A Appendix: simulation code and model files

A.1 R code to simulate data

```
library(LaplacesDemon)
library(MASS)

################################################################
############ function to simulate survival time ###############
################################################################
# survival function is given by: S(t)= exp(- exp(B) * (exp(A*t) - 1) ) / A)
sim_Ti = function(n=500, alpha, delta=c(1,1), gamma=c(1,1)){
  Time = numeric(n)
  S = runif(n) # survival probability
  H = matrix(rnorm(2*n), ncol=2)
  W = matrix(rnorm(2*n), ncol=2)
  # random effects
  U = mvrnorm(n, mu=c(0,0), Sigma=matrix(c(0.09, 0.09*0.16, 0.09*0.16, 0.09),
   nrow=2, byrow=T))
  attributes(U)[[2]]=NULL # remove 'dimnames' attribute

  # calcualate survival time
  if(alpha[1]==0 & alpha[2]==0) Time = - log(S) / exp(gamma %*% t(W))

  else{
    B = alpha[1] * delta[1] * H[,1] + alpha[2] * U[,1] + gamma %*% t(W)
    A = alpha[2] * U[,2] + alpha[1] * delta[2] * H[,2]
    Time = log(1-log(S)*A/exp(B))/A
  }

  Ti_id = which(!is.na(Time))
  Time = Time[Ti_id][1:250] # true survival time: take the first 250 that are
   not NA
  Ci = rbeta(250, 4, 1)*2 # censoring time
  Ti = pmin(Time, Ci) # observed survival time: choose the smaller one
  event = as.numeric(Time == Ti) # 1 for event, 0 for censor
  U = U[Ti_id, ][1:250, ]
  H = H[Ti_id, ][1:250, ]
  W = W[Ti_id, ][1:250, ]

  list(Ti=Ti, event=event, H=H, U=U, W=W)
}

################################################################
###### function to simulate longitudinal data #################
################################################################
sim_longitudinal_data = function(survival_data=surdata, n=250, time=c(0, 0.25,
    0.5, 0.75, 1, 3), tau, sigma=1, beta=c(1,1), delta=c(1,1)){
  # survival_data - data simulated from survival model
  # n - # of subjects
  # time - time points of observations
```

```
  # tau - quantile
  # sigma - scale parameter for ALD
  time = time # at most # = length(time) observations per patient
  y = matrix(NA, nrow=n, ncol=length(time)) # wide format
  Ti = survival_data$Ti
  U = survival_data$U # random effects
  H = survival_data$H
  X = cbind(1, rnorm(n))
  count = sapply(Ti, function(x) sum(x > time)) # number of observations after
    drop-outs

  for (i in 1:n){
    for (j in 1:count[i]){
      location = beta %*% X[i, ] + delta %*% c(H[i,1], H[i,2]*time[j]) + U[i,]
    %*% c(1, time[j])
      y[i,j] = ralaplace(1, location, scale=sigma, kappa=tau)
    }
  }

  list(y = y, X = X, J=count)
}

#################################################################
###### function to simulate multiple joint data sets ##########
#################################################################
sim_multiple_data = function(N, sur_fun=sim_Ti, longi_fun=sim_longitudinal_
    data, alpha, tau){
  # N - number of data sets to generate
  # sur_fun - function to simulate survival data
  # longi_fun - function to sumualte longitudinal data
  # alpha - association parameters for JM
  # tau - quantile

  outdata = vector(mode='list', N)
  for (i in 1:N){
    sur_data = sur_fun(alpha=alpha)
    longi_data = longi_fun(sur_data, tau=tau)
    outdata[[i]] = list(survival_data=sur_data, longitudinal_data=longi_data)
  }
  outdata
}
```

A.2 JAGS model file

```
model{
  zero[1] <- 0
  zero[2] <- 0
  k1 <- (1-2*qt)/(qt*(1-qt))
  k2 <- 2/(qt*(1-qt))

  for (i in 1:I){
  # prior for random effects
```

```
  u[i, 1:2] ~ dmnorm(zero[], precision[,])

  # longitudinal process, BQR mixed model using ALD representation
    for (j in 1:J[i]){
      er[i,j] ~ dexp(sigma)
      mu[i,j] <- u[i,1] + u[i,2]*t[j] + inprod(X[i,], beta[]) + delta[1]*H[i
    ,1] + delta[2]*H[i,2]*t[j] + k1*er[i,j]
      prec[i,j] <- sigma/(k2*er[i,j])
      y[i,j] ~ dnorm(mu[i,j], prec[i,j])
    } #end of j loop

  # survival process, baseline hazard is set to 1
  A[i] <- alpha2*u[i,2] + alpha1*delta[2]*H[i,2]
  B[i] <- alpha1*delta[1]*H[i,1] + alpha2*u[i,1] + inprod(gamma, W[i,])
  S[i] <- exp(- exp(B[i])*(pow(exp(A[i]), Ti[i])-1)/A[i])
  h[i] <- exp(inprod(gamma, W[i,]) + alpha1*(delta[1]*H[i,1] + delta[2]*H[i
    ,2]*Ti[i]) + alpha2*(u[i,1] + u[i,2]*Ti[i]))
  L[i] <- pow(h[i], event[i])*S[i]/1.0E+08

  # zero trick
  phi[i] <- -log(L[i])
  zeros[i] ~ dpois(phi[i])

  }#end of i loop

  precision[1:2,1:2] <- inverse(Sigma[,])
  Sigma[1,1] <- 1
  Sigma[1,2] <- rho*sig1
  Sigma[2,1] <- Sigma[1,2]
  Sigma[2,2] <- sig1*sig1

  # priors for other parameters
  alpha1 ~ dnorm(0, 0.001)
  alpha2 ~ dnorm(0, 0.001)
  beta[1] ~ dnorm(0, 0.001)
  beta[2] ~ dnorm(0, 0.001)
  delta[1] ~ dnorm(0, 0.001)
  delta[2] ~ dnorm(0, 0.001)
  gamma[1] ~ dnorm(0, 0.001)
  gamma[2] ~ dnorm(0, 0.001)
  sigma ~ dgamma(0.001, 0.001)
  rho ~ dunif(-1, 1)
  sig1 ~  dgamma(0.01, 0.01)
}
```