HW9

ASG

2023-04-18

4.10

a

```
library(HRW)   ;   library(rstan)   ;   library(lattice)
```

```
## Loading required package: StanHeaders
```

```
## Loading required package: ggplot2
```

```
## rstan (Version 2.21.8, GitRev: 2e1f913d3ca3)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
```

```
library(mgcv)
```

```
## Loading required package: nlme
```

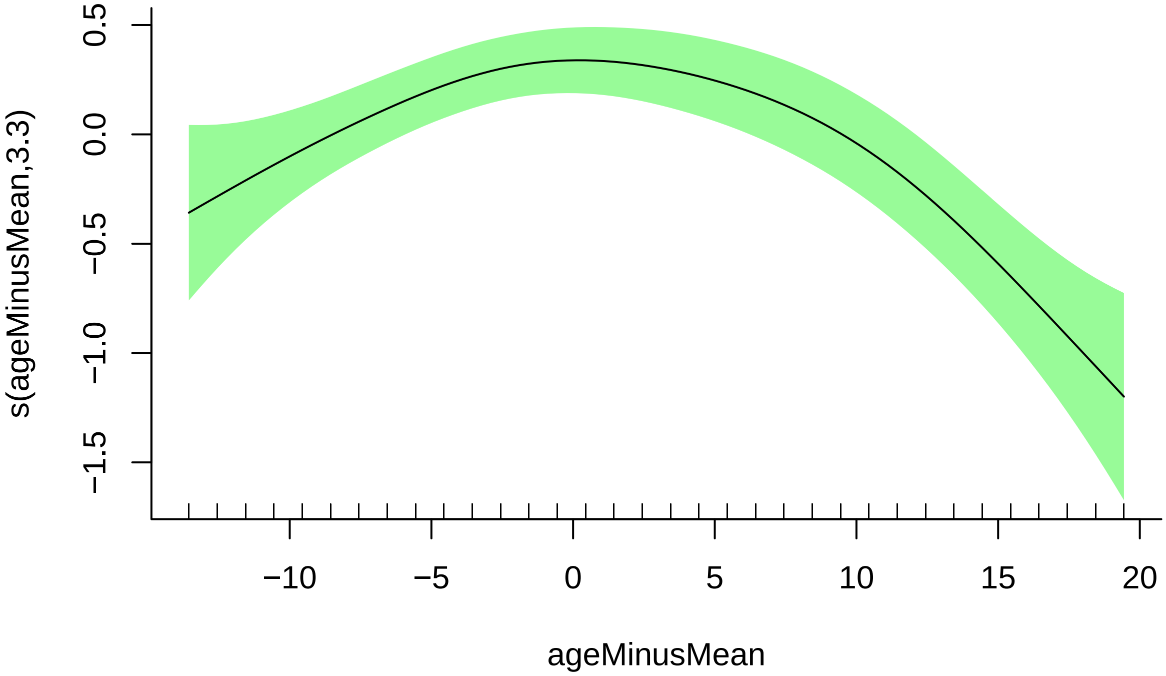
```
## This is mgcv 1.8-42. For overview type 'help("mgcv-package")'.
```

```
data(BanglaContrac)
colnames(BanglaContrac)
```

```
## [1] "districtID"         "usingContraception" "childCode"
## [4] "ageMinusMean"       "isUrban"
```

```
idnum <- BanglaContrac$districtID
idnum[idnum==61]<-54
BanglaContrac$oneChild=ifelse(BanglaContrac$childCode==2,1,0)
BanglaContrac$twoChild=ifelse(BanglaContrac$childCode==3,1,0)
BanglaContrac$moreChild=ifelse(BanglaContrac$childCode==4,1,0)
fit <- gamm(usingContraception~oneChild+twoChild+moreChild+s(ageMinusMean)+isUrban,random = list(idnum =
##
##  Maximum number of PQL iterations:  20

## iteration 1

## iteration 2

## iteration 3
```

```
plot(fit$gam,shade = TRUE,shade.col = "palegreen",bty = "l")
```



```
summary(fit)
```

```
##     Length Class Mode
## lme 20     lme   list
## gam 31     gam   list
```

b

```
########## R script: indonRespirBayes ##########

# For conducting a Bayesian additive mixed
# model analysis on the spinal bone mineral
# density data using Markov chain Monte Carlo
# and Stan.

# Last changed: 22 AUG 2017

# Set flag for code compilation (needed if
# running script first time in current session):

compileCode <- TRUE

# Set MCMC sample size parameters:

nWarm <- 5000
nKept <- 5000
nThin <- 5

# Load required packages:

library(HRW)   ;   library(rstan)   ;   library(lattice)

# Load in the Indonesian respiratory data and extract the
# component variables:

data(BanglaContrac)

idnum <- BanglaContrac$districtID
idnum[idnum==61]<-54
BanglaContrac$oneChild=ifelse(BanglaContrac$childCode==2,1,0)
BanglaContrac$twoChild=ifelse(BanglaContrac$childCode==3,1,0)
BanglaContrac$moreChild=ifelse(BanglaContrac$childCode==4,1,0)
y <- BanglaContrac$usingContraception
x1 <- BanglaContrac$oneChild
x2 <- BanglaContrac$twoChild
x3 <- BanglaContrac$moreChild
x4Orig <-BanglaContrac$ageMinusMean
x5 <- BanglaContrac$isUrban

numObs <- length(y)
numGrp <- length(unique(idnum))

# Standardize continuous data for Bayesian analysis:

sd.x4 <- sd(x4Orig)
x4 <- (x4Orig)/sd.x4

# Set up matrices for additive model:

numObs <- length(y)
X <- cbind(rep(1,numObs),x1,x2,x3,x4,x5)
ncX <- ncol(X)

numIntKnots <- 15
```

```
intKnots <- quantile(unique(x4),seq(0,1,length =
                 (numIntKnots+2))[-c(1,(numIntKnots+2))])
range.x4 <- c(1.01*min(x4)-0.01*max(x4),1.01*max(x4)-0.01*min(x4))
Zspl <- ZOSull(x4,intKnots = intKnots,range.x = range.x4)
ncZspl <- ncol(Zspl)

numSpl <- ncol(Zspl)
numGrp <- length(unique(idnum))
numObs <- length(y)

# Set hyperparameters:

sigmaBeta <- 1e5     ;     Agrp <- 1e5    ;     Aspl <- 1e5

# Specify model in Stan:

logistAddMixModModel <-
'data
{
   int<lower=1> numObs;              int<lower=1> numGrp;
   int<lower=1> ncX;                 int<lower=1> ncZspl;
   real<lower=0> sigmaBeta;
   real<lower=0> Agrp;               real<lower=0> Aspl;
   int<lower=0,upper=1> y[numObs];   int<lower=1> idnum[numObs];
   matrix[numObs,ncX] X;             matrix[numObs,ncZspl] Zspl;
}
parameters
{
   vector[ncX] beta;            vector[numGrp] U;
   vector[ncZspl] u;            real<lower=0> sigmaGrp;
   real<lower=0> sigmaSpl;
}
model
{
   y ~ bernoulli_logit(X*beta + U[idnum] + Zspl*u);
   U ~ normal(0,sigmaGrp);           u ~ normal(0,sigmaSpl);
   beta  ~ normal(0,sigmaBeta) ;     sigmaGrp ~ cauchy(0,Agrp) ;
   sigmaSpl ~ cauchy(0,Aspl);
}'

# Fit model using MCMC via Stan:

allData <- list(numObs = numObs,numGrp = numGrp,ncX = ncX,ncZspl = ncZspl,
                idnum = idnum,X = X,y = y,Zspl = Zspl,sigmaBeta = sigmaBeta,
                Agrp = Agrp,Aspl = Aspl)

# Compile code for model if required:

if (compileCode)
   stanCompilObj <- stan(model_code = logistAddMixModModel,data = allData,
                         iter = 1,chains = 1)
```

```
##
```

```
## SAMPLING FOR MODEL '5d25270b4c3dce6bb619abb780fa1755' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000592 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 5.92 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: WARNING: No variance estimation is
## Chain 1:          performed for num_warmup < 20
## Chain 1:
## Chain 1: Iteration: 1 / 1 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 1e-06 seconds (Warm-up)
## Chain 1:                0.000475 seconds (Sampling)
## Chain 1:                0.000476 seconds (Total)
## Chain 1:
```

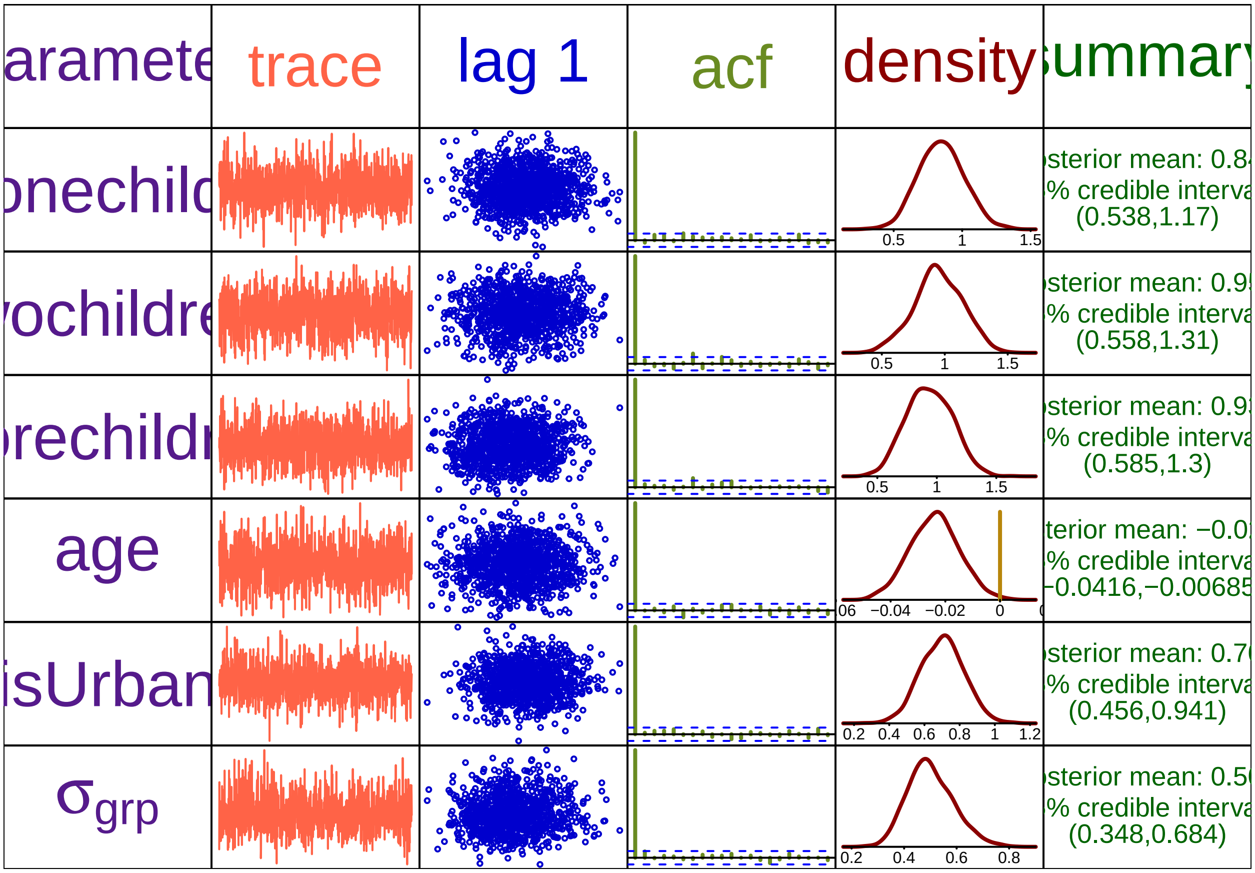
```
## Warning: There were 1 divergent transitions after warmup. See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

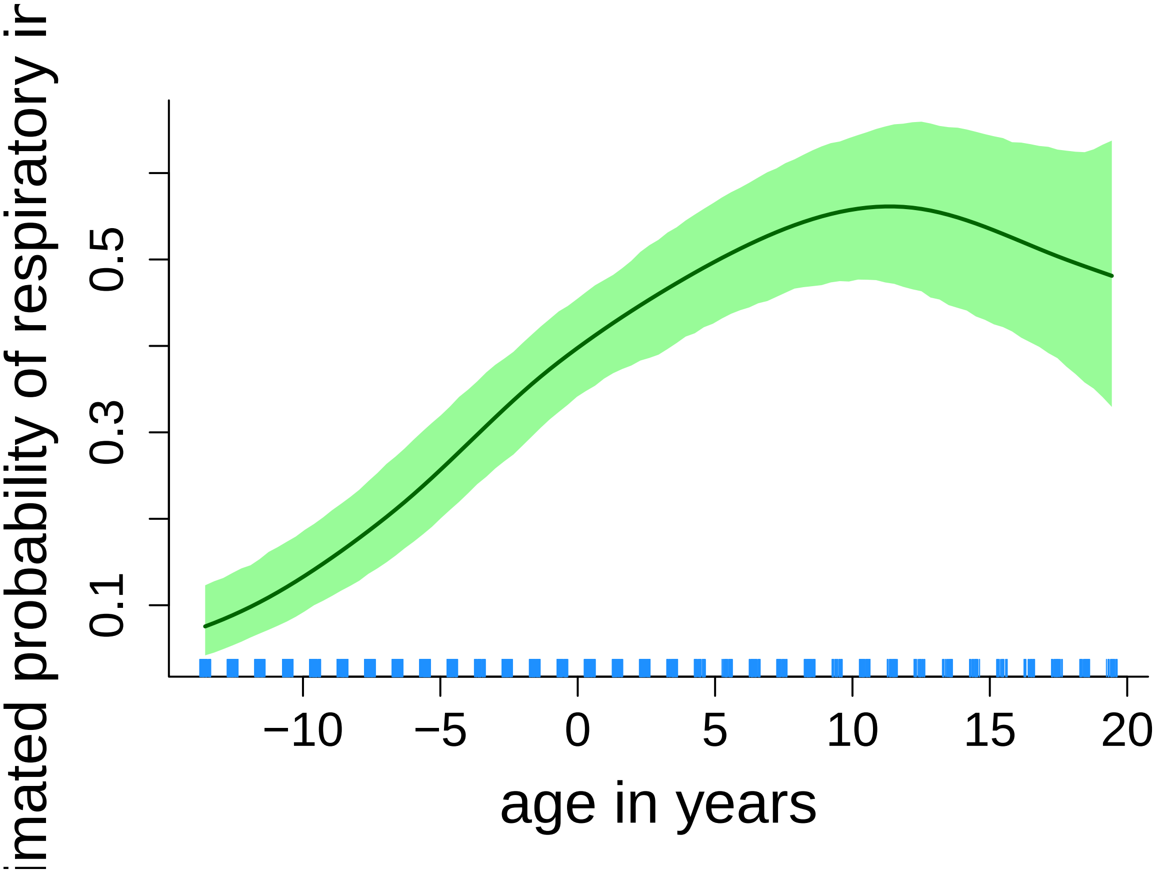
```r
# Perform MCMC:

stanObj <-  stan(model_code = logistAddMixModModel,data = allData,warmup = nWarm,
                 iter = (nWarm + nKept),chains = 1,thin = nThin,refresh = 1000,
                 fit = stanCompilObj)
```

```
##
## SAMPLING FOR MODEL '5d25270b4c3dce6bb619abb780fa1755' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000301 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 3.01 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 10000 [  0%]  (Warmup)
## Chain 1: Iteration: 1000 / 10000 [ 10%]  (Warmup)
## Chain 1: Iteration: 2000 / 10000 [ 20%]  (Warmup)
## Chain 1: Iteration: 3000 / 10000 [ 30%]  (Warmup)
## Chain 1: Iteration: 4000 / 10000 [ 40%]  (Warmup)
## Chain 1: Iteration: 5000 / 10000 [ 50%]  (Warmup)
## Chain 1: Iteration: 5001 / 10000 [ 50%]  (Sampling)
## Chain 1: Iteration: 6000 / 10000 [ 60%]  (Sampling)
## Chain 1: Iteration: 7000 / 10000 [ 70%]  (Sampling)
## Chain 1: Iteration: 8000 / 10000 [ 80%]  (Sampling)
## Chain 1: Iteration: 9000 / 10000 [ 90%]  (Sampling)
## Chain 1: Iteration: 10000 / 10000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 20.5608 seconds (Warm-up)
## Chain 1:                18.8151 seconds (Sampling)
## Chain 1:                39.376 seconds (Total)
## Chain 1:
```

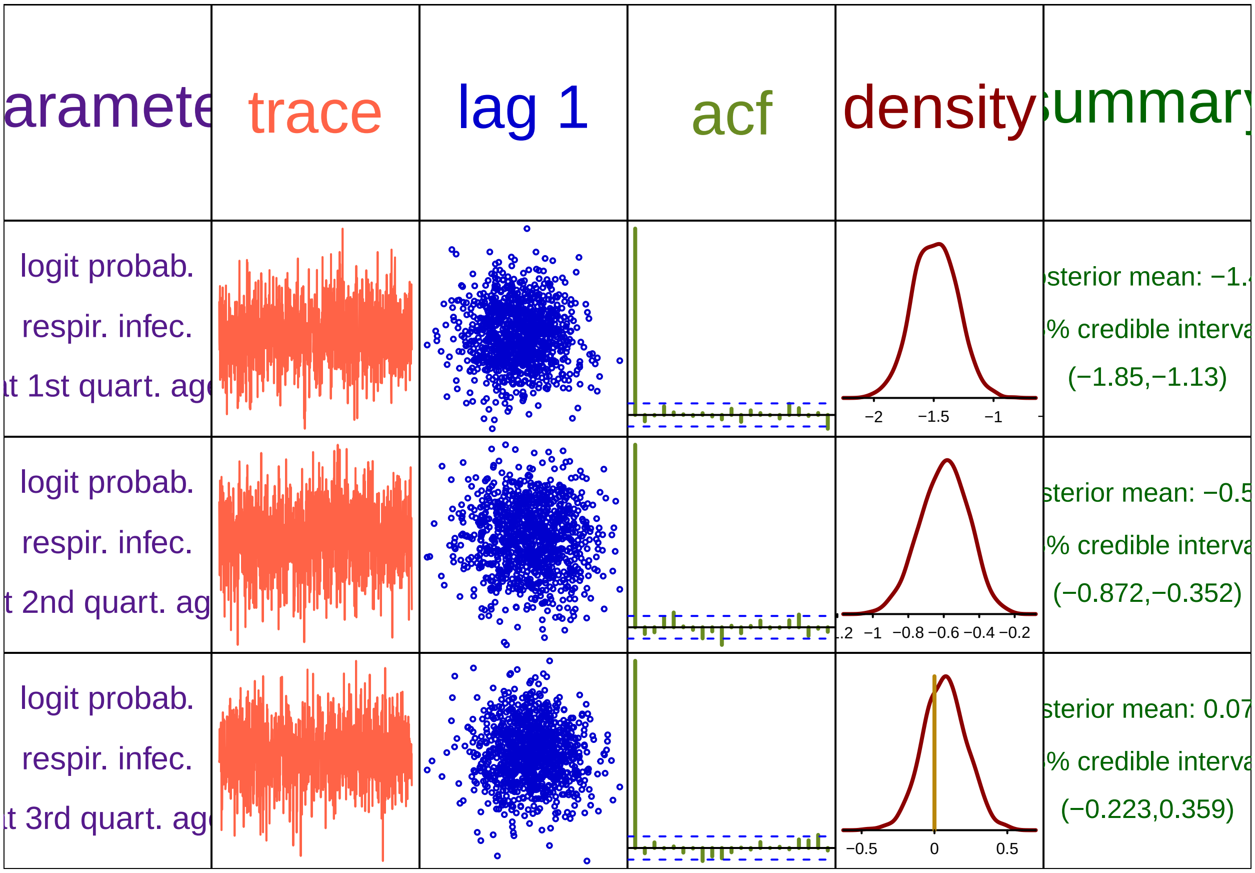
```
# Save and extract relevant MCMC samples:

beta0MCMC <- extract(stanObj,"beta[1]",permuted = FALSE)
beta1MCMC <- extract(stanObj,"beta[2]",permuted = FALSE)
beta2MCMC <- extract(stanObj,"beta[3]",permuted = FALSE)
beta3MCMC <- extract(stanObj,"beta[4]",permuted = FALSE)
beta4MCMC <- extract(stanObj,"beta[5]",permuted = FALSE)
beta5MCMC <- extract(stanObj,"beta[6]",permuted = FALSE)
sigmaGrpMCMC <- extract(stanObj,"sigmaGrp",permuted = FALSE)
sigmaSplMCMC <- extract(stanObj,"sigmaSpl",permuted = FALSE)

uMCMC <- NULL
for (k in 1:ncZspl)
{
   charVar <- paste("u[",as.character(k),"]",sep = "")
   uMCMC <- rbind(uMCMC,extract(stanObj,charVar,permuted = FALSE))
}

# Convert to coefficient for continuous predictor to correspond
# to original scale:

beta4MCMCorig <- beta4MCMC/sd.x4

# Do parameters plot:

parms <- list(cbind(beta1MCMC,beta2MCMC,beta3MCMC,beta4MCMCorig,beta5MCMC,sigmaGrpMCMC))
parNamesVal <-   list(c("onechild"),c("twochildren"),c("Morechildren"),
                      c("age"), c("isUrban"),c(expression(sigma[grp])))

summMCMC(parms,parNames = parNamesVal)
```



```
# Do plot for estimated probability function given age:

par(mai = c(1,1.2,0.5,0.2))
cex.labVal <- 1.8   ;  cex.axisVal <- 1.5
ng <- 101
x4g <- seq(min(x4),max(x4),length = ng)
XotherPreds <- X[,2:6][,-4]
otherPredsMeans <- apply(XotherPreds,2,mean)
covarAddOn <- t(matrix(rep(otherPredsMeans,ng),
                    length(otherPredsMeans),ng))
Xg <- cbind(rep(1,ng),covarAddOn,x4g)
Zg <- ZOSull(x4g,intKnots = intKnots,range.x = range.x4)

betaMCMC <- rbind(beta0MCMC,beta1MCMC,beta2MCMC,beta3MCMC,beta4MCMCorig,beta5MCMC)

etaHatMCMC <- Xg%*%betaMCMC + Zg%*%uMCMC
muHatMCMC <- 1/(1+exp(-etaHatMCMC))

credLower <- apply(muHatMCMC,1,quantile,0.025)
credUpper <- apply(muHatMCMC,1,quantile,0.975)
muHatg <- apply(muHatMCMC,1,mean)
ylimVal <- range(c(credLower,credUpper))

x4gOrig <- sd.x4*x4g
plot(x4gOrig,muHatg,type = "n",xlab = "age in years",
     ylab = "estimated probability of respiratory infection",
     ylim = ylimVal,bty = "l",cex.lab = cex.labVal,cex.axis = cex.axisVal)
polygon(c(x4gOrig,rev(x4gOrig)),c(credLower,rev(credUpper)),
        col = "palegreen",border = FALSE)
lines(x4gOrig,muHatg,col = "darkgreen",lwd = 2)
rug(jitter(x4Orig),col = "dodgerblue")
```



```
# Do a summaries of the MCMC samples for slices of
# the logit probability function at the quartiles:

indQ1 <- length(x4g[x4g<quantile(x4,0.25)])
indQ2 <- length(x4g[x4g<quantile(x4,0.50)])
indQ3 <- length(x4g[x4g<quantile(x4,0.75)])

etaHatMCMCQ1 <- etaHatMCMC[indQ1,]
etaHatMCMCQ2 <- etaHatMCMC[indQ2,]
etaHatMCMCQ3 <- etaHatMCMC[indQ3,]

parms <- list(cbind(etaHatMCMCQ1,etaHatMCMCQ2,etaHatMCMCQ3))
parNamesVal <- list(c("logit probab.","respir. infec.","at 1st quart. age"),
                    c("logit probab.","respir. infec.","at 2nd quart. age"),
                    c("logit probab.","respir. infec.","at 3rd quart. age"))
summMCMC(parms,parNames = parNamesVal)
```



```
########## End of indonRespirBayes ##########
```

c

```
### Odds Ratio

expBetaMCMC<-cbind(beta1MCMC,beta2MCMC,beta3MCMC,beta5MCMC)
expBetaMCMC<-exp(expBetaMCMC)
credlowerexpBetaMCMC<-apply(expBetaMCMC,1,quantile,0.025)
credupperexpBetaMCMC<-apply(expBetaMCMC,1,quantile,0.975)
```

2

One of the major difference between mixed and marginal models is how they model random effects. Mixed effect models , models both the random and fixed effect while the marginal model doesnt explicitly model the random effect. Mixed effect models usually are used when we want to focus on individual level effect while marginal models are more focused of population mean level effects. Based on the last point we can say that when we there are heirarchical and nested type of structure for example modelling the indviduals of every depaertment in a school. where there are classes and departments. It would be helpful to use the mixed effects model. Meanwhile marginal models are used for population level mean studies. For example, it can be used to estimate the average effect of a drug on individuals on a clinical study