Hierarchical Bayes

for Generalized Linear Mixed Effect Model

30 July, 2020

The LMM Example

NHTS data

The data source

```
NHTS2017 <- (read.csv("~/trippub.csv"))[,c(1,30,62,64,69,72,85)]
# NHTS2017 <- NHTS2017[complete.cases(NHTS2017),]
NHTS2017 <- NHTS2017[NHTS2017$VMT_MILE!=-1&NHTS2017$HHFAMINC>=0&NHTS2017$HH_CBSA!="XXXXX", ]
nhts2017 <- NHTS2017[sample(nrow(NHTS2017), 10000,replace =F), ]
save(nhts2017, file="nhts2017.RData")
```

Select “HOUSEID”, “VMT_MILE”, and five regressors

excluded the zero-miles VMT, negative household income, and unknown CBSA id (XXXXX)

Sample 10000 observations from the original data

Add a new column “log_VMT”

```
load("nhts2017.RData")
nhts2017$log_VMT <- log(nhts2017[,2])
str(nhts2017)
```

```
## 'data.frame':    10000 obs. of  8 variables:
##  $ HOUSEID : int  40395139 40580647 30333668 30141033 30154959 30457173 30367388 40151648 40276032 4
##  $ VMT_MILE: num  1.76 9.54 38.65 12.47 28.25 ...
##  $ HHSIZE  : int  2 2 3 3 2 1 4 3 1 2 ...
##  $ HHFAMINC: int  4 5 7 11 6 10 11 5 3 3 ...
##  $ WRKCOUNT: int  1 1 2 2 1 1 3 1 0 2 ...
##  $ LIF_CYC : int  10 10 6 6 10 1 6 3 9 2 ...
##  $ HH_CBSA : Factor w/ 53 levels "12060","12420",..: 1 22 3 9 6 12 47 12 7 45 ...
##  $ log_VMT : num  0.565 2.256 3.654 2.523 3.341 ...
```

```
summary(nhts2017)
```

```
##     HOUSEID             VMT_MILE            HHSIZE          HHFAMINC         WRKCOUNT        LIF_CYC
##  Min.   :30000271   Min.   :   0.001   Min.   : 1.00   Min.   : 1.00   Min.   :0.00   Min.   : 1.00
##  1st Qu.:30263716   1st Qu.:   1.870   1st Qu.: 2.00   1st Qu.: 5.00   1st Qu.:1.00   1st Qu.: 2.00
##  Median :30525842   Median :   4.262   Median : 2.00   Median : 7.00   Median :1.00   Median : 5.00
##  Mean   :35001678   Mean   :   9.993   Mean   : 2.54   Mean   : 7.07   Mean   :1.34   Mean   : 5.34
##  3rd Qu.:40365461   3rd Qu.:  10.737   3rd Qu.: 3.00   3rd Qu.: 9.00   3rd Qu.:2.00   3rd Qu.: 9.00
##  Max.   :40794179   Max.   :1876.680   Max.   :11.00   Max.   :11.00   Max.   :7.00   Max.   :10.00
##
```

```
table(nhts2017$HH_CBSA)
```

```
##
## 12060 12420 12580 13820 14460 15380 16740 16980 17140 17460 18140 19100 19740 19820 24340 25540 2642
##   526   397    61    15    82   153   166   174    35    44    34  1735    44    55    19    18   905
```

There are $m = 52$ levels of CBSA.

$\mathbf{Y}_j$ is a n_j Vector.

$\mathbf{X}_j$ is a $n_j \times p$ Matrix

```
ids<-sort(unique(nhts2017$HH_CBSA))
m<-length(ids)
Y<-list() ; X<-list() ; N<-NULL
for(j in 1:m)
{
  Y[[j]]<-nhts2017[nhts2017$HH_CBSA==ids[j],8]
  N[j]<- sum(nhts2017$HH_CBSA==ids[j])
  xj<-nhts2017[nhts2017$HH_CBSA==ids[j], 4]
  xj<-(xj-mean(xj))
  X[[j]]<-cbind( rep(1,N[j]), xj)
}
```

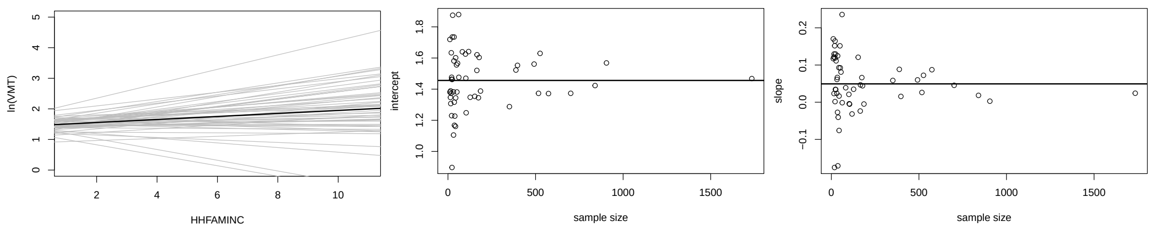
OLS fits

```
S2.LS<-BETA.LS<-NULL
for(j in 1:m) {
  fit<-lm(Y[[j]]~-1+X[[j]] )
  BETA.LS<-rbind(BETA.LS,c(fit$coef))
  S2.LS<-c(S2.LS, summary(fit)$sigma^2)
}
```

The first panel plots least squares estimates of the regression lines for the 52 CBSA, along with an average of these lines in black. A large majority show an slight increase in expected VMT with increasing household income, although a few show a negative relationship.

The second and third panels of the figure relate the least squares estimates to sample size. Notice that CBSAs with the higher sample sizes have regression coefficients that are generally closer to the average, whereas CBSAs with extreme coefficients are generally those with low sample sizes. This phenomenon confirms that the smaller the sample size for the group, the more probable that unrepresentative data are sampled and an extreme least squares estimate is produced.

```
plot( range(nhts2017[,4]),c(0,5),type="n",xlab="HHFAMINC", ylab="ln(VMT)")
for(j in 1:m) {    abline(BETA.LS[j,1],BETA.LS[j,2],col="gray")  }

BETA.MLS<-apply(BETA.LS,2,mean)
abline(BETA.MLS[1],BETA.MLS[2],lwd=2)

plot(N,BETA.LS[,1],xlab="sample size",ylab="intercept")
abline(h= BETA.MLS[1],col="black",lwd=2)
plot(N,BETA.LS[,2],xlab="sample size",ylab="slope")
abline(h= BETA.MLS[2],col="black",lwd=2)
```



A hierarchical regression model

$$\mathbf{Y}_{i,j} = \boldsymbol{\beta_j^T x_{i,j}} + \varepsilon_{i,j} = \boldsymbol{\theta^T x_{i,j} + \gamma_j^T x_{i,j}} + \varepsilon_{i,j}$$

- mvnormal simulation

$\boldsymbol{\beta}_{1:m} \overset{iid}{\sim} N_p(\boldsymbol{\theta}, \Sigma)$, $\boldsymbol{\gamma}_{1:m} \overset{iid}{\sim} N_p(0, \Sigma)$

$\boldsymbol{\theta} \sim N_p(\boldsymbol{\mu_0}, \boldsymbol{\Lambda_0})$,

```
rmvnorm<-function(n,mu,Sigma)
{
  E<-matrix(rnorm(n*length(mu)),n,length(mu))
  t(  t(E%*%chol(Sigma)) +c(mu))
}
```

- Wishart simulation

$\Sigma \sim Inverse - Wishart(\eta_0, \boldsymbol{S_0^{-1}})$,

```
rwish<-function(n,nu0,S0)
{
  sS0 <- chol(S0)
  S<-array( dim=c( dim(S0),n ) )
  for(i in 1:n)
  {
     Z <- matrix(rnorm(nu0 * dim(S0)[1]), nu0, dim(S0)[1]) %*% sS0
     S[,,i]<- t(Z)%*%Z
  }
  S[,,1:n]
}
```

$\sigma^2 \sim Inverse - Gamma(\frac{1}{2}\nu_0, \frac{1}{2}\nu_0\sigma_0^2)$

MCMC

Full conditional distributions

$$\{\beta|\boldsymbol{y_j},\boldsymbol{X_j},\boldsymbol{\theta},\sigma^2,\Sigma\}\sim N_p\left([\Sigma^{-1}+\frac{1}{\sigma^2}\mathbf{X_j'X_j}]^{-1}[\Sigma^{-1}\boldsymbol{\theta}+\frac{1}{\sigma^2}\mathbf{X_j'y_j}],[\Sigma^{-1}+\frac{1}{\sigma^2}\mathbf{X_j'X_j}]^{-1}\right)$$

$$\{\boldsymbol{\theta}|\boldsymbol{\beta_{1:m}},\Sigma\}\sim N_p(\boldsymbol{\mu_m},\boldsymbol{\Lambda_m});\quad \Lambda_m=(\Lambda_0^{-1}+m\Sigma^{-1})^{-1};\quad \boldsymbol{\mu_m}=\Lambda_m(\Lambda_0^{-1}\boldsymbol{\mu_0}+m\Sigma^{-1}\bar{\boldsymbol{\beta}})$$

$$\{\Sigma|\boldsymbol{\theta},\boldsymbol{\beta_{1:m}}\}\sim Inverse-Wishart(\eta_0+m,[\boldsymbol{S_0}+\boldsymbol{S_\theta}]^{\boldsymbol{-1}});\quad \boldsymbol{S_\theta}=\sum_{j=1}^{m}\boldsymbol{(\beta_j-\theta)(\beta_j-\theta)^T}$$

$$\sigma^2\sim Inverse-Gamma(\frac{1}{2}[\nu_0+\sum n_j],\frac{1}{2}[\nu_0\sigma_0^2+\mathrm{SSR}]);\quad \mathrm{SSR}=\sum_{j=1}^{m}\sum_{i=1}^{n}(y_{i,j}-\boldsymbol{\beta_j^T x_{i,j}})^2$$

- Setup

take μ_0, the prior expectation of θ, to be equal to the average of the ordinary least squares regression estimates and the prior variance Λ_0 to be their sample covariance. Such a prior distribution represents the information of someone with unbiased but weak prior information.

```
p<-dim(X[[1]])[2]
theta<-mu0<-apply(BETA.LS,2,mean)
nu0<-1 ; s2<-s20<-mean(S2.LS)
eta0<-p+2 ; Sigma<-S0<-L0<-cov(BETA.LS) ; BETA<-BETA.LS
THETA.b<-S2.b<-NULL
iL0<-solve(L0) ; iSigma<-solve(Sigma)
Sigma.ps<-matrix(0,p,p)
SIGMA.PS<-NULL
BETA.ps<-BETA*0
BETA.pp<-NULL
set.seed(1)
mu0[2]+c(-1.96,1.96)*sqrt(L0[2,2])
## [1] -2.451  2.477
```

For example, a 95% prior confidence interval for the slope parameter θ_2 under this prior is (-2.451151, 2.477191), which is quite a large range when considering what the extremes of the interval imply in terms of average change in VMT per unit change in income. Similarly, we will take the prior sum of squares matrix S_0 to be equal to the covariance of the least squares estimate, but we'll take the prior degrees of freedom η_0 to be $p+2=4$, so that the prior distribution of Σ is reasonably diffuse but has an expectation equal to the sample covariance of the least squares estimates. Finally, we'll take σ_0^2 to be the average of the within-group sample variance but set $\nu_0=1$.

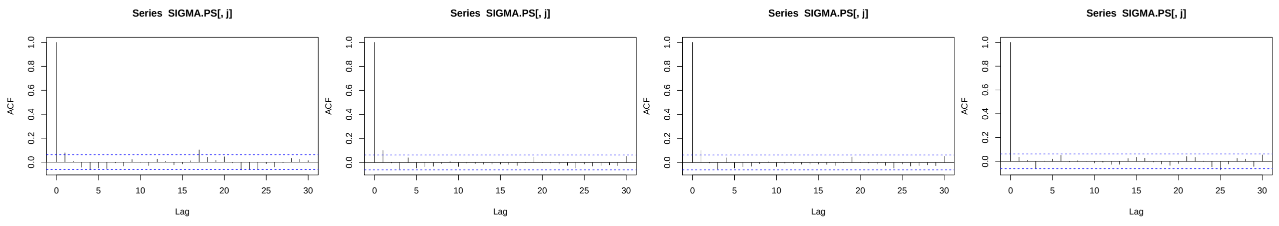
```
for(s in 1:10000) {
  ##update beta_j
  for(j in 1:m)
  {
    Vj<-solve( iSigma + t(X[[j]])%*%X[[j]]/s2 )
    Ej<-Vj%*%( iSigma%*%theta + t(X[[j]])%*%Y[[j]]/s2 )
    BETA[j,]<-rmvnorm(1,Ej,Vj)
  }
  ##

  ##update theta
  Lm<-  solve( iL0 +  m*iSigma )
  mum<- Lm%*%( iL0%*%mu0 + iSigma%*%apply(BETA,2,sum))
  theta<-t(rmvnorm(1,mum,Lm))
  ##

  ##update Sigma
  mtheta<-matrix(theta,m,p,byrow=TRUE)
  iSigma<-rwish(1, eta0+m, solve( S0+t(BETA-mtheta)%*%(BETA-mtheta) ) )
  ##

  ##update s2
  RSS<-0
  for(j in 1:m) { RSS<-RSS+sum( (Y[[j]]-X[[j]]%*%BETA[j,] )^2 ) }
  s2<-1/rgamma(1,(nu0+sum(N))/2, (nu0*s20+RSS)/2 )
  ##
  ##store results
  if(s%%10==0)
  {
   # cat(s,s2,"\n")
    S2.b<-c(S2.b,s2);THETA.b<-rbind(THETA.b,t(theta))
    Sigma.ps<-Sigma.ps+solve(iSigma) ; BETA.ps<-BETA.ps+BETA
    SIGMA.PS<-rbind(SIGMA.PS,c(solve(iSigma)))
    BETA.pp<-rbind(BETA.pp,rmvnorm(1,theta,solve(iSigma)) )
  }
  ##
}
```

- MCMC diagnostics

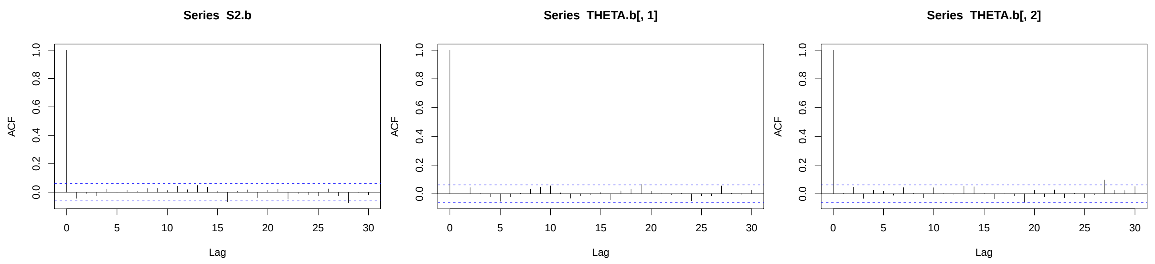
```
library(coda)
effectiveSize(S2.b)
## var1
## 1099
effectiveSize(THETA.b[,1])
##  var1
## 829.1
effectiveSize(THETA.b[,2])
##  var1
## 724.4
apply(SIGMA.PS,2,effectiveSize)
## [1] 747.2 674.0 674.0 724.0
```

```
tmp<-NULL; for(j in 1:dim(SIGMA.PS)[2]) { tmp<-c(tmp,acf(SIGMA.PS[,j])$acf[2]) }
tmp
```

```
## [1] 0.1442 0.1943 0.1943 0.1596
```

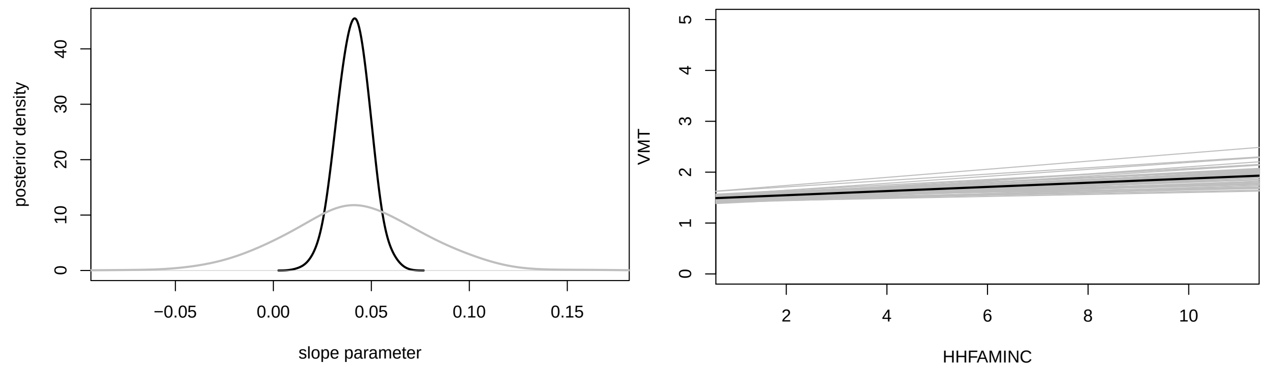


```r
acf(S2.b)
acf(THETA.b[,1])
acf(THETA.b[,2])
```



Running a Gibbs sampler for 10,000 scans and saving every 10th scan produces a sequence of 1,000 values for each parameter, each sequence having a fairly low autocorrelation. For example, the lag-10 autocorrelations of θ_1 and θ_2 are 0.027257. We can use these simulated values to make Monte Carlo approximations to various posterior quantities of interest.

```r
plot(density(THETA.b[,2],adj=2),xlim=range(BETA.pp[,2]),
      main="",xlab="slope parameter",ylab="posterior density",lwd=2)
lines(density(BETA.pp[,2],adj=2),col="gray",lwd=2)
legend( -2 ,1.0 ,legend=c( expression(theta[2]),expression(tilde(beta)[2])),
        lwd=c(2,2),col=c("black","gray"),bty="n")

quantile(THETA.b[,2],prob=c(.025,.5,.975))
##      2.5%       50%     97.5%
## -0.123708  0.239083  0.581743
mean(BETA.pp[,2]<0)
## [1] 0.346

BETA.PM<-BETA.ps/1000
plot( range(nhts2017[,4]),c(0,5),type="n",xlab="HHFAMINC", ylab="VMT") # range(nels[,3]),range(nels[,4]
for(j in 1:m) {    abline(BETA.PM[j,1],BETA.PM[j,2],col="gray")  }
abline( mean(THETA.b[,1]),mean(THETA.b[,2]),lwd=2 )
```



The first panel plots the posterior density of the expected within-school slope θ_2 of a randomly sampled income, as well as the posterior predictive distribution of a randomly sampled slope. A 95% quantile-based posterior confidence interval for θ_2 is (-0.123708, 0.581743), which, compared to our prior interval of (-2.451151, 2.477191), indicates a strong alteration in our information about θ_2.

The fact that θ_2 is unlikely to be negative only indicates that the population average of CBSA-level slopes is positive. It does not indicate that any given within-CSA slope cannot be negative. To clarify this distinction, the posterior predictive distribution of $\tilde{\beta}_2$, the slope for a to-be-sampled CBSA, is plotted in the same figure. Samples from this distribution can be generated by sampling a value $\tilde{\boldsymbol{\beta}}^{(s)}$ from a multivariate normal$(\boldsymbol{\theta}^{(s)}, \Sigma^{(s)})$ distribution for each scan s of the Gibbs sampler. Notice that this posterior predictive distribution is much more spread out than the posterior distribution of θ_2, reflecting the heterogeneity in slopes across CBSA.

Using the Monte Carlo approximation, we have $Pr(\tilde{\beta}_2 < 0|\mathbf{y_{1:m}}, \mathbf{X_{1:m}}) \approx 0.346$, which is small but not negligible.

The second panel gives posterior expectations of the 52 CBSA-specific regression lines, with the average line given by the posterior mean of θ in black. Comparing this to the first panel indicates how the hierarchical model is able to share information across groups, shrinking extreme regression lines towards the across-group average. In particular, hardly any of the slopes are negative when we share information across groups.

The GLMM Example

A basic generalized linear mixed model is as follows:

$$\boldsymbol{\beta}_{1:m} \overset{iid}{\sim} N_p(\boldsymbol{\theta}, \Sigma)$$

$$p(\boldsymbol{y_j}|\boldsymbol{\beta_j}, \boldsymbol{X_j}, \gamma) = \prod_{i=1}^{n_j} p(y_{i,j}|\boldsymbol{\beta_j^T x_{i,j}}, \gamma)$$

Generalized linear mixed effects models

Estimation for the linear mixed effects model was straightforward because the full conditional distribution of each parameter was standard, allowing for the easy implementation of a Gibbs sampling algorithm.

In contrast, for nonnormal generalized linear mixed models, typically only θ and Σ have standard full conditional distributions. This suggests we use a Metropolis-Hastings algorithm to approximate the posterior distribution of the parameters, using a combination of Gibbs steps for updating (θ, Σ) with a Metropolis step for updating each β_j.

If there is such a γ parameter, it can be updated using a Gibbs step if a full conditional distribution is available, and a Metropolis step if not.

Gibbs steps for $\boldsymbol{\theta}, \Sigma)$

Whether $p(y|\boldsymbol{\beta^T x})$ is a normal model, a Poisson model, or some other generalized linear model, the full conditional distributions of θ and Σ will be the multivariate normal and inverse-Wishart distributions.

Metropolis step for $\boldsymbol{\beta}_j$

Updating β_j in a Markov chain can proceed by proposing a new value of $\beta_j^\star$ based on the current parameter values and then accepting or rejecting it with the appropriate probability. A standard proposal distribution in this situation would be a multivariate normal distribution with mean equal to the current value $\beta_j^{(s)}$ and with some proposal variance $V_j^{(s)}$). In this case the Metropolis step is as follows:

1. Sample $\beta_j^\star \sim$ multivariate normal $(\beta_j^{(s)}, V_j^{(s)})$.
2. Compute the acceptance ratio $r = \frac{p(y_j|X_j,\beta_j^\star)p(\beta_j^\star|\boldsymbol{\theta}^{(s)},\Sigma^{(s)})}{p(y_j|X_j,\beta_j^{(s)})p(\beta_j^{(s)}|\boldsymbol{\theta}^{(s)},\Sigma^{(s)})}$.
3. Sample $u \sim$ uniform(0,1). Set $\beta_j^{(s+1)}$ to $\beta_j^\star$ if $u < r$ and to $\beta_j^{(s)}$ if $u > r$.

In many cases, setting $V_j^{(s)})$ equal to a scaled version of $\Sigma^{(s)}$ produces a well mixing Markov chain, although the task of finding the right scale might have to proceed by trial and error.

- A Metropolis-Hastings approximation algorithm

Given current values at scan s of the Markov chain, we obtain new values as follows:

1. Sample $\boldsymbol{\theta}^{(s+1)}$ from its full conditional distribution.
2. Sample $\Sigma^{(s+1)}$ from its full conditional distribution.
3. For each $j \in \{1, ..., m\}$,
    a) propose a new value $\beta_j^\star$;
    b) set $\beta_j^{(s+1)}$ equal to $\beta_j^\star$ or $\beta_j^{(s)}$ with the appropriate probability.

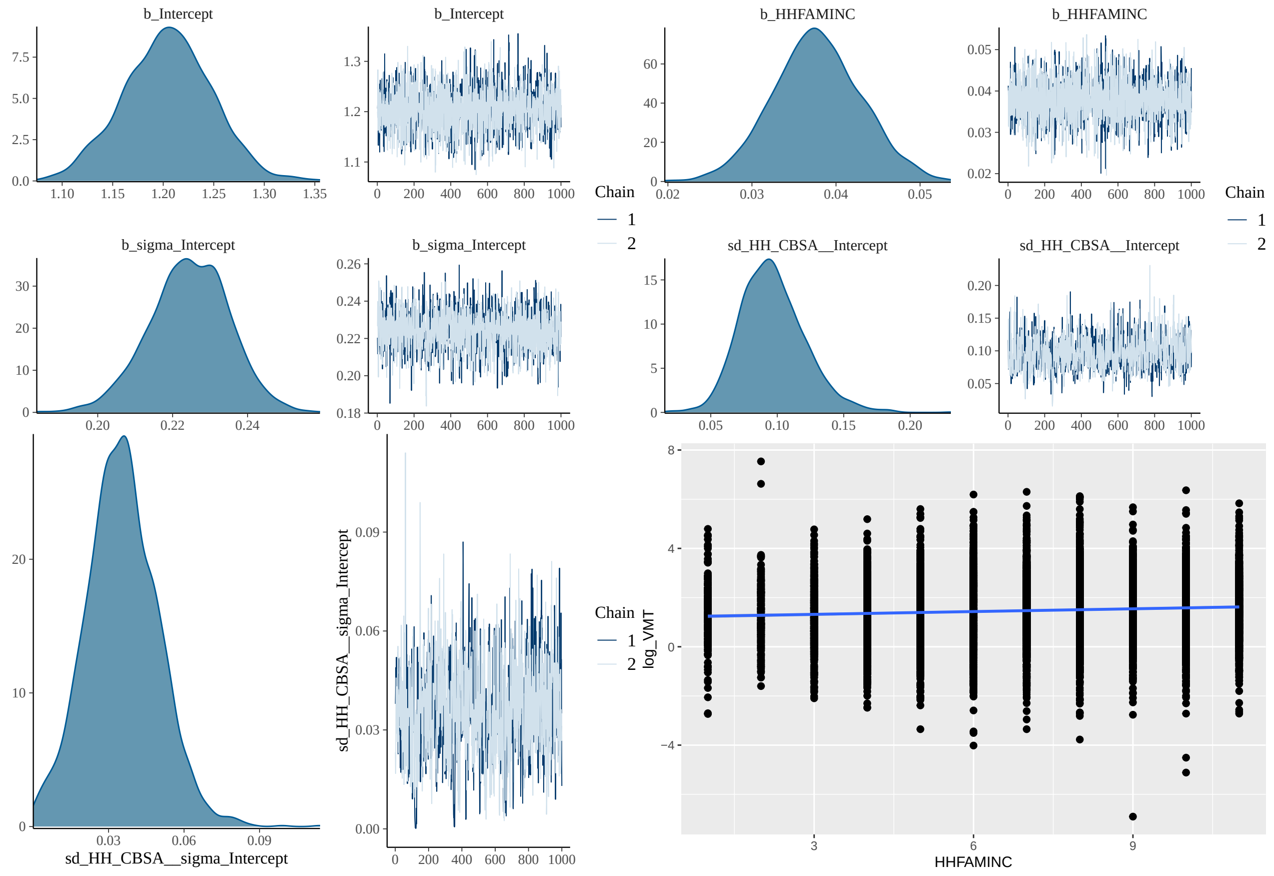
Compare with the results using 'brms'

```
fit1 <- brm(bf(log_VMT ~ HHFAMINC + (1|HH_CBSA), sigma ~ (1|HH_CBSA)),  # HHSIZE + HHFAMINC + WRKCOUNT
            chains = 2, data = nhts2017, family = gaussian())
```

```
## Running /usr/lib64/R/bin/R CMD SHLIB foo.c
## gcc -I"/usr/include/R/" -DNDEBUG   -I"/home/qs26/R/x86_64-pc-linux-gnu-library/4.0/Rcpp/include/"  -I
## In file included from /home/qs26/R/x86_64-pc-linux-gnu-library/4.0/RcppEigen/include/Eigen/Core:88,
##                  from /home/qs26/R/x86_64-pc-linux-gnu-library/4.0/RcppEigen/include/Eigen/Dense:1,
##                  from /home/qs26/R/x86_64-pc-linux-gnu-library/4.0/StanHeaders/include/stan/math/pri
##                  from <command-line>:
## /home/qs26/R/x86_64-pc-linux-gnu-library/4.0/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:1: e
##   613 | namespace Eigen {
##       | ^~~~~~~~~
## /home/qs26/R/x86_64-pc-linux-gnu-library/4.0/RcppEigen/include/Eigen/src/Core/util/Macros.h:613:17: 
##   613 | namespace Eigen {
##       |                 ^
## In file included from /home/qs26/R/x86_64-pc-linux-gnu-library/4.0/RcppEigen/include/Eigen/Dense:1,
##                  from /home/qs26/R/x86_64-pc-linux-gnu-library/4.0/StanHeaders/include/stan/math/pri
##                  from <command-line>:
## /home/qs26/R/x86_64-pc-linux-gnu-library/4.0/RcppEigen/include/Eigen/Core:96:10: fatal error: comple
##    96 | #include <complex>
##       |          ^~~~~~~~~
## compilation terminated.
## make: *** [/usr/lib64/R/etc/Makeconf:167: foo.o] Error 1
##
## SAMPLING FOR MODEL '25751eea7ce0cb33402b938aadb57c70' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.004279 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 42.79 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
```

```
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 136.846 seconds (Warm-up)
## Chain 1:                35.4429 seconds (Sampling)
## Chain 1:                172.289 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '25751eea7ce0cb33402b938aadb57c70' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0.002301 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 23.01 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 151.411 seconds (Warm-up)
## Chain 2:                35.2616 seconds (Sampling)
## Chain 2:                186.673 seconds (Total)
## Chain 2:
```

```r
summary(fit1)
##  Family: gaussian
##   Links: mu = identity; sigma = log
## Formula: log_VMT ~ HHFAMINC + (1 | HH_CBSA)
##          sigma ~ (1 | HH_CBSA)
##    Data: nhts2017 (Number of observations: 10000)
## Samples: 2 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 2000
##
## Group-Level Effects:
## ~HH_CBSA (Number of levels: 52)
##                     Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)           0.10      0.02     0.05     0.15 1.01      819      816
## sd(sigma_Intercept)     0.04      0.01     0.01     0.06 1.00      577      388
##
## Population-Level Effects:
##                   Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept             1.20      0.04     1.12     1.29 1.00     2706     1569
## sigma_Intercept       0.22      0.01     0.20     0.24 1.00     1731     1693
## HHFAMINC              0.04      0.01     0.03     0.05 1.01     5337     1588
##
## Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
plot(fit1, N = 2, ask = FALSE)
plot(conditional_effects(fit1), points = TRUE)
```



These are close to the previous results:

The expected value of θ_2 is 0.237312, the 2.5%, 50%, and 97.5% quantiles are -0.123708, 0.239083, 0.581743

The expected value of β_2 is 0.233812, the 2.5%, 50%, and 97.5% quantiles are -0.904736, 0.240037, 1.494535

The conditional_effects of group don't reveal that the variances of both groups are indeed different.

We can compute the residual standard deviations on the original scale using the hypothesis method.

```
hyp <- c("exp(sigma_Intercept) = 0",
         "exp(sigma_Intercept + HHFAMINC) = 0")
hypothesis(fit1, hyp)
```

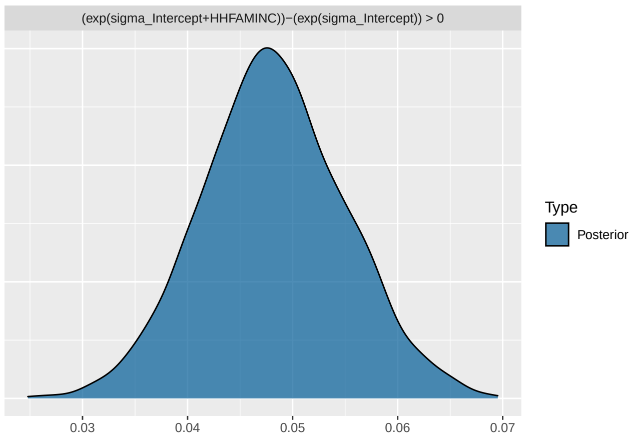
```
## Hypothesis Tests for class b:
##                Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio Post.Prob Star
## 1 (exp(sigma_Interc... = 0     1.25      0.01     1.22     1.28         NA        NA    *
## 2 (exp(sigma_Interc... = 0     1.30      0.02     1.27     1.33         NA        NA    *
## ---
## 'CI': 90%-CI for one-sided and 95%-CI for two-sided hypotheses.
## '*': For one-sided hypotheses, the posterior probability exceeds 95%;
## for two-sided hypotheses, the value tested against lies outside the 95%-CI.
## Posterior probabilities of point hypotheses assume equal prior probabilities.
```

Or directly compare them and plot the posterior distribution of their difference.

```
hyp <- "exp(sigma_Intercept + HHFAMINC) > exp(sigma_Intercept)"
(hyp <- hypothesis(fit1, hyp))
```

```
## Hypothesis Tests for class b:
##                Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio Post.Prob Star
## 1 (exp(sigma_Interc... > 0     0.05      0.01     0.04     0.06        Inf         1    *
## ---
## 'CI': 90%-CI for one-sided and 95%-CI for two-sided hypotheses.
## '*': For one-sided hypotheses, the posterior probability exceeds 95%;
## for two-sided hypotheses, the value tested against lies outside the 95%-CI.
## Posterior probabilities of point hypotheses assume equal prior probabilities.
```

```
plot(hyp, chars = NULL)
```


(exp(sigma_Intercept+HHFAMINC))−(exp(sigma_Intercept)) > 0
Type
Posterior