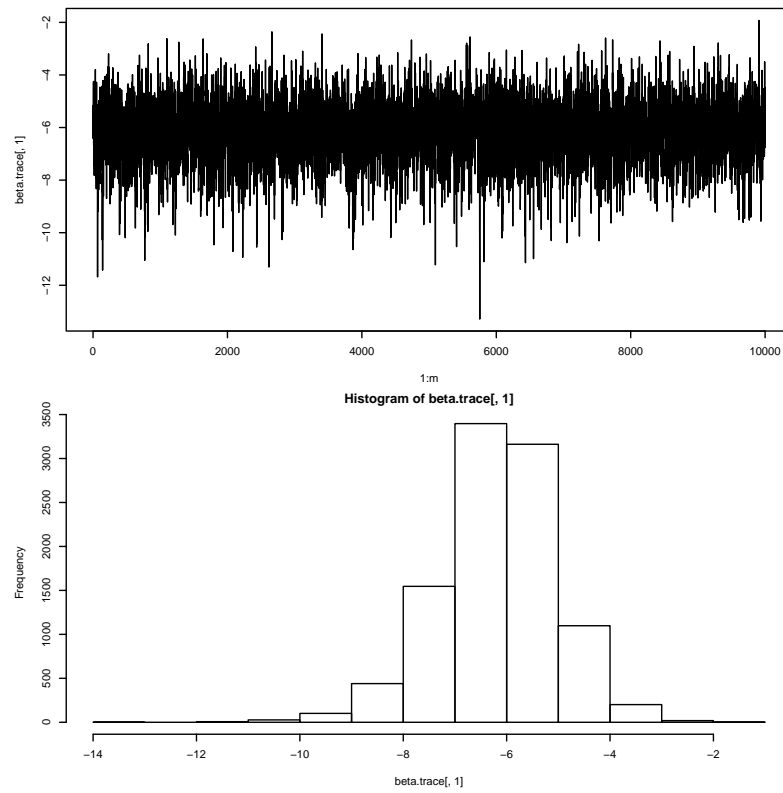


Figure 4: 8.2: Trace plot and histogram for β_0 .

The plot in Figure 5 on page 7 gives the trace plot and histogram for β_1 . The chain appears to be mixing well, and the distribution of β_1 is roughly symmetric.

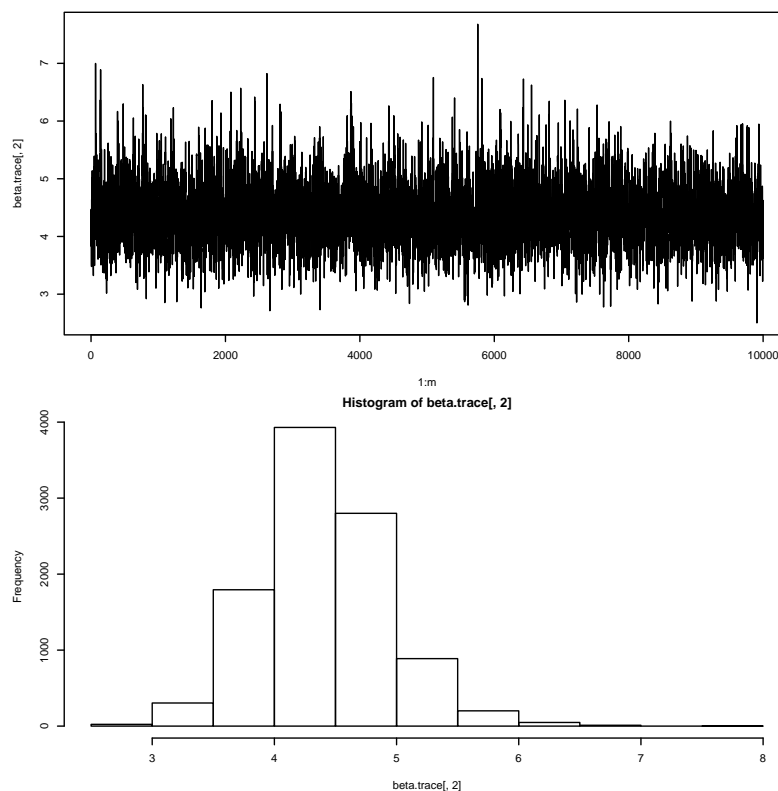
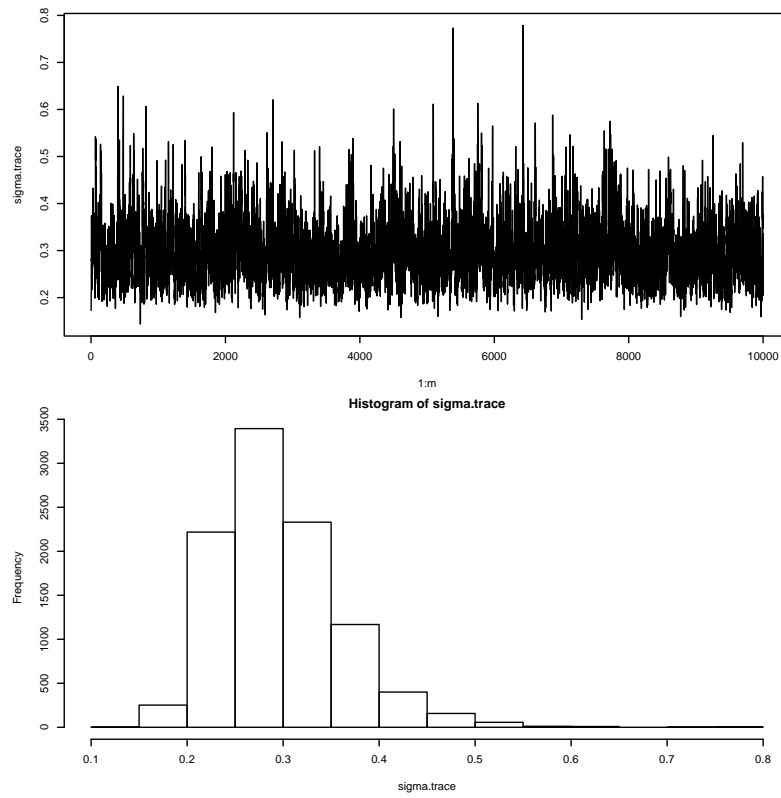


Figure 5: 8.2: Trace plot and histogram for β_1 .

The plot in Figure 6 on page 8 gives the trace plot and histogram for σ . The chain appears to be mixing well, and the distribution of σ is roughly symmetric.

The plot in Figure 7 on page 9 gives the β_0, β_1 bivariate draws. There is very high correlation between these parameters.

The plot in Figure 8 on page 10 gives the imputed data and the best fit line based on the mean of the β_0, β_1 iterates. This best fit line is very similar to what is obtained using the EM algorithm.

Figure 6: 8.2: Trace plot and histogram for σ .

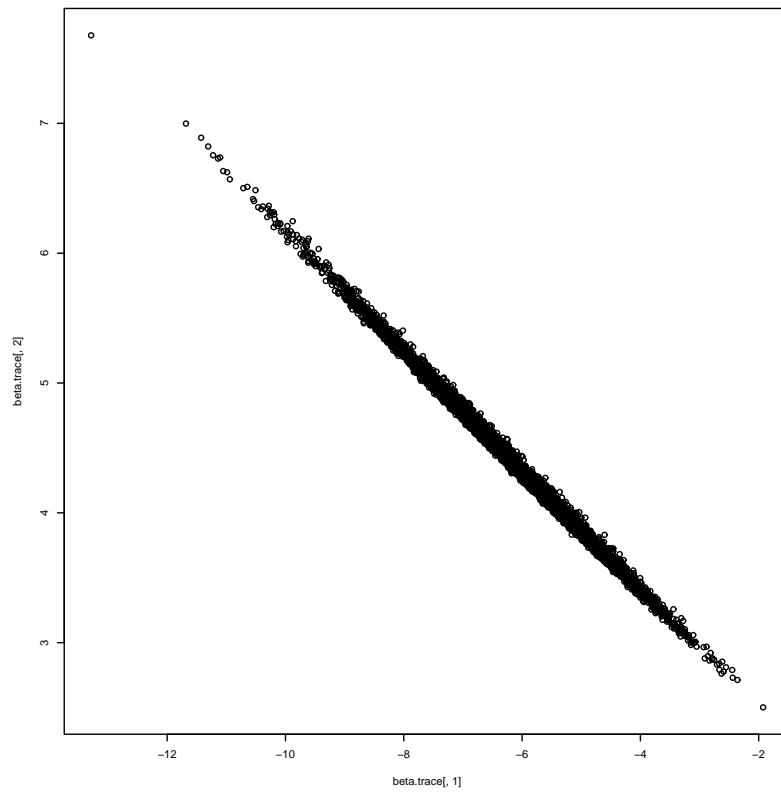


Figure 7: 8.2: β_0, β_1 bivariate draws.

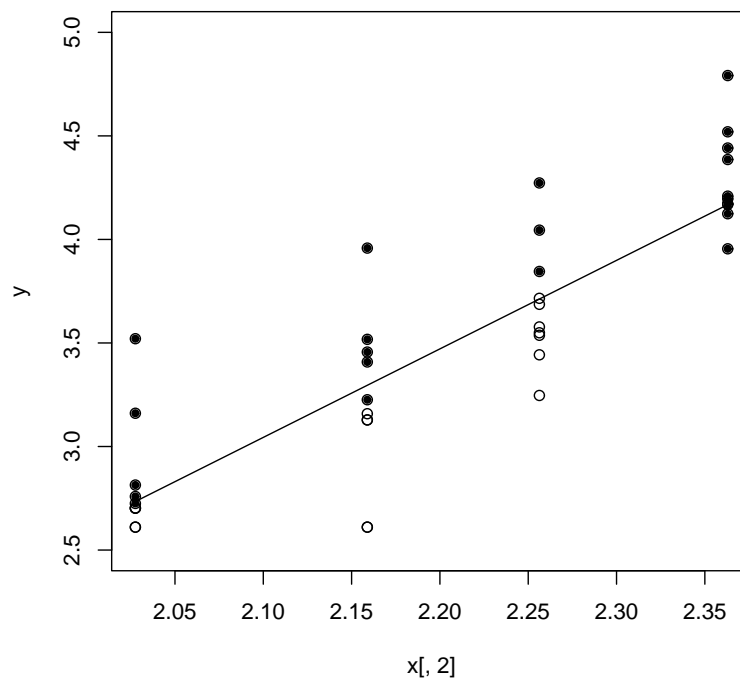


Figure 8: 8.2: Best fit line based on the means of the iterates of β_0, β_1 and imputed censored values (solid).

Method Comparison EM (estimates only):

$$\underline{\beta} = (-6.01925, 4.311247)', \quad \sigma = 0.2591827$$

Data Augmentation (also quantiles):

$$\underline{\beta} = (-6.181908, 4.397502)', \quad \sigma = 0.2961307$$

The central 95% credible interval for the parameters given by the data augmentation algorithm are given in the table. These values for $\underline{\beta}$ are very vague given the high correlation between β_0 and β_1 . A better confidence set is given by the ellipse seen in the plot in Figure 7 on page 9.

param	$Q_{.025}$	$Q_{.5}$	$Q_{.9751}$
β_0	-8.579352	-6.132612	-4.048179
β_1	3.435535	4.371296	5.525904
σ	0.1997597	0.2867794	0.4469327

The plot in Figure 9 on page 12 illustrates how similar the best fit estimates are between methods.

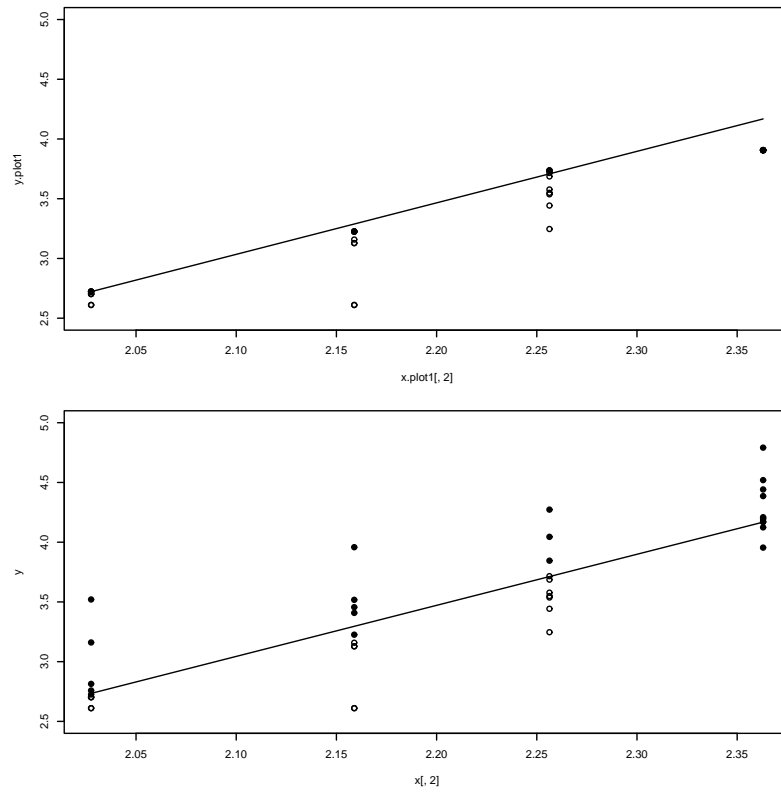


Figure 9: 8.3: EM/Data augmentation comparison.

Appendix

R code used for the above analysis

```

set.seed(271)
# options for printing
ps.options(horizontal=FALSE,height=6,width=6)
print.switch=0; # 0=display only, 1=home directory, 2=school directory
#### if(print.switch==1){postscript("f:\\USERS\\Erik\\UNM\\stat_computing\\schw05_.ps");}
#### if(print.switch==2){postscript("x:\\stat_computing\\schw05_.ps");}
####plot(x,prior1,type="l")
####title("Prior with mean=1, std=.5");
#### if(print.switch>0){dev.off()};
#####
#####
# Tanner p. 67 EM application for a censored regression model
# dataset
Y.data = c(-8064,-8064,-8064,-8064,-8064,-8064,-8064,-8064,-8064,
           1764, 2772, 3444, 3542, 3780, 4860, 5196,-5448,-5448,-5448,
           408, 408, 1344, 1344, 1440,-1680,-1680,-1680,-1680,-1680,
           408, 408, 504, 504, 504, 504, -528, -528, -528, -528)
temp = c(rep(150,10),rep(170,10),rep(190,10),rep(220,10))
times = abs(Y.data) # event times
cens = 1*(Y.data<0) # censoring indicator
motorette = data.frame(cbind(temp,times,cens))
y=log10(motorette$times) # event times
x=1000/(motorette$temp+273.2) # temperature groups
cen=motorette$cens==1 # censoring indicator (TRUE/FALSE)
cen.n=which(cen==TRUE) # censoring indexes
x = cbind(1,x) # x matrix
n = length(y)
p = dim(x)[2]
H = function(x){dnorm(x)/(1-pnorm(x))} # H ratio of normals function
ny = y # censored times to be updated by EM
beta = rep(0,p) # starts for betas
sigma = 1 # start for sigma
tol=1e-8 # EM convergence
iter = 0
m=100
beta.trace=matrix(data=NA,nrow=m,ncol=2)
sigma.trace=matrix(data=NA,nrow=m,ncol=1)
diff.trace=matrix(data=NA,nrow=m,ncol=1)
more = TRUE
while(more)
{
  iter = iter + 1
  # E-step
  mu = x%*%beta # expected values for event times
  # M-step
  res = y-mu # residuals
  ny[cen] = mu[cen]+sigma*H(res[cen]/sigma) # E(Z)
  ny2 = mu[cen]^2+sigma^2+sigma*(y[cen]+mu[cen])*H(res[cen]/sigma) # E(Z^2)
  vy = ny2-ny[cen]^2 # Var(Z)
  fit = lm(ny~x-1) # LS fit for beta MLEs (no intercept)
  # attributes(fit)
  b = fit$coefficients
  sig = sqrt((sum(fit$residuals^2)+sum(vy))/n) # MLE of sigma
  # convergence
  diff.conv = sum(abs(beta-b))+abs(log(sigma)-log(sig))
  more = (diff.conv>tol) # check convergence
  # update beta and sigma
  beta = b # update betas
  sigma = sig # update sigma
  beta.trace[iter,]=beta
  sigma.trace[iter]=sigma
  diff.trace[iter]=diff.conv
}
# iterates
matrix(data=c(1:iter,beta.trace[1:iter,],sigma.trace[1:iter],diff.trace[1:iter]),ncol=5,byrow=F)
cat("betas =",beta,"\t sigma =",sigma,"\t iter =",iter,"\n")
### betas = -6.01925 4.311247 sigma = 0.2591827 iter = 54
if(print.switch==1){postscript("f:\\USERS\\Erik\\UNM\\stat_computing\\schw08_1a.ps");}
if(print.switch==2){postscript("x:\\stat_computing\\schw08_1a.ps");}
par(mfrow=c(3,1), mar=c(4,4,2,2),cex=.5);
plot(1:iter,beta.trace[1:iter,1],"l")

```

```

plot(1:iter,beta.trace[1:iter,2],"1")
plot(1:iter,sigma.trace[1:iter],"1")
if(print.switch>0){dev.off()};
if(print.switch==1){postscript("f:\\USERS\\Erik\\UNM\\stat_computing\\schw08_1.ps");}
if(print.switch==2){postscript("x:\\stat_computing\\schw08_1.ps");}
par(mfrow=c(1,1));
x.plot1=x;y.plot1=y;beta.plot1=beta
plot(x.plot1[,2],y.plot1,ylim=c(2.5,5)) # plot data
points(x.plot1[cen.n,2],y.plot1[cen.n],type="p",pch=20)
lines(c(x.plot1[1,2],x.plot1[40,2]),c(t(x.plot1[1,])%*%t(beta.plot1)),t(x.plot1[40,])%*%t(beta.plot1))) # LS lin
if(print.switch>0){dev.off()};

#####
#####
# Tanner p. 96 Data Augmentation application for a censored regression model
# MLEs of betas and sigma (Gabriel Huerta's code [Code Class Feb 15-17] regression.s)
MLE.betas.sigma = function(x,y,n)
{
  Xdes = x
  # the inverse of t(X)X and its Chol dec.
  XX.t = (t(Xdes)%*%Xdes)
  XX.inv = solve(t(Xdes)%*%Xdes)
  XX.inv = 0.5*(XX.inv + t(XX.inv))
  XX.inv.cd = t(chol(XX.inv))
  # the MLE for ph
  betahat = XX.inv%*%t(Xdes)%*%y
  resid = (y - (Xdes)%*%betahat)
  ssq = sum (resid*resid)

  betahatmat = matrix(betahat,2,1)
  v = rchisq(1,(n-2))
  sg =ssq/v # sigma^2
  # generation of betas
  norbi = matrix(rnorm(2*1),ncol=1)
  normv = t(sqrt(sg)*t(norbi))
  beta = (XX.inv.cd%*%normv) + betahatmat # betas
  # result
  beta = t(beta)
  return(c(beta,sg))
}

# draw from truncated normal distribution
rnbnd <- function(N, mn, sd, lo, hi, tol) # http://www.r-project.org/nocvs/mail/r-help/2002/3789.html
{
  if (hi-mn < tol*sd || mn-lo < tol*sd) # Alan Zaslavsky's method
    return(mn + sd * qnorm(runif(N, pnorm((lo-mn)/sd), pnorm((hi-mn)/sd))))
  x <- rep(NA, N) # Rejection method
  bnd <- function(z, lo, hi) ifelse(z<lo | z>hi, NA, z)
  while (any(q <- is.na(x))) x[q] <- bnd(rnorm(sum(q), mn, sd), lo, hi)
  return(x)
}

# dataset
Y.data = c(-8064,-8064,-8064,-8064,-8064,-8064,-8064,-8064,-8064,-8064,
1764, 2772, 3444, 3542, 3780, 4860, 5196,-5448,-5448,-5448,
408, 408, 1344, 1344, 1440,-1680,-1680,-1680,-1680,-1680,
408, 408, 504, 504, 504, 504, -528, -528, -528, -528)
temp = c(rep(150,10),rep(170,10),rep(190,10),rep(220,10))
times = abs(Y.data) # event times
cens = 1*(Y.data<0) # censoring indicator
motorette = data.frame(cbind(temp,times,cens))
y=log10(motorette$times) # event times
y.org=y
x=1000/(motorette$temp+273.2) # temperature groups
cen=motorette$cens==1 # censoring indicator (TRUE/FALSE)
cen.n=which(cen==TRUE) # censoring indexes
x = cbind(1,x) # x matrix
n = length(y)

y = as.matrix(y) # censored times to be updated by EM
n = length(y)
m=10000 # iterations
beta.trace=matrix(data=NA,nrow=m,ncol=2)
sigma.trace=matrix(data=NA,nrow=m,ncol=1)
for (iter in 1:m)
{
  ## step a1 - generate beta0,beta1 and sigma2 from current guess of posterior
  MLEs = MLE.betas.sigma(x,y,n)
  beta.star = t(t(MLEs[1:2]))
  sigma.star = sqrt(MLEs[3])
}

```

```

# update beta and sigma
beta.trace[iter,]=beta.star
sigma.trace[iter]=sigma.star

## step a2 - impute censored values by drawing from truncated normal
for (cen.draw in cen.n)
{
  cens.time = y.org[cen.draw] # current y
  mean.star = x[cen.draw,]*%beta.star # mean x'b
  z.lower = (cens.time-mean.star)/sigma.star # z lower limit (cens time)
  z.star = rnbnd(N=1, mn=0, sd=1, lo=z.lower, hi=10000, tol=3) # draw from left-truncated normal
  y[cen.draw] = z.star*sigma.star+mean.star # impute new time into old time
}

cat("betas =", colMeans(beta.trace), "\t sigma =", mean(sigma.trace), "\t iter =", iter, "\n")
###   betas = -6.01925 4.311247   sigma = 0.2591827   iter = 54

# Confidence intervals
sorted=sort(beta.trace[,1]);cat(sorted[.025*m],sorted[.5*m],sorted[.976*m+1],"\n")
sorted=sort(beta.trace[,2]);cat(sorted[.025*m],sorted[.5*m],sorted[.976*m+1],"\n")
sorted=sort(sigma.trace);cat(sorted[.025*m],sorted[.5*m],sorted[.976*m+1],"\n")

###   -8.579352   -6.132612   -4.048179
###   3.435535    4.371296    5.525904
###   0.1997597   0.2867794   0.4469327

# Summary plot
if(print.switch==1){postscript("f:\\USERS\\Erik\\UNM\\stat_computing\\schw08_2.ps");}
if(print.switch==2){postscript("x:\\stat_computing\\schw08_2.ps");}
par(mfrow=c(4,2), mar=c(4,4,2,2),cex=.5);
plot(1:m,beta.trace[,1], "1")
hist(beta.trace[,1])
plot(1:m,beta.trace[,2], "1")
hist(beta.trace[,2])
plot(1:m,sigma.trace, "1")
hist(sigma.trace)
plot(beta.trace[,1],beta.trace[,2])
plot(x[,2],y) # plot data
points(x[cen.n,2],y[cen.n],type = "p",pch=20)
lines(c(x[1,2],x[40,2]),c(t(x[1,1])%*%t(t(beta)),t(x[40,1])%*%t(t(colMeans(beta.trace)))))) # LS line
if(print.switch>0){dev.off()};

# Individual plots
if(print.switch==1){postscript("f:\\USERS\\Erik\\UNM\\stat_computing\\schw08_2a.ps");}
if(print.switch==2){postscript("x:\\stat_computing\\schw08_2a.ps");}
par(mfrow=c(2,1), mar=c(4,4,2,2),cex=.5);
plot(1:m,beta.trace[,1], "1")
hist(beta.trace[,1])
if(print.switch>0){dev.off()};

if(print.switch==1){postscript("f:\\USERS\\Erik\\UNM\\stat_computing\\schw08_2b.ps");}
if(print.switch==2){postscript("x:\\stat_computing\\schw08_2b.ps");}
par(mfrow=c(2,1), mar=c(4,4,2,2),cex=.5);
plot(1:m,beta.trace[,2], "1")
hist(beta.trace[,2])
if(print.switch>0){dev.off()};

if(print.switch==1){postscript("f:\\USERS\\Erik\\UNM\\stat_computing\\schw08_2c.ps");}
if(print.switch==2){postscript("x:\\stat_computing\\schw08_2c.ps");}
par(mfrow=c(2,1), mar=c(4,4,2,2),cex=.5);
plot(1:m,sigma.trace, "1")
hist(sigma.trace)
if(print.switch>0){dev.off()};

if(print.switch==1){postscript("f:\\USERS\\Erik\\UNM\\stat_computing\\schw08_2d.ps");}
if(print.switch==2){postscript("x:\\stat_computing\\schw08_2d.ps");}
par(mfrow=c(1,1), mar=c(4,4,2,2),cex=.5);
plot(beta.trace[,1],beta.trace[,2])
if(print.switch>0){dev.off()};

if(print.switch==1){postscript("f:\\USERS\\Erik\\UNM\\stat_computing\\schw08_2e.ps");}
if(print.switch==2){postscript("x:\\stat_computing\\schw08_2e.ps");}
par(mfrow=c(1,1));
plot(x[,2],y,ylim=c(2.5,5)) # plot data
points(x[cen.n,2],y[cen.n],type = "p",pch=20)
lines(c(x[1,2],x[40,2]),c(t(x[1,1])%*%t(t(beta)),t(x[40,1])%*%t(t(colMeans(beta.trace)))))) # LS line
if(print.switch>0){dev.off()};

#####
# method comparison
if(print.switch==1){postscript("f:\\USERS\\Erik\\UNM\\stat_computing\\schw08_3.ps");}

```

```
if(print.switch==2){postscript("x:\\stat_computing\\schw08_3.ps")};
par(mfrow=c(2,1), mar=c(4,4,2,2),cex=.5);
plot(x.plot1[,2],y.plot1,ylim=c(2.5,5)) # plot data
points(x.plot1[cen.n,2],y.plot1[cen.n],type = "p",pch=20)
lines(c(x.plot1[1,2],x.plot1[40,2]),c(t(x.plot1[1,])%*%t(t(beta.plot1)),t(x.plot1[40,])%*%t(t(beta.plot1)))) # LS line
plot(x[,2],y,ylim=c(2.5,5)) # plot data
points(x[cen.n,2],y[cen.n],type = "p",pch=20)
lines(c(x[1,2],x[40,2]),c(t(x[1,])%*%t(t(beta)),t(x[40,])%*%t(t(colMeans(beta.trace)))) # LS line
if(print.switch>0){dev.off()};
# end
```