Bayesian Cities Model

Dina Sinclair

June 9, 2018

This file aims to walk the reader through a 1 layer (1D) REM cities model where sample size per city J is not constant.

NOTE: This file is out of date but the intial matrix descriptions could be useful later.

```r
library("rstan")
```

Inputs

Notation

Given

$$Y = (Y_1, \dots, Y_i)$$

where each Y_i has its own given $\sigma_{i,j}$,assume $Y \sim N(\theta_i, \sigma_{i,j}^2)$.

In matrix form we'll be using

$$Y = \begin{bmatrix} Y_{study1} \\ \vdots \\ Y_{studyN} \end{bmatrix} = \begin{bmatrix} Y_{11} & \dots & Y_{1J} \\ & \ddots & \\ Y_{N1} & \dots & Y_{NJ} \end{bmatrix}$$

Here, we'll use Y_1 to refer to the vector $(Y_{11}, \dots, Y_{1J})$ and assume for now that all studies have the same sample size J (Not very accurate, we'll have to change that later!).

Arbitrary Example Inputs

For now, we'll use that classic arbitrary dataset I've been using We'll hope that we can recover mu and tauSq correctly!

```r
# Set input parameters
set.seed(17)
mu <- 10
tauSq <- 2
I <- 3
J <- 3
sigmaSq <- c(0.1,0.4,.2,0.1,0.2,0.2,0.5,0.6,0.3)

# Calculate theta, reshape sigmaSq
theta <- rnorm(I, mu, tauSq)
sigmaSq <- matrix(sigmaSq, I , J)

# Calculate and reshape Y
Y_vect <- numeric()
for(i in 1:I){
  Y_vect <- c(Y_vect, rnorm(J,mean=theta[i],sd=sigmaSq[i,]))
}
Y <- matrix(Y_vect , I, J, byrow = TRUE)
```

```r
# Save our input data together in a list
basic_dat_generated <- list(J,I,Y,sigmaSq)

# Display the generated input data for reference
print("Theta:")
```

```
## [1] "Theta:"
```

```r
theta
```

```
## [1] 7.969983 9.840727 9.534026
```

```r
print("SigmaSq:")
```

```
## [1] "SigmaSq:"
```

```r
sigmaSq
```

```
##      [,1] [,2] [,3]
## [1,]  0.1  0.1  0.5
## [2,]  0.4  0.2  0.6
## [3,]  0.2  0.2  0.3
```

```r
print("Y:")
```

```
## [1] "Y:"
```

```r
Y
```

```
##           [,1]      [,2]     [,3]
## [1,]  7.888256  8.047192 7.887177
## [2,] 10.229876 10.184033 9.993869
## [3,]  9.607342  9.770184 9.726984
```

Step 1: Calculate θ_i, μ, τ^2 using the original priors Y

We're assuming a **random effects model**, that is that

$$\theta_i \sim N(\mu, \tau^2) \text{ and } Y_i \sim N(\theta_i, \sigma_{ij}^2)$$

To make use of this assumption, we need to estimate scalars μ and τ^2 along with the vector $\theta = (\theta_1, \ldots, \theta_N)$. We can do this using stan!

```r
fit <- stan(file = '../randomEffectsModel1D.stan',
            data = basic_dat_generated,
            iter = 1000, chains = 2)
```

```
## In file included from C:/Users/Dina/Documents/R/win-library/3.3/BH/include/boost/config.hpp:39:0,
##                  from C:/Users/Dina/Documents/R/win-library/3.3/BH/include/boost/math/tools/config.hp
##                  from C:/Users/Dina/Documents/R/win-library/3.3/StanHeaders/include/stan/math/rev/co
##                  from C:/Users/Dina/Documents/R/win-library/3.3/StanHeaders/include/stan/math/rev/co
##                  from C:/Users/Dina/Documents/R/win-library/3.3/StanHeaders/include/stan/math/rev/co
##                  from C:/Users/Dina/Documents/R/win-library/3.3/StanHeaders/include/stan/math/rev/ma
##                  from C:/Users/Dina/Documents/R/win-library/3.3/StanHeaders/include/stan/math.hpp:4,
##                  from C:/Users/Dina/Documents/R/win-library/3.3/StanHeaders/include/src/stan/model/mo
##                  from file8783bc0335f.cpp:8:
## C:/Users/Dina/Documents/R/win-library/3.3/BH/include/boost/config/compiler/gcc.hpp:186:0: warning: "
##  #  define BOOST_NO_CXX11_RVALUE_REFERENCES
##  ^
## <command-line>:0:0: note: this is the location of the previous definition
```

```
## failed to create the sampler; sampling not done
```

```r
fit
```

```
## Stan model 'randomEffectsModel1D' does not contain samples.
```

Step 2: Pick K studies to pilot, calculate Y'_k

Theoretically

First, pick K studies k to update through new information. For these studies, we'll assume a **fixed effects model** and draw J new samples, imagining that this is a new study done in the same place (with fixed $\mu_k = \theta_k$ from step 1) that gets us a new dataset. So we can define

$$Y'_k = (Y'_{k1}, \dots, Y'_{kJ})$$

from that fixed effects model. We need to know σ_{kj} for these new studies. For now we will arbitrarily decide that $\sigma_{kj} = \frac{1}{5}\sigma_{ij}$ (AKA that these new studies done will have a fifth of the error we were encountering in the first study). So for $i \in K$,

$$Y'_i \sim N(\theta_i, \frac{1}{5}\sigma_{i,j})$$

Concretely

Here we're first going to go with the boring simple version where the subset K is just k=2. So we keep Y_1, Y_3 unchanged but need to make a Y'_2 using the generated θ_2. We use an arbitrary 5x smaller sigma in Y'_2 than in Y_2 as noted in the norm above

```r
# Define set K of cities to try the pilot on and collect more data
K <- c(2)

# Calculate new data for all K new pilots
params <- extract(fit)
```

```
## Stan model 'randomEffectsModel1D' does not contain samples.
```

```r
for (k in K){
  print("Theta chosen for new pilot:")
  print(mean(params$theta[,k]))
  new_draw <- rnorm( J , mean = mean(params$theta[,k]) , sd = (1/5)*sigmaSq[k,] )
  print("New Draw Results:")
  print(new_draw)
}
```

```
## [1] "Theta chosen for new pilot:"
```

```
## Warning in mean.default(params$theta[, k]): argument is not numeric or
## logical: returning NA
```

```
## [1] NA
```

```
## Warning in mean.default(params$theta[, k]): argument is not numeric or
## logical: returning NA
## Warning in rnorm(J, mean = mean(params$theta[, k]), sd = (1/5) *
## sigmaSq[k, : NAs produced
```

```
## [1] "New Draw Results:"
## [1] NaN NaN NaN
```

This leaves me with some **lingering questions**: * Should we be calculating J new draws and getting a complete new dataset, or calculating one more draw only to get a new mu? * If we calculate only one more draw, what do we use as that draw's sigmaSq value? 1/5 of the average of the previous sigmaSq values?

Step 3: Update all Y values to get Y^U

Once those k new values get calculated, update by combining to get

$$update(Y_k, Y_k') = Y_k^U$$

This draws on the idea that the mean can be combined with the equation (where A is a matrix form of Y and B is a matrix form of σ^2)

$$mean = \frac{A\tau^2 + B\mu}{B + \tau^2}$$

and variance can be combined using

$$var = \frac{B\tau^2}{B + \tau^2} = \frac{1}{\frac{1}{\tau^2} + \frac{1}{B}}$$

While I have values for A,B,τ^2 and μ, I'm confused.

Specifically, my quesitons are: * These equations don't seem to check out dimensionally. A and B are matrices, but I was under the impression that μ and τ^2 are scalars. Are μ and τ^2 vectors? * It looks to me like this equation would give me a scalar mean and variance. Does that mean that A and B are scalars from one dist, and μ and τ are scalars from the other dist? That would make sense. But then I'm getting a result that's also just a pair of scalars rather than a dataset. Don't I need a dataset to run that final bayesian model in the next step? Would I need to generate that data? * How do I decide on a scalar sigmaSq for Y if sigmaSq varies across individuals?

What I've done below as a result of these questions is coded up a simple scalar -> scalar version and will ask you about clarifications later.

Concretely

Now that we have the generated new information Y_k', we need to combine it with the old information Y_k. We can do this by combining the means weighted by their standard deviations. I thought Eva described how to do this in the How Much Can We Generalize paper section 2.2 equation (8) as well as in this suff_theta code she wrote in her multiDrawRandomGrouping.R code, but the more I look at both that equation and this code the more questions I have.

```r
# Eva's Code
suff_theta <- function(Y, mu, tau2, sigma2){
  #vectorize for the whole sample size instead of one sample each run
  n <- length(Y)
  m <- length(tau2)
  A <- matrix(Y, ncol = n, nrow = m, byrow = TRUE)
  B <- matrix(sigma2, ncol = n, nrow = m, byrow = TRUE)
  mean <- (A*tau2 + B*mu) /  (B + tau2)
  var <- B*tau2 /( B+ tau2)
  #return the mean and variance for all e_i samples
  #in matrix with nrow = # of e_i and ncol = #samples
  return(list(mean = mean, var = var))
}

# My One-Dimensional Code
update_Y <- function(mu1, mu2, sigSq1, sigSq2){
  update_mu <- (mu1*sigSq2 + mu2*sigSq1)/(sigSq1 + sigSq2)
  update_sigSq <- (sigSq1*sigSq2)/(sigSq1 + sigSq2)
  return(list(mu = update_mu, sigmaSq = update_sigSq))
}

# Using my 1D code and the average(??) sigma from Y as sigSq1
# Calculate new data for all K new pilots
for (k in K){
  print("Theta chosen from new pilot:")
  print(mean(params$theta[,k]))
  new_draw <- rnorm( 1 , mean = mean(params$theta[,k]) , sd = (1/5)*sigmaSq[k,] )
  print("New Draw Result:")
  print(new_draw)
  print("Average sigmaSq from original Y")
  avgSigSq <- mean(sigmaSq[k,])
  print(avgSigSq)
  Y_U <- update_Y(mean(params$theta[,k]),new_draw,avgSigSq,avgSigSq/5)
}
```

```
## [1] "Theta chosen from new pilot:"

## Warning in mean.default(params$theta[, k]): argument is not numeric or
## logical: returning NA

## [1] NA

## Warning in mean.default(params$theta[, k]): argument is not numeric or
## logical: returning NA

## Warning in rnorm(1, mean = mean(params$theta[, k]), sd = (1/5) *
## sigmaSq[k, : NAs produced

## [1] "New Draw Result:"
## [1] NaN
## [1] "Average sigmaSq from original Y"
## [1] 0.4

## Warning in mean.default(params$theta[, k]): argument is not numeric or
## logical: returning NA
```

```r
print("updated Y")
```

```
## [1] "updated Y"
```

```r
Y_U
```

```
## $mu
## [1] NA
##
## $sigmaSq
## [1] 0.06666667
```

Step 4: Use Y^U to get final θ^U

and once we have all the updated Y_i^U we can use them to get θ_i^U based off of μ^U, τ^{U2} with $Y^U \sim N(\theta_i^U, \sigma_{i,j}^U)$

```r
updated_dat_generated <- list(J, I, Y_U, sigmaSq_U)
fit_updated <- stan(file = '../randomEffectsModel1D.stan',
            data = updated_dat_generated,
            iter = 1000, chains = 2)
fit_updated
```