```r
library(runjags)
library(coda)
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
data<- read.csv("CEdata.csv")
urban<- data$UrbanRural
inc<- data$Income
race<- data$Race
exp<- data$Expenditure
data$logexp<- log(data$Expenditure)
data$loginc<- log(data$Income)
data$rural= fastDummies::dummy_cols(data$UrbanRural)[,names(fastDummies::dummy_cols(data$UrbanRural))==
data$raceblack= fastDummies::dummy_cols(data$Race)[,names(fastDummies::dummy_cols(data$Race)) == ".data
data$racena= fastDummies::dummy_cols(data$Race)[,names(fastDummies::dummy_cols(data$Race)) == ".data_3"]
data$raceasian= fastDummies::dummy_cols(data$Race)[,names(fastDummies::dummy_cols(data$Race)) == ".data
data$racepi= fastDummies::dummy_cols(data$Race)[,names(fastDummies::dummy_cols(data$Race)) == ".data_5"]
data$racem= fastDummies::dummy_cols(data$Race)[,names(fastDummies::dummy_cols(data$Race)) == ".data_6"]
```

```r
modelString <-"
model {
for (i in 1:N){
y[i] ~ dnorm(beta0 + beta1*x_inc[i] + beta2*x_rural[i] +
beta3*x_raceb[i] + beta4*x_racen[i] +
beta5*x_racea[i] + beta6*x_racep[i] +
beta7*x_racem[i], invsigma2)
}
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))
}
"
y= as.vector(data$logexp)
x_inc= as.vector(data$loginc)
x_rural= as.vector(data$rural)
x_raceb= as.vector(data$raceblack)
x_racen= as.vector(data$racena)
```

```
x_racea= as.vector(data$raceasian)
x_racep= as.vector(data$racepi)
x_racem= as.vector(data$racem)
N= length(y)
```

```
the_data<- list("y" = y, "x_inc"= x_inc,
                "x_rural"= x_rural, "x_raceb"= x_raceb,
                "x_racen"= x_racen, "x_racea"= x_racea,
                "x_racep" = x_racep, "x_racem" = x_racem,
                "N" = N,
                "mu0" = 0, "g0" = 1, "mu1" = 0, "g1" = 1,
                "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)

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))
}
posterior<- 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 = 5,
                     inits = initsfunction)
```

```
## Loading required namespace: rjags
```

```
## Compiling rjags model...
## Calling the simulation using the rjags method...
## Note: the model did not require adaptation
## Burning in the model for 5000 iterations...
## Running the model for 25000 iterations...
## Simulation complete
## Calculating summary statistics...
```

```
## Warning: Convergence cannot be assessed with only 1 chain
```

```
## Finished running the simulation
```

```
posterior
```

```
##
## JAGS model summary statistics from 5000 samples (thin = 5; adapt+burnin = 6000):
##
##         Lower95    Median   Upper95     Mean       SD Mode      MCerr
## beta0    3.5585    4.0084    4.4503   4.0153  0.22884   --   0.018834
```

```
## beta1   0.38885   0.42709   0.46689  0.42707 0.019843   --   0.001608
## beta2  0.069024   0.26685   0.48634  0.26935  0.10473   --  0.0041164
## beta3  -0.34665  -0.19472 -0.056693 -0.19451 0.073547   --  0.0010401
## beta4  -0.49349 0.0090753    0.5403  0.01013  0.26292   --  0.0037182
## beta5 -0.064514   0.15892   0.39297  0.15914  0.11846   --  0.0016753
## beta6  -0.47796  0.088754   0.63652  0.08928  0.28506   --  0.0040314
## beta7  -0.32221  0.037415   0.36707 0.039882  0.17667   --  0.0024984
## sigma   0.69011   0.72156   0.75264  0.72182 0.015937   -- 0.00022538
##
##       MC%ofSD SSeff      AC.50 psrf
## beta0     8.2   148    0.55384   --
## beta1     8.1   152    0.52638   --
## beta2     3.9   647   0.081115   --
## beta3     1.4  5000   0.013834   --
## beta4     1.4  5000  -0.022457   --
## beta5     1.4  5000  -0.006785   --
## beta6     1.4  5000  -0.013474   --
## beta7     1.4  5000 -0.0091476   --
## sigma     1.4  5000 -0.0028526   --
##
## Total time taken: 13.5 seconds
```

```
post<- as.mcmc(posterior)
```

```
syn<- function(X, index, n){
  mean_Y<- post[index, "beta0"] + X$x_inc * post[index, "beta1"] + X$x_rural * post[index, "beta2"] + X
  syny<- rnorm(n,mean_Y, post[index,"sigma"])
  data.frame(X$x_inc, syny)
}
```

2i.

```
set.seed(123)
m<- 20
n<- dim(data)[1]
newsyn<- vector("list",m)
new<- data.frame(x_inc, x_rural, x_raceb, x_racen, x_racea, x_racep, x_racem)
for (i in 1:m){
  synhe <- syn(new, 4980+i, n)
  names(synhe) <- c("oriinc", "syninc")
  newsyn[[i]] <- synhe
}
```

ii.

```
mean <- c()
median <- c()
variance <- c()
for (i in 1:m){
  mean[i] = mean(newsyn[[i]]$syninc)
  median[i] = median(newsyn[[i]]$syninc)
  variance[i] = var(newsyn[[i]]$syninc)
  print(lm(data$logexp ~ newsyn[[i]]$syninc))
}
```

```
##
## Call:
```

```
## lm(formula = data$logexp ~ newsyn[[i]]$syninc)
##
## Coefficients:
##        (Intercept)  newsyn[[i]]$syninc
##             5.8213              0.3358
##
##
## Call:
## lm(formula = data$logexp ~ newsyn[[i]]$syninc)
##
## Coefficients:
##        (Intercept)  newsyn[[i]]$syninc
##             5.9501              0.3215
##
##
## Call:
## lm(formula = data$logexp ~ newsyn[[i]]$syninc)
##
## Coefficients:
##        (Intercept)  newsyn[[i]]$syninc
##             6.0856              0.3094
##
##
## Call:
## lm(formula = data$logexp ~ newsyn[[i]]$syninc)
##
## Coefficients:
##        (Intercept)  newsyn[[i]]$syninc
##             6.0273              0.3128
##
##
## Call:
## lm(formula = data$logexp ~ newsyn[[i]]$syninc)
##
## Coefficients:
##        (Intercept)  newsyn[[i]]$syninc
##             5.5646              0.3674
##
##
## Call:
## lm(formula = data$logexp ~ newsyn[[i]]$syninc)
##
## Coefficients:
##        (Intercept)  newsyn[[i]]$syninc
##             5.6332              0.3599
##
##
## Call:
## lm(formula = data$logexp ~ newsyn[[i]]$syninc)
##
## Coefficients:
##        (Intercept)  newsyn[[i]]$syninc
##              5.634               0.358
##
```

```
##
## Call:
## lm(formula = data$logexp ~ newsyn[[i]]$syninc)
##
## Coefficients:
##        (Intercept)  newsyn[[i]]$syninc
##             5.4922              0.3753
##
##
## Call:
## lm(formula = data$logexp ~ newsyn[[i]]$syninc)
##
## Coefficients:
##        (Intercept)  newsyn[[i]]$syninc
##             5.8257              0.3366
##
##
## Call:
## lm(formula = data$logexp ~ newsyn[[i]]$syninc)
##
## Coefficients:
##        (Intercept)  newsyn[[i]]$syninc
##             6.1370              0.3031
##
##
## Call:
## lm(formula = data$logexp ~ newsyn[[i]]$syninc)
##
## Coefficients:
##        (Intercept)  newsyn[[i]]$syninc
##             5.6723              0.3543
##
##
## Call:
## lm(formula = data$logexp ~ newsyn[[i]]$syninc)
##
## Coefficients:
##        (Intercept)  newsyn[[i]]$syninc
##             6.0639              0.3094
##
##
## Call:
## lm(formula = data$logexp ~ newsyn[[i]]$syninc)
##
## Coefficients:
##        (Intercept)  newsyn[[i]]$syninc
##             5.8214              0.3379
##
##
## Call:
## lm(formula = data$logexp ~ newsyn[[i]]$syninc)
##
## Coefficients:
##        (Intercept)  newsyn[[i]]$syninc
```

```
##             5.9021              0.3279
##
##
## Call:
## lm(formula = data$logexp ~ newsyn[[i]]$syninc)
##
## Coefficients:
##        (Intercept)  newsyn[[i]]$syninc
##             5.7947              0.3397
##
##
## Call:
## lm(formula = data$logexp ~ newsyn[[i]]$syninc)
##
## Coefficients:
##        (Intercept)  newsyn[[i]]$syninc
##             6.0275              0.3143
##
##
## Call:
## lm(formula = data$logexp ~ newsyn[[i]]$syninc)
##
## Coefficients:
##        (Intercept)  newsyn[[i]]$syninc
##             5.7870              0.3431
##
##
## Call:
## lm(formula = data$logexp ~ newsyn[[i]]$syninc)
##
## Coefficients:
##        (Intercept)  newsyn[[i]]$syninc
##             5.6084              0.3613
##
##
## Call:
## lm(formula = data$logexp ~ newsyn[[i]]$syninc)
##
## Coefficients:
##        (Intercept)  newsyn[[i]]$syninc
##             5.7241              0.3495
##
##
## Call:
## lm(formula = data$logexp ~ newsyn[[i]]$syninc)
##
## Coefficients:
##        (Intercept)  newsyn[[i]]$syninc
##             5.7862              0.3411
```

iii

```
bmean= sum(mean - sum(mean)/m)^2/(m-1)
ubar= sum(var(mean))/m
tmean= (1 + (m^(-1)))*bmean - ubar
```

```
qbar= sum(mean)/m
qbar
```

```
## [1] 8.777655
```

```
tmean
```

```
## [1] -3.90537e-05
```

```
u_median = var(median)
qbar_median = sum(median)/m
b_median = sum(median - qbar_median)^2/(m-1)
ubar_median = sum(u_median)/m
T_median = (1 + (m^-1))*b_median - ubar_median
qbar_median
```

```
## [1] 8.798681
```

```
T_median
```

```
## [1] -5.842937e-05
```

```
u_var = var(variance)
qbar_var = sum(variance)/m
b_var = sum(variance - qbar_var)^2/(m-1)
ubar_var = sum(u_var)/m
T_var = (1 + (m^-1))*b_var - ubar_var
qbar_var
```

```
## [1] 0.8103088
```

```
T_var
```

```
## [1] -7.40848e-05
```

3.

```
ls = quantile(newsyn[[1]]$syninc, 0.025)
us = quantile(newsyn[[1]]$syninc, 0.975)
lo = quantile(newsyn[[1]]$oriinc, 0.025)
uo = quantile(newsyn[[1]]$oriinc, 0.975)
l = max(ls, lo)
u = min(us, uo)
interval = (u - l) / (2 * (uo - lo)) + (u - l)/ (2 * (us - ls))
interval
```

```
##     97.5%
## 0.7213914
```

4.

```
data2<- read.csv("frmgham2.csv")
data2$sexd= fastDummies::dummy_cols(data2$SEX)[,names(fastDummies::dummy_cols(data2$SEX))== ".data_1"]
data2$educhs= fastDummies::dummy_cols(data2$educ)[,names(fastDummies::dummy_cols(data2$educ)) == ".data
data2$educsc= fastDummies::dummy_cols(data2$educ)[,names(fastDummies::dummy_cols(data2$educ)) == ".data
data2$educc= fastDummies::dummy_cols(data2$educ)[,names(fastDummies::dummy_cols(data2$educ)) == ".data_
data2 <- na.omit(data2)
```

```
modelString2 <-"
model {
for (i in 1:N){
```

```
y[i] ~ dnorm(beta0 + beta1*x_age[i] + beta2*x_sexd[i] +
beta3*x_educhs[i] + beta4*x_educsc[i] +
beta5*x_educc[i], invsigma2)
}
beta0 ~ dnorm(mu0, g0)
beta1 ~ dnorm(mu1, g1)
beta2 ~ dnorm(mu2, g2)
beta3 ~ dnorm(mu3, g3)
beta4 ~ dnorm(mu4, g4)
beta5 ~ dnorm(mu5, g5)
invsigma2 ~ dgamma(a, b)
sigma <- sqrt(pow(invsigma2, -1))
}
"
y= as.vector(data2$BMI)
x_age= as.vector(data2$AGE)
x_sexd= as.vector(data2$sexd)
x_educhs= as.vector(data2$educhs)
x_educsc= as.vector(data2$educsc)
x_educc= as.vector(data2$educc)
N= length(y)
```

```
the_data<- list("y" = y, "x_age"= x_age,
                "x_sexd"= x_sexd, "x_educhs"= x_educhs,
                "x_educsc"= x_educsc, "x_educc"= x_educc,
                "N" = N,
                "mu0" = 0, "g0" = 1, "mu1" = 0, "g1" = 1,
                "mu2" = 0, "g2" = 1, "mu3" = 0, "g3" = 1,
                "mu4" = 0, "g4" = 1, "mu5" = 0, "g5" = 1,
                "a" = 1, "b" = 1)

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))
}
posterior<- run.jags(modelString2,
                     n.chains = 1,
                     data = the_data,
                     monitor = c("beta0", "beta1", "beta2",
                                 "beta3", "beta4", "beta5",
                                 "sigma"),
                     adapt = 1000,
                     burnin = 5000,
                     sample = 5000,
                     thin = 1,
                     inits = initsfunction)
```

```
## Compiling rjags model...
## Calling the simulation using the rjags method...
## Note: the model did not require adaptation
## Burning in the model for 5000 iterations...
```

```
## Running the model for 5000 iterations...
## Simulation complete
## Calculating summary statistics...
```

```
## Warning: Convergence cannot be assessed with only 1 chain
```

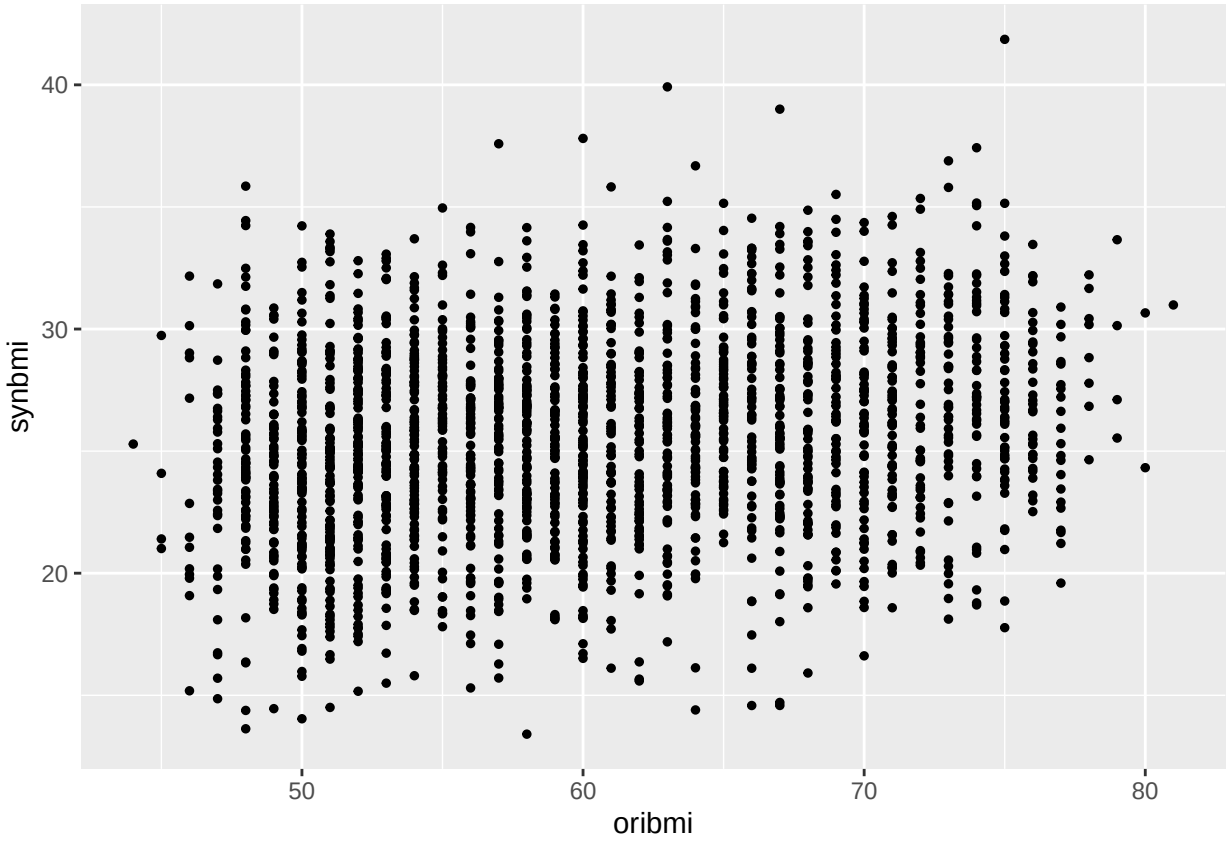
```
## Finished running the simulation
```

```
posterior
```

```
##
## JAGS model summary statistics from 5000 samples (adapt+burnin = 6000):
##
##        Lower95   Median  Upper95     Mean        SD Mode     MCerr MC%ofSD
## beta0   17.904   19.145   20.382   19.152   0.63679   --  0.088165    13.8
## beta1 0.082442  0.10161  0.12175  0.10176 0.0099933   -- 0.0014264    14.3
## beta2  0.69838    1.008   1.3397   1.0091   0.16627   -- 0.0037247     2.2
## beta3 -0.16589  0.21878  0.60871  0.22017   0.19887   -- 0.0070986     3.6
## beta4  -1.0781 -0.62646 -0.17261 -0.62898   0.23253   -- 0.0052512     2.3
## beta5 -0.90959 -0.40818  0.08918 -0.41333   0.25698   -- 0.0052736     2.1
## sigma   3.8519   3.9798   4.1022   3.9817  0.064504   -- 0.0027398     4.2
##
##       SSeff       AC.10 psrf
## beta0    52     0.80953   --
## beta1    49     0.80366   --
## beta2  1993  -0.0085434   --
## beta3   785    0.016964   --
## beta4  1961 -0.00021548   --
## beta5  2375  -0.0061096   --
## sigma   554    0.094295   --
##
## Total time taken: 5.9 seconds
```

```
post<- as.mcmc(posterior)
syn<- function(X, index, n){
  mean_Y<- post[index, "beta0"] + X$x_age * post[index, "beta1"] + X$x_sexd * post[index, "beta2"] + X$
  syny<- rnorm(n,mean_Y, post[index,"sigma"])
  data.frame(X$x_age, syny)
}
n<- dim(data2)[1]
frame<- data.frame(y, x_age, x_sexd, x_educhs, x_educsc, x_educc)
syndata<- syn(frame, 1, n)
names(syndata)<- c("oribmi", "synbmi")
```

```
ggplot(syndata, aes(x= oribmi, y= synbmi)) + geom_point(size = 1)
```


40
30
20
synbmi
50
60
70
80
oribmi