Sample R Code to Estimate Random-Effects and Mixed Models

```
rm(list=ls())

#########################################################################
# Simple hierarchical random-effects model

# sufficient statistics for P(theta|mu, tau, Y)
suff_theta <- function(Y, mu, tau2, sigma2){
    n <- length(Y)
    m <- length(tau2)
    mean <- (matrix(Y, ncol = n, nrow = m, byrow = TRUE)*tau2 +
        matrix(sigma2, ncol = n, nrow = m, byrow = TRUE)*mu) /
        (matrix(sigma2, ncol = n, nrow = m, byrow = TRUE) + tau2)
    var <- matrix(sigma2, ncol = n, nrow = m, byrow = TRUE)*tau2 /
        (matrix(sigma2, ncol = n, nrow = m, byrow = TRUE) + tau2)
    # return the mean and variance for all e_i samples
    # in a matrix with nrow = # of e_i and ncol = # samples
    return(list(mean = mean, var = var))
}

# sufficient statistics for P(mu|tau, Y)
suff_mu <- function(Y, sigma2, tau2){
    # using lambda for the numerator of the mean of mu|tau, Y
    # omega for the denominator of the mean of mu|tau, Y and the
 variance
    lambda <- sum(Y/(sigma2 + tau2))
    omegai <- sum(1/(sigma2 + tau2))
    omega <- 1/omegai
    # return lambda and omega
    return(list(lambda = lambda, omega = omega))
}

# posterior samples from P(tau|Y)
# also gets all the sufficient statistics for P(mu|tau, Y)
post_tau2 <- function(Y, sigma2, tau2, n.sims){
    N <- length(tau2) # grid length
    log_p <- rep(NA, N)
    lambda <- log_p
    omega <- log_p
    # run all the points we choose on the tau_grid
    for(ii in 1:N){
        suff <- suff_mu(Y, sigma2, tau2[ii])
        lambda[ii] <- suff$lambda*suff$omega
        omega[ii] <- suff$omega
        log_p[ii] <- 0.5*log(omega[ii]) - 0.5*sum(log(sigma2+tau2[ii]))
        - 0.5*sum((Y - lambda[ii])^2/(sigma2 + tau2[ii]))
    }
    log_p <- log_p - max(log_p)
    p <- exp(log_p)
    p <- p/sum(p)
    # saving omega and lambda to avoid calculating them again
    index <- sample(1:N, n.sims, replace = T, prob = p)
    tau2 <- tau2[index]
    mean <- lambda[index]
    # factorizing the omega using cholesky decomposition
    omega <- (sqrt(omega))[index]
```

```
    # return all the selected tau2, mean and variance of mu|tau, Y
    return(list(tau2 = tau2, mean = mean, omega = omega))
}

# combine the above functions for a joint sample of (theta, mu, tau)
# data is the raw data from the .csv selecting a certain group
# tau2 is in the grid
# Y <- data$treatmentcoefficient
# n_sim is the number of simulations
post_mix_s <- function(data, Y, tau2, n_sim){
    sigma2 <- (data$treatmentstandarderror)^2
    # first sample tau2
    tau2 <- post_tau2(Y, sigma2, tau2, n_sim)
    # also get mean and variance for mu|tau, Y
    mean <- tau2$mean
    omega <- tau2$omega
    tau2 <- tau2$tau2
    # calculate mu
    mu <- rnorm(n_sim)
    mu <- mu*omega + mean
    # calculate sufficient statistics for theta
    theta <- suff_theta(Y, mu, tau2, sigma2)
    # sample theta
    theta <- array(rnorm(length(theta$mean), theta$mean,
 sqrt(theta$var)),
                   dim = dim(theta$mean))
    # calculate I2
    s2 <- (nrow(data)-1)*sum(1/sigma2)/((sum(1/sigma2))^2 -
 sum(1/sigma2^2))
    temp <- (mean(sqrt(tau2)))^2
    I2 <- 100*temp/(temp + s2)
    return(list(theta = t(theta), tau2 = tau2, mu = mu, I2 = I2))
}

#########################################################################
# Mixed model with moderator variable

# sufficient statistics for P(e_i|beta, tau, Y)
suff_ei <- function(X, Y, beta, tau2, sigma2){
    n <- length(Y)
    m <- length(tau2)
    tau2 <- matrix(tau2, nrow = n, ncol = m, byrow = TRUE)
    mean <- (Y - X%*%beta)*tau2/(sigma2 + tau2)
    var <- sigma2*tau2/(sigma2 + tau2)
    # return the mean and variance for all e_i samples
    # in a matrix with nrow = # of e_i and ncol = # samples
    return(list(mean = mean, var = var))
}

# sufficient statistics for P(beta|tau, Y)
suff_beta <- function(X, Y, sigma2, tau2){
    lambda <- t(Y/(sigma2 + tau2))%*%X
    omegai <- t(X)%*%(X/(sigma2 + tau2))
    omega <- solve(omegai)
    # return lambda and omega
    return(list(lambda = lambda, omega = omega))
}
```

```
# posterior samples from P(tau|Y)
# also gets all the sufficient statistics for P(beta|tau, Y)
post_tau1 <- function(X, Y, sigma2, tau2, n.sims){
    Np <- ncol(X)
    N <- length(tau2) #grid length
    log_p <- rep(NA, N)
    lambda <- matrix(NA, ncol = Np , nrow = N)
    mean <- lambda
    omega <- array(NA, dim = c(N, Np , Np ))
    # run all the points we choose on the tau_grid
    for(ii in 1:N){
        suff <- suff_beta(X, Y, sigma2, tau2[ii])
        lambda[ii,] <- suff$lambda
        omega[ii,,] <- suff$omega
        mean[ii,] <- omega[ii,,]%*%lambda[ii,]
        log_p[ii] <- - 0.5*sum(log(sigma2+tau2[ii])) +
 0.5*log(det(suff$omega)) -
            0.5*sum((Y-X%*%mean[ii,])^2/(sigma2 + tau2[ii]))
    }
    log_p <- log_p - max(log_p)
    p <- exp(log_p)
    p <- p/sum(p)
    # saving omega and lambda to avoid calculating them again
    index <- sample(1:N, n.sims, replace = T, prob = p)
    tau2 <- tau2[index]
    mean <- mean[index,]
    # factorizing the omega using cholesky decomposition
    omega <- t(apply(omega, 1, chol))
    omega <- omega[index,]
    # return all the selected tau2, mean and variance of beta|tau, Y
    return(list(tau2 = tau2, mean = mean, omega = omega))
}

# combine the above functions for a joint sample of (e_i, beta, tau)
# data is the raw data from the .csv selecting a certain group
# tau2 is in the grid
# Y <- data$treatmentcoefficient
# X <- cbind(1, data$mod1)
# n_sim is the number of simulations
post_mix <- function(data,X,Y ,tau2, n_sim){
    if(is.data.frame(X) ) X <- as.matrix(X)
    sigma2 <- (data$treatmentstandarderror)^2
    nn <- ncol(X)
    # first sample tau2
    tau2 <- post_tau1(X, Y, sigma2, tau2, n_sim)
    # also get mean and variance for beta|tau, Y
    mean <- tau2$mean
    omega <- array(tau2$omega, dim = c(nrow(tau2$omega), nn, nn))
    tau2 <- tau2$tau2
    # calculate beta
    beta <- matrix(rnorm(nn*n_sim), ncol = nn)
    beta <- do.call(rbind, lapply(seq_len(dim(omega)[1]),
              function(jj) t(beta[jj,])%*%omega[jj,,]))
    beta <- beta + mean
    # calculate sufficient statistics for e_i
    e_i <- suff_ei(X, Y, t(beta), tau2, sigma2)
    # sample e_i
    e_i <- array(rnorm(length(e_i$mean), e_i$mean, sqrt(e_i$var)),
```

```
                dim = dim(e_i$mean))
    # calculate I2
    s2 <- (nrow(data)-1)*sum(1/sigma2)/((sum(1/sigma2))^2 -
 sum(1/sigma2^2))
    temp <- (mean(sqrt(tau2)))^2
    I2 <- 100*temp/(temp + s2)
    return(list(e_i = t(e_i), tau2 = tau2, beta = beta, I2 = I2))
}

#########################################################################
# Example run

# setwd()
# read in data
# data should include group_id for intervention-outcome identifier
# treatmentcoefficient for point estimate
# treatmentstandarderror for standard error
ds <- read.csv("data.csv")
head(ds)

# number of results per group
group <- unique(ds$group_id)
groupsize <- sapply(group, function(x) sum(ds$group_id==x))

# restrict attention to those groups with at least 3 results
group <- group[groupsize>2]

# estimate true theta and tau for each group using all data
TrueResult <- list()
# number of simulations
n_sim = 1e5

# random-effects model example
set.seed(100)
for(ii in 1:length(group)){
  data <- ds[ds$group_id == group[ii], ]
  gridmax <- 10*sd(data$treatmentcoefficient)
  n.grid <- 2000
  tau.grid <- seq(gridmax/n.grid, gridmax, length=n.grid)
  # discrete distribution for tau
  tau2 <- tau.grid^2
  stat <- post_mix_s(data,Y = data$treatmentcoefficient,tau2, n_sim)
  TrueResult[[ii]] <- matrix(c(group[ii], mean(stat$tau2),
 mean(stat$mu),rowMeans(stat$theta)),nrow=1)
  colnames(TrueResult[[ii]]) <-
 c("group_id","tau2","mu",paste0("theta",1:nrow(data)))
}
names(TrueResult) <- group
head(TrueResult)

# mixed model example
# this assumes explanatory variables are being simulated;
# if the data contain an explanatory variable xi, do not need to
 simulate
set.seed(100)
for(ii in 1:length(group)){
  data <- ds[ds$group_id == group[ii], ]
  gridmax <- 10*sd(data$treatmentcoefficient)
```

```
  n.grid <- 2000
  tau.grid <- seq(gridmax/n.grid, gridmax, length=n.grid)
  # discrete distribution for tau
  tau2 <- tau.grid^2
  # this yields R^2 of 0.5. Adjust multiplier on sd for different R^2
  TrueResult[[ii]]  <- replicate(1000, {  data$xi <-
data$treatmentcoefficient+rnorm(nrow(data),0,sd=1*sd(data$treatmentcoef
ficient))
  stat1 <- post_mix(data,X=cbind(1,data$xi),
Y=data$treatmentcoefficient, tau2, n_sim=1000)

c(group[ii],mean(stat1$tau2),colMeans(stat1$beta),colMeans(stat1$e_i))
  })
  rownames(TrueResult[[ii]]) <-
c("group_id","tau2","beta0","beta1",paste0("e_",1:nrow(data)))
  }
names(TrueResult) <- group
head(TrueResult)
```