SynthesisModel

Sarah Boese

2/6/2020

```
library(ProbBayes)
library(dplyr)
library(ggplot2)
require(gridExtra)
library(reshape)
library(runjags)
library(coda)
library(tidyverse)
library(fastDummies)
crcblue <- "#2905a1"
```

```
CESample <- read.csv("CEsample2.csv")
```

I decided that I wanted to use all variables within CESampe logExpenditure, UrbanRural, Race) as estimators for logIncome. Thus, I used a Multilinear regression model in which I scaled Log Income and Log Expenditure by centering at 0 and dividing by standard deviation. I use the following MLR model (where * denotes a standardized continuous variable):

$$\begin{aligned} Y_i^* \mid \beta_0, \beta_1, \cdots, \beta_7, \sigma, \mathbf{x}_i^* \overset{ind}{\sim} \text{Normal}(\beta_0 &+ \beta_1 x^*_{i,expenditure} + \beta_2 x_{i,rural} \\ &+ \beta_3 x_{i,race_B} + \beta_4 x_{i,race_N} \\ &+ \beta_5 x_{i,race_A} + \beta_6 x_{i,race_P} \\ &+ \beta_7 x_{i,race_M}, \sigma). \end{aligned} \tag{1}$$

```
CESample <- CESample %>%
  mutate(LogTotalIncome = log(TotalIncomeLastYear))
CESample <- CESample %>%
  mutate(LogTotalExp = log(TotalExpLastQ))
```

```
CESample$Log_TotalExpSTD <- scale(CESample$LogTotalExp)
CESample$Log_TotalIncomeSTD <- scale(CESample$LogTotalIncome)
## create indictor variable for Rural
CESample$Rural = fastDummies::dummy_cols(CESample$UrbanRural)[,names(fastDummies::dummy_cols(CESample$U
 == ".data_2"]
```

```
## create indicator variables for Black (2), Native American (3),
## Asian (4), Pacific Islander (5), and Multi-race (6)
CESample$Race_Black = fastDummies::dummy_cols(CESample$Race)[,names(fastDummies::dummy_cols(CESample$Rac
CESample$Race_NA = fastDummies::dummy_cols(CESample$Race)[,names(fastDummies::dummy_cols(CESample$Race)
CESample$Race_Asian = fastDummies::dummy_cols(CESample$Race)[,names(fastDummies::dummy_cols(CESample$Rac
CESample$Race_PI = fastDummies::dummy_cols(CESample$Race)[,names(fastDummies::dummy_cols(CESample$Race)
CESample$Race_M = fastDummies::dummy_cols(CESample$Race)[,names(fastDummies::dummy_cols(CESample$Race))
```

```
modelString <-"
model {
## sampling
for (i in 1:N){
y[i] ~ dnorm(beta0 + beta1*x_exp[i] + beta2*x_rural[i] +
beta3*x_race_B[i] + beta4*x_race_N[i] +
beta5*x_race_A[i] + beta6*x_race_P[i] +
beta7*x_race_M[i], invsigma2)
}
## priors
beta0 ~ dnorm(mu0, g0)
beta1 ~ dnorm(mu1, g1)
beta2 ~ dnorm(mu2, g2)
beta3 ~ dnorm(mu3, g3)
beta4 ~ dnorm(mu4, g4)
beta5 ~ dnorm(mu5, g5)
beta6 ~ dnorm(mu6, g6)
beta7 ~ dnorm(mu7, g7)
invsigma2 ~ dgamma(a, b)
sigma <- sqrt(pow(invsigma2, -1))
}
"
```

- Pass the data and hyperparameter values to JAGS:

```
y = as.vector(CESample$Log_TotalIncomeSTD)
x_exp = as.vector(CESample$Log_TotalExpSTD)
x_rural = as.vector(CESample$Rural)
x_race_B = as.vector(CESample$Race_Black)
x_race_N = as.vector(CESample$Race_NA)
x_race_A = as.vector(CESample$Race_Asian)
x_race_P = as.vector(CESample$Race_PI)
x_race_M = as.vector(CESample$Race_M)
N = length(y)  # Compute the number of observations
```

- Pass the data and hyperparameter values to JAGS:

```
the_data <- list("y" = y, "x_exp" = x_exp,
                 "x_rural" = x_rural, "x_race_B" = x_race_B,
                 "x_race_N" = x_race_N, "x_race_A" = x_race_A,
                 "x_race_P" = x_race_P, "x_race_M" = x_race_M,
                 "N" = N,
                 "mu0" = 0, "g0" = 0.0001, "mu1" = 0, "g1" = 0.0001,
                 "mu2" = 0, "g2" = 1, "mu3" = 0, "g3" = 1,
                 "mu4" = 0, "g4" = 1, "mu5" = 0, "g5" = 1,
                 "mu6" = 0, "g6" = 1, "mu7" = 0, "g7" = 1,
                 "a" = 1, "b" = 1)
```

- Pass the data and hyperparameter values to JAGS:

```
initsfunction <- function(chain){
  .RNG.seed <- c(1,2)[chain]
  .RNG.name <- c("base::Super-Duper",
                 "base::Wichmann-Hill")[chain]
  return(list(.RNG.seed=.RNG.seed,
              .RNG.name=.RNG.name))
}
```

- Run the JAGS code for this model:

```
posterior_MLR <- run.jags(modelString,
                      n.chains = 1,
                      data = the_data,
                      monitor = c("beta0", "beta1", "beta2",
                                  "beta3", "beta4", "beta5",
                                  "beta6", "beta7", "sigma"),
                      adapt = 1000,
                      burnin = 5000,
                      sample = 5000,
                      thin = 2,
                      inits = initsfunction)
```

```
## Calling the simulation...
## Welcome to JAGS 4.3.0 on Tue Feb 11 12:42:06 2020
## JAGS is free software and comes with ABSOLUTELY NO WARRANTY
## Loading module: basemod: ok
## Loading module: bugs: ok
## . . Reading data file data.txt
## . Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 994
##    Unobserved stochastic nodes: 9
##    Total graph size: 9984
## . Reading parameter file inits1.txt
## . Initializing model
## . Adaptation skipped: model is not in adaptive mode.
## . Updating 5000
## -------------------------------------------------| 5000
## ************************************************** 100%
## . . . . . . . . . . Updating 10000
## -------------------------------------------------| 10000
## ************************************************** 100%
## . . . . Updating 0
## . Deleting model
## .
## Note: the model did not require adaptation
## Simulation complete.  Reading coda files...
## Coda files loaded successfully
## Calculating summary statistics...

## Warning: Convergence cannot be assessed with only 1 chain
## Finished running the simulation
```

JAGS output for the MLR model

```
summary(posterior_MLR)
```

```
##           Lower95      Median    Upper95        Mean         SD Mode
## beta0 -0.0311177  0.02608835  0.0869957  0.02606177 0.02980286   NA
## beta1  0.4918340  0.54553450  0.5955600  0.54534824 0.02668857   NA
## beta2 -0.2743130 -0.04835345  0.1952340 -0.04751894 0.11885285   NA
## beta3 -0.3097090 -0.14332450  0.0261562 -0.14148264 0.08690774   NA
## beta4 -1.1226000 -0.51939250  0.0605914 -0.52345066 0.30013430   NA
## beta5 -0.1497080  0.11270250  0.3803880  0.11387820 0.13525668   NA
## beta6 -0.9134750 -0.28965850  0.3296200 -0.28791357 0.31720398   NA
## beta7 -0.8128310 -0.42725700 -0.0305582 -0.43054482 0.20023325   NA
## sigma  0.7932920  0.82832250  0.8669630  0.82850990 0.01870803   NA
##               MCerr MC%ofSD SSeff        AC.20 psrf
## beta0 0.0004448655     1.5  4488 -0.002126058   NA
## beta1 0.0003774333     1.4  5000 -0.001473396   NA
## beta2 0.0016808331     1.4  5000 -0.009477521   NA
## beta3 0.0012297711     1.4  4994 -0.010830759   NA
## beta4 0.0039695622     1.3  5717  0.034892325   NA
## beta5 0.0019128183     1.4  5000  0.004351354   NA
## beta6 0.0044859417     1.4  5000  0.021879566   NA
## beta7 0.0027856122     1.4  5167 -0.015481990   NA
## sigma 0.0002733078     1.5  4685  0.001118300   NA
```

```
post_MLR <- as.mcmc(posterior_MLR)
```

```
synthesize <- function(X, index, n){
  synth_Y=vector(mode="numeric", length = n)
  for(i in 1:n){
  mean_Y <- post_MLR[index, "beta0"] + X[i,1] * post_MLR[index, "beta1"] + X[i,2] *post_MLR[index, "beta
    X[i,3]*post_MLR[index, "beta3"] + X[i,4] *post_MLR[index, "beta4"] + X[i,5] *post_MLR[index, "beta5
    X[i,6]*post_MLR[index, "beta6"] + X[i,7] *post_MLR[index, "beta7"]
  synth_Y[i]<- rnorm(1, mean_Y, post_MLR[index, "sigma"])
  }
  synthetic_frame<-as.data.frame(X, row.names = NULL, optional = FALSE)
  synthetic_frame<-add_column(synthetic_frame, synth_Y)
  return(synthetic_frame)
}
```

```
n <- dim(CESample)[1]
matrix_of_X<-matrix(nrow=n, ncol=7)
matrix_of_X[,1]<-as.vector(CESample$Log_TotalExpSTD)
matrix_of_X[,2]<-as.vector(CESample$Rural)
matrix_of_X[,3]<-as.vector(CESample$Race_Black)
matrix_of_X[,4]<-as.vector(CESample$Race_NA)
matrix_of_X[,5]<-as.vector(CESample$Race_Asian)
matrix_of_X[,6]<-as.vector(CESample$Race_PI)
matrix_of_X[,7]<-as.vector(CESample$Race_M)
```

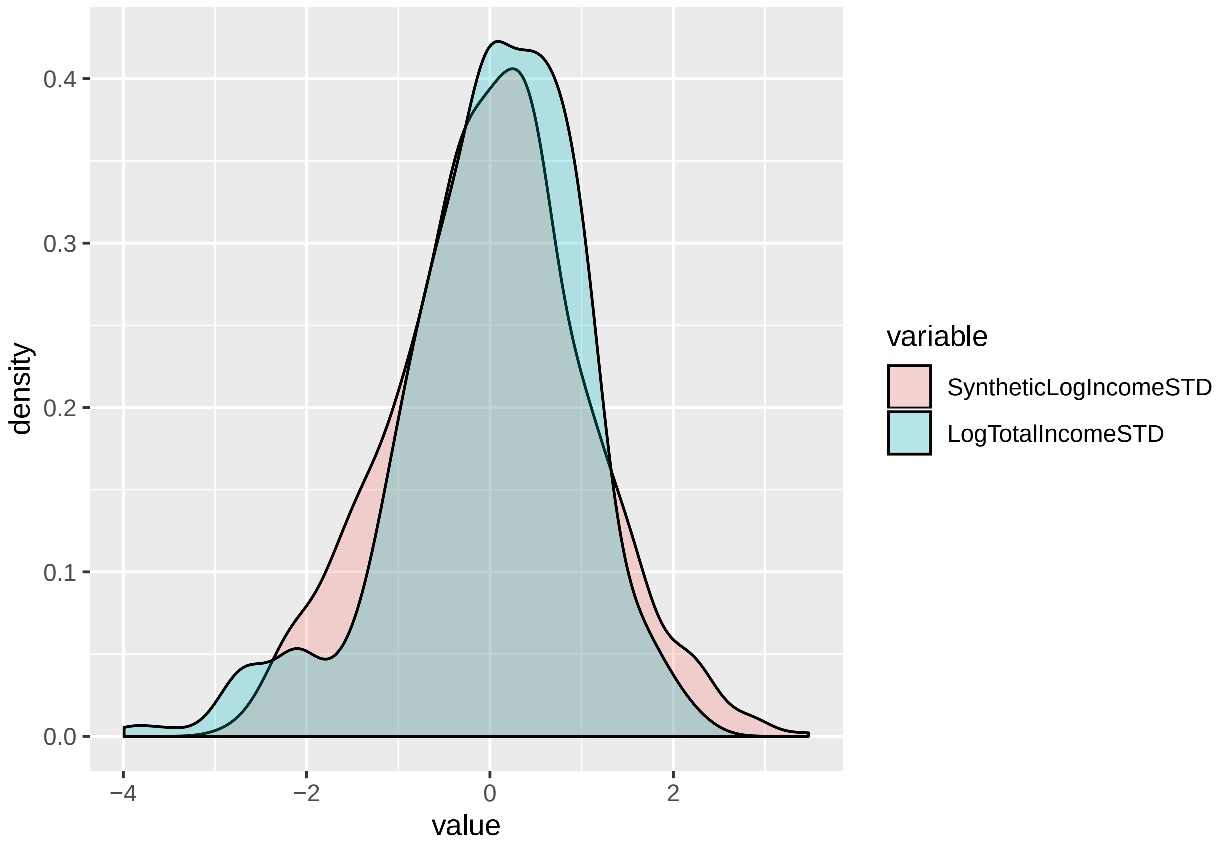
```
synthetic_new <- synthesize(matrix_of_X, 1, n)
names(synthetic_new) <- c("logExpenditure", "Rural", "Black", "Native American", "Asian", "Pacific Islar
```

```
SyntheticData <- data.frame(synthetic_new$logIncome_syn, CESample$Log_TotalIncomeSTD)
names(SyntheticData) = c("SyntheticLogIncomeSTD", "LogTotalIncomeSTD")
```

```
data<- melt(SyntheticData)
```

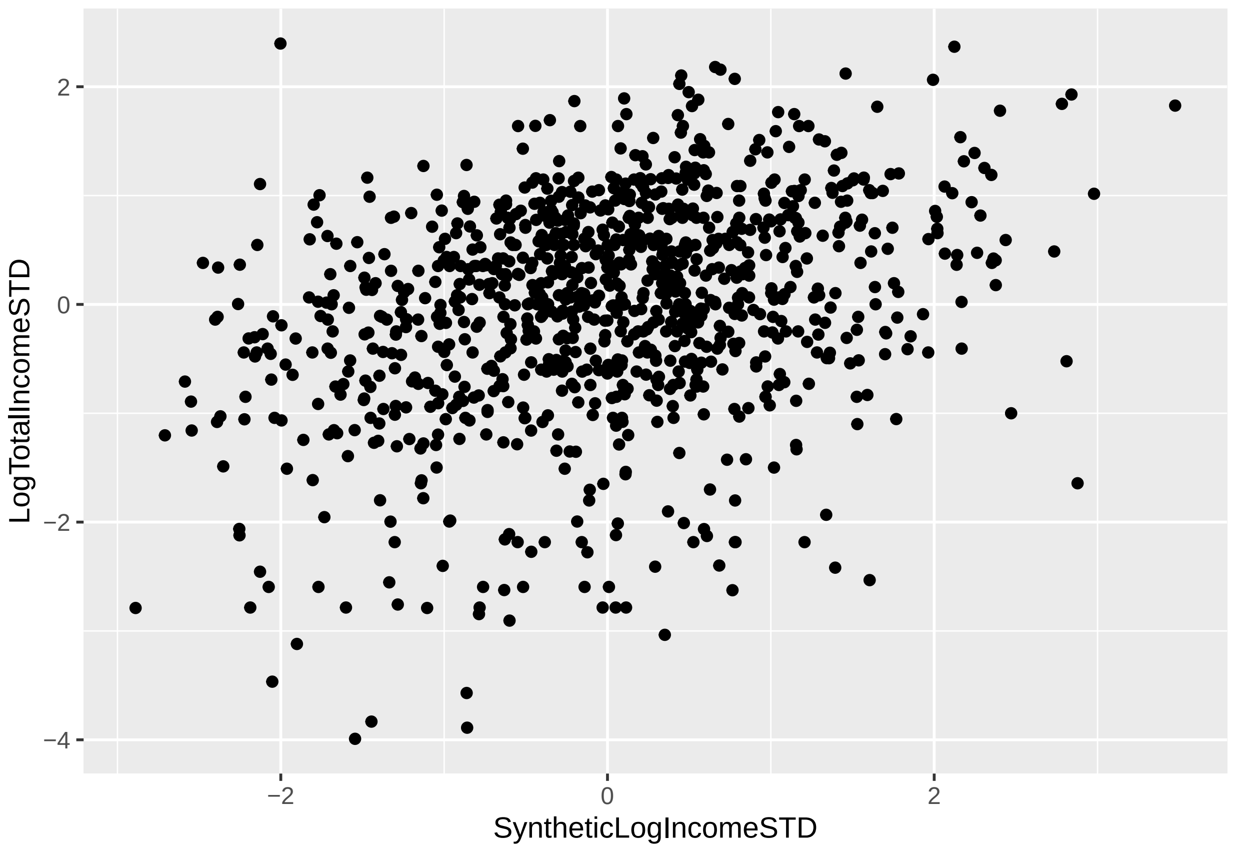
```
## Using  as id variables
```

```
ggplot(data,aes(x=value, fill=variable)) + geom_density(alpha=0.25)
```



Here we can see the effect our model has on the data. The distribution curve, while similar to the data, looks to follow a normal curve more faithfully.

```
ggplot(SyntheticData, aes(x=SyntheticLogIncomeSTD, y=LogTotalIncomeSTD)) +
  geom_point()
```



```
summary = summarize_all(SyntheticData, .funs=c(mean, median))
names(summary) = c("SyntheticLogIncomeMean", "SyntheticLogIncomeMedian","LogTotalIncomeMean", "LogTotal
summary
```

```
##   SyntheticLogIncomeMean SyntheticLogIncomeMedian LogTotalIncomeMean
## 1           -0.003130824             1.086035e-16         0.04843122
##   LogTotalIncomeMedian
## 1           0.09593896
```

These look farily close to each other to me.