NELS 23 Schools and HLM

Carolyn J. Anderson

10/19/2019

These examples use the NELS data from 23 (non-random) sample from the full 1003 schools in the data set (See Kreft & de Leeuw text). lmer from the lme4 package will be used to fit model used REML and runjags will be used to fit Bayesian models.

The variables in the data set are

* SCHOOL = SCHOOL ID
* STUDENT = STUDENT ID
* SEX = STUDENT SEX
* RACE = STUDENT RACE
* HOMEW = TIME ON MATH HOMEWORK
* SCHTYPE = SCHOOL TYPE
* SES = SOCIOECONOMIC STATUS
* PARED = PARENTAL EDUCATION
* MATH = MATH SCORE
* CLASSSTR = CLASS STRUCTURE
* SCHSIZE = SCHOOL SIZE
* URBAN = URBANICITY
* GEO = GEOGRAPHIC REGION
* MINORITY = PERCENT MINORITY
* RATIO = STUDENT-TEACHER RATIO
* PUBLIC = PUBLIC SCHOOL BINARY
* WHITE = WHITE RACE BINARY

Included in this document are

* Graphs of data
* Random intercept math~ 1 + homew + (1 |school.id)
* Random intercept & slope (un-correlated): math~ 1 + homew + (1 |school.id) + (0 + homew | school.id)
* Random intercept & slope (correlated): math~ 1 + homew + (1 + homew |school.id)
* Random intercept & slope (correlated) with more predictors: math~ 1 + homew + ses + public + homew\*public + (1 + homew|school.id)
* Statistics and graphics using posertior samples (approximate posterior distributions)

# Set up

These are libraries that we’ll use (at least most of them)

library(lme4)  
library(lmerTest)  
library(coda)  
library(rjags)  
library(runjags)  
library(mvtnorm)

Set working directory and read in the data

setwd("C:\\Users\\cja\\Dropbox\\edps 590BAY\\Lectures\\10 Multilevel models")  
  
nels <- read.table("school23\_data.txt",header=TRUE)  
head(nels)

## school student sex race homew schtype ses pared math classtr schsize  
## 1 7472 3 2 4 1 1 -0.13 2 48 2 3  
## 2 7472 8 1 4 0 1 -0.39 2 48 2 3  
## 3 7472 13 1 4 0 1 -0.80 2 53 2 3  
## 4 7472 17 1 4 1 1 -0.72 2 42 2 3  
## 5 7472 27 2 4 2 1 -0.74 2 43 2 3  
## 6 7472 28 2 4 1 1 -0.58 2 57 2 3  
## urban geo minority ratio  
## 1 2 2 0 19  
## 2 2 2 0 19  
## 3 2 2 0 19  
## 4 2 2 0 19  
## 5 2 2 0 19  
## 6 2 2 0 19

tail(nels)

## school student sex race homew schtype ses pared math classtr schsize  
## 514 72991 62 2 4 1 1 -1.01 3 55 4 7  
## 515 72991 63 2 1 5 1 0.63 5 63 4 7  
## 516 72991 70 2 4 4 1 0.56 4 67 4 7  
## 517 72991 77 2 4 1 1 0.46 3 53 4 7  
## 518 72991 87 1 4 1 1 0.06 3 56 4 7  
## 519 72991 96 2 4 2 1 -0.05 2 53 4 7  
## urban geo minority ratio  
## 514 2 1 2 13  
## 515 2 1 2 13  
## 516 2 1 2 13  
## 517 2 1 2 13  
## 518 2 1 2 13  
## 519 2 1 2 13

We’ll need some various statistics to create nice plot of the data.

# sort by schools:  
nels <- nels[order(nels$school,nels$student) , ]  
  
# Get information on number schools & create consecutive integer school.id  
school <- unique(nels$school) ##   
N <- length(school) ## need N and school.id  
school.int <- as.data.frame(cbind(school,seq(1:N))) ## for dataList & model   
names(school.int) <- c("school", "school.id") ##  
nels <- merge(nels,school.int,by="school") ##  
  
# total number of students  
n <- length(nels$math) ## need n for dataList & model

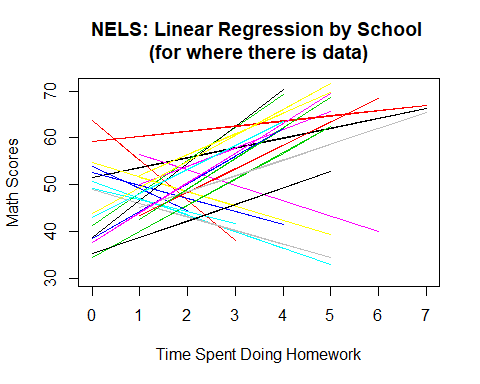
# Double-check set-up  
head(nels,n=50)

# You don't need this but might want it  
nj <- matrix(999,nrow=N,ncol=1)  
for (j in 1:N) {  
 nj[j] <- nrow(subset(nels,nels$school.id==j))  
}

We now have everything need to create a graph showing the variability between schools.

I would suggest putting par(mfrow=c(2,2)) as this point because if make comparisons a bit easier; however, remove is before running Bayesian.

# Graph to illustrate need for random intercept and slope  
plot(nels$homew,nels$math,  
 type="n",   
 col="blue",  
 lwd=2,  
 main="NELS: Linear Regression by School \n(for where there is data)",  
 xlab="Time Spent Doing Homework",  
 ylab="Math Scores"  
 )  
for (j in 1:N) {  
 sub <- subset(nels,nels$school.id==j)  
 fitted <- fitted(lm(math~homew,sub))  
 lines(sub$homew,fitted,col=j)  
}

 It looks like different intercepts and difference slopes for time spent doing homework.

Now for our first model:

###################################################################  
# Random intercept math~ 1 + homew + (1 |school.id) #  
###################################################################  
ri1.lmer <- lmer(math~ 1 + homew + (1 |school.id), data=nels, REML=TRUE)

We’ll look at ri1.lmer latter after we fit the model using Bayesian methods

N = 23  
n = length(nels$math)  
  
dataList <- list(  
 y = nels$math,  
 hmwk = nels$homew,  
 school.id = nels$school.id,  
 N = N,  
 n = n,  
 sdY = sd(nels$math)  
 )  
   
ri.mod1 <- "model {   
 for (i in 1:n) {  
 y[i] ~ dnorm(mu[i],precision)   
 mu[i] <- g0 + U0j[school.id[i]] + g1\*hmwk[i]   
 }   
 for (j in 1:N) {  
 U0j[j] ~ dnorm(0,ptau)  
 }   
 g0 ~ dnorm(0,1/(100\*sdY^2))  
 g1 ~ dnorm(0,1/(100\*sdY^2))  
  
 tau ~ dunif(0.0001,200)  
 ptau <- 1/tau^2  
   
 sigma ~ dunif(0.0001,2000)  
 precision <- 1/sigma^2   
 }"  
  
writeLines(ri.mod1, con="ri.mod1.txt")  
  
start1 = list("g0"=mean(nels$math), "g1"=rnorm(1,1,3), "sigma"=sd(nels$math), "tau"=.5,   
 .RNG.name="base::Wichmann-Hill", .RNG.seed=523)   
   
start2 = list("g0"=rnorm(1,0,3), "g1"=rnorm(1,-2,3), "sigma"=runif(1,.001,10), "tau"=runif(1,0.0001,10),   
 .RNG.name="base::Marsaglia-Multicarry", .RNG.seed=57)  
   
start3 = list("g0"=rnorm(1,3,4), "g1"=rnorm(1,0,3), "sigma"=runif(1,.001,10), "tau"=runif(1,0.0001,10),   
 .RNG.name="base::Super-Duper", .RNG.seed=24)  
   
start4 = list("g0"=rnorm(1,-3,10), "g1"=rnorm(1,5,3), "sigma"=runif(1,.001,10), "tau"=runif(1,0.0001,10),   
 .RNG.name="base::Mersenne-Twister", .RNG.seed=72100)  
  
start <- list(start1,start2,start3,start4)

When writing the model, it is easy to make mistakes. Rather than running code that could take a long time, a quick check of your code can save time. The following is a fast check

#############################################################################  
# Fast way to check your model: rjags and then run for sampling runjags with parallel.  
 ri.mod1x <- jags.model(file="ri.mod1.txt", # compiles and intializes model  
 data=dataList,  
 inits=start1,  
 n.chains=4,  
 n.adapt=500)   
#############################################################################

Since I have already run this model and know the code is OK, I did not run the above. So, let’s use runjags (I’m lazy and runjags will compute statistics I want automatically).

ri.mod1.runjags <- run.jags(model=ri.mod1,   
 method="parallel",   
 monitor=c("g0","g1","sigma", "tau"),  
 data=dataList,  
 sample=10000,   
 n.chains=4,  
 inits=start)

## Calling 4 simulations using the parallel method...  
## Following the progress of chain 1 (the program will wait for all  
## chains to finish before continuing):  
## Welcome to JAGS 4.3.0 on Tue Oct 22 16:20:15 2019  
## 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: 519  
## Unobserved stochastic nodes: 27  
## Total graph size: 1726  
## . Reading parameter file inits1.txt  
## . Initializing model  
## . Adapting 1000  
## -------------------------------------------------| 1000  
## ++++++++++++++++++++++++++++++++++++++++++++++++++ 100%  
## Adaptation successful  
## . Updating 4000  
## -------------------------------------------------| 4000  
## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 100%  
## . . . . . Updating 10000  
## -------------------------------------------------| 10000  
## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 100%  
## . . . . Updating 0  
## . Deleting model  
## .   
## All chains have finished  
## Simulation complete. Reading coda files...  
## Coda files loaded successfully  
## Calculating summary statistics...  
## Calculating the Gelman-Rubin statistic for 4 variables....  
## Finished running the simulation

add.summary(ri.mod1.runjags)

## Calculating summary statistics...  
## Calculating the Gelman-Rubin statistic for 4 variables....

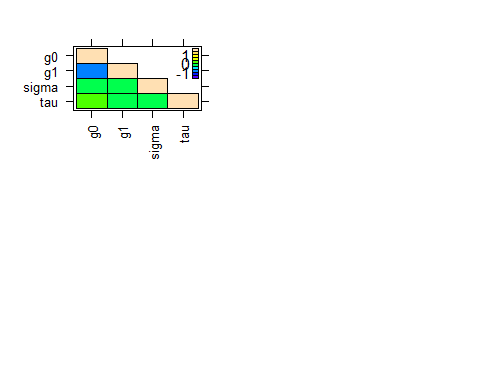
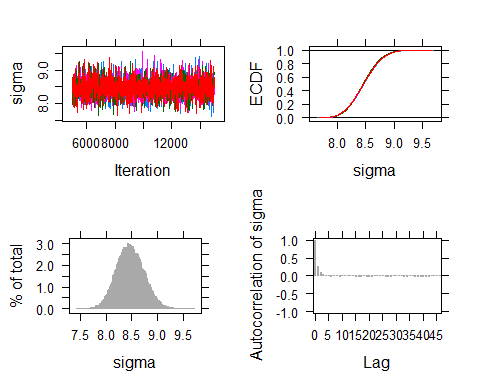
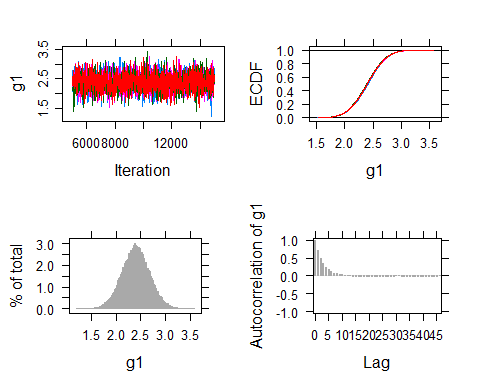
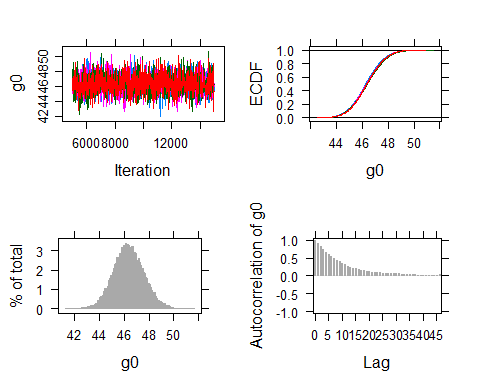
##   
## JAGS model summary statistics from 40000 samples (chains = 4; adapt+burnin = 5000):  
##   
## Lower95 Median Upper95 Mean SD Mode MCerr MC%ofSD SSeff  
## g0 43.97 46.317 48.8 46.348 1.2333 -- 0.027387 2.2 2028  
## g1 1.8392 2.3949 2.941 2.3941 0.2799 -- 0.0033827 1.2 6846  
## sigma 7.955 8.46 9.0085 8.4672 0.27081 -- 0.0018184 0.7 22178  
## tau 3.278 4.7994 6.7853 4.9157 0.92981 -- 0.0088333 1 11080  
##   
## AC.10 psrf  
## g0 0.35062 1.001  
## g1 0.028642 1.0002  
## sigma 0.0067123 1.0002  
## tau 0.02466 1.0003  
##   
## Total time taken: 11.9 seconds

summary(ri1.lmer)

## Linear mixed model fit by REML. t-tests use Satterthwaite's method [  
## lmerModLmerTest]  
## Formula: math ~ 1 + homew + (1 | school.id)  
## Data: nels  
##   
## REML criterion at convergence: 3729.3  
##   
## Scaled residuals:   
## Min 1Q Median 3Q Max   
## -2.5940 -0.7066 0.0052 0.6613 3.2075   
##   
## Random effects:  
## Groups Name Variance Std.Dev.  
## school.id (Intercept) 21.34 4.620   
## Residual 71.28 8.443   
## Number of obs: 519, groups: school.id, 23  
##   
## Fixed effects:  
## Estimate Std. Error df t value Pr(>|t|)   
## (Intercept) 46.3558 1.1628 33.0123 39.866 <2e-16 \*\*\*  
## homew 2.3999 0.2772 512.8988 8.658 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Correlation of Fixed Effects:  
## (Intr)  
## homew -0.437

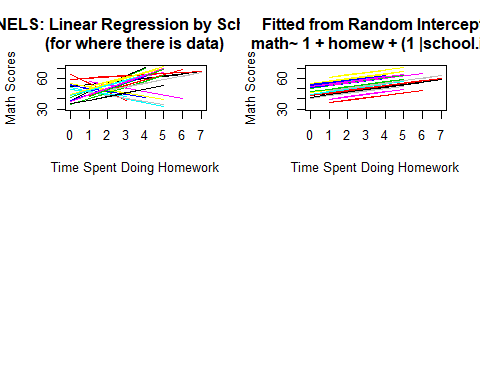
plot(ri.mod1.runjags)

## Generating plots...



Some graphics

#-------------  
# Graph to illustrate need for random intercept and slope  
par(mfrow=c(2,2))  
plot(nels$homew,nels$math, type="n", lwd=2,  
 main="NELS: Linear Regression by School \n(for where there is data)",  
 xlab="Time Spent Doing Homework", ylab="Math Scores" )  
for (j in 1:N) {  
 sub <- subset(nels,nels$school.id==j)  
 fitted <- fitted(lm(math~homew,sub))  
 lines(sub$homew,fitted,col=j)  
}   
  
# Compute conditional means (regressions)  
S<-1  
math.post <- matrix(999,nrow=S,ncol=n)  
homework <- matrix(nels$homew,nrow=n,ncol=1)  
for (s in 1:S) {  
 pointer <- 1   
 for (j in 1:N) {  
 U0 <-rnorm(1, 46.32, sd=4.9156)  
 for (i in 1:nj[j]) {  
 math.post[s,pointer] = U0 + 2.4006\*homework[pointer]   
 pointer <- pointer + 1  
 }   
 }  
}   
nels$math.ri1 <- matrix(math.post,nrow=n,ncol=1)  
plot(nels$homew,nels$math, type="n", lwd=2,  
 main="Fitted from Random Intercept \nmath~ 1 + homew + (1 |school.id)",  
 xlab="Time Spent Doing Homework", ylab="Math Scores" )  
for (j in 1:N) {  
 sub <- subset(nels,nels$school.id==j)  
 lines(sub$homew,sub$math.ri1,col=j)  
}



We knew that we needed more than just a random intercept and the graph also shows this…data (left) versus conditional fitted values (right).

So we add a random slope, but we’ll start with uncorrelated random effects.

###################################################################  
# Random intercept & slope math~ 1 + homew + (1 |school.id) #  
# Un-correlated random effects #  
###################################################################  
ris1.lmer <- lmer(math~ 1 + homew + (1 |school.id) + (0 + homew | school.id), data=nels, REML=TRUE)  
summary(ris1.lmer)

## Linear mixed model fit by REML. t-tests use Satterthwaite's method [  
## lmerModLmerTest]  
## Formula: math ~ 1 + homew + (1 | school.id) + (0 + homew | school.id)  
## Data: nels  
##   
## REML criterion at convergence: 3657.2  
##   
## Scaled residuals:   
## Min 1Q Median 3Q Max   
## -2.26033 -0.66620 0.01046 0.65144 2.73355   
##   
## Random effects:  
## Groups Name Variance Std.Dev.  
## school.id (Intercept) 50.74 7.123   
## school.id.1 homew 13.76 3.709   
## Residual 53.94 7.345   
## Number of obs: 519, groups: school.id, 23  
##   
## Fixed effects:  
## Estimate Std. Error df t value Pr(>|t|)   
## (Intercept) 46.4644 1.6090 22.0320 28.879 <2e-16 \*\*\*  
## homew 1.9745 0.8315 19.4008 2.375 0.028 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Correlation of Fixed Effects:  
## (Intr)  
## homew -0.110

dataList <- list(  
 math = nels$math,  
 hmwk = nels$homew,  
 school.id = nels$school.id,  
 N = N,  
 n = n,  
 sdY = sd(nels$math)  
 )  
   
ris.mod1 <- "model {   
 for (i in 1:n) {  
 math[i] ~ dnorm(mu[i],precision)  
 mu[i] <- g0 + g1\*hmwk[i] + b0j[school.id[i]] + b1j[school.id[i]]\*hmwk[i]  
 }  
   
 for (j in 1:N) {  
 b0j[j] ~ dnorm(0,ptau0)  
 b1j[j] ~ dnorm(0,ptau1)  
 }  
   
 g0 ~ dnorm(0,1/(100\*sdY^2))  
 g1 ~ dnorm(0,1/(100\*sdY^2))  
  
 tau0 ~ dunif(0.0001,200)  
 ptau0 <- 1/tau0^2  
   
 tau1 ~ dunif(0.0001,200)  
 ptau1 <- 1/tau1^2  
   
 sigma ~ dunif(0.0001,2000)  
 precision <- 1/sigma^2   
 }"  
  
writeLines(ris.mod1, con="ris.mod1.txt")  
  
start1 = list("g0"=mean(nels$math), "g1"=rnorm(1,1,3),   
 "sigma"=sd(nels$math), "tau0"=.5, "tau1"=.5,   
 .RNG.name="base::Wichmann-Hill", .RNG.seed=523)   
   
start2 = list("g0"=rnorm(1,0,3), "g1"=rnorm(1,-2,3),   
 "sigma"=runif(1,.001,10), "tau0"=runif(1,0.0001,10), "tau1"=runif(1,0.0001,10),   
 .RNG.name="base::Marsaglia-Multicarry", .RNG.seed=57)  
   
start3 = list("g0"=rnorm(1,3,4), "g1"=rnorm(1,0,3),  
 "sigma"=runif(1,.001,10), "tau0"=runif(1,0.0001,10), "tau1"=runif(1,0.0001,10),   
 .RNG.name="base::Super-Duper", .RNG.seed=24)  
   
start4 = list("g0"=rnorm(1,-3,10), "g1"=rnorm(1,5,3),   
 "sigma"=runif(1,.001,10), "tau0"=runif(1,0.0001,10), "tau1"=runif(1,0.0001,10),   
 .RNG.name="base::Mersenne-Twister", .RNG.seed=72100)  
  
start <- list(start1,start2,start3,start4)

#############################################################################  
# Fast way to check your model: rjags and then run for sampling runjags with parallel.  
 ris.mod1 <- jags.model(file="ris.mod1.txt", # compiles and initializes model  
 data=dataList,  
 inits=start1,  
 n.chains=4,  
 n.adapt=500)   
#############################################################################

We’ll go ahead and run this,

ris.mod1 <- run.jags(model=ris.mod1,   
 method="parallel",   
 monitor=c("g0","g1","sigma", "tau0","tau1"),  
 data=dataList,  
 sample=10000,   
 n.chains=4,  
 inits=start,  
 thin=10)

## Calling 4 simulations using the parallel method...  
## Following the progress of chain 1 (the program will wait for all  
## chains to finish before continuing):  
## Welcome to JAGS 4.3.0 on Tue Oct 22 16:20:36 2019  
## 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: 519  
## Unobserved stochastic nodes: 51  
## Total graph size: 1869  
## . Reading parameter file inits1.txt  
## . Initializing model  
## . Adapting 1000  
## -------------------------------------------------| 1000  
## ++++++++++++++++++++++++++++++++++++++++++++++++++ 100%  
## Adaptation successful  
## . Updating 4000  
## -------------------------------------------------| 4000  
## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 100%  
## . . . . . . Updating 100000  
## -------------------------------------------------| 100000  
## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 100%  
## . . . . Updating 0  
## . Deleting model  
## .   
## All chains have finished  
## Simulation complete. Reading coda files...  
## Coda files loaded successfully  
## Calculating summary statistics...  
## Calculating the Gelman-Rubin statistic for 5 variables....  
## Finished running the simulation

summary(ris.mod1)

## Lower95 Median Upper95 Mean SD Mode MCerr  
## g0 42.990700 46.455850 49.86730 46.457762 1.7287349 NA 0.020010475  
## g1 0.183483 1.968780 3.73254 1.962372 0.8939192 NA 0.013668182  
## sigma 6.893600 7.358380 7.84224 7.363531 0.2432793 NA 0.001231074  
## tau0 5.059670 7.446765 10.45680 7.614498 1.4290001 NA 0.008244494  
## tau1 2.588030 3.880060 5.52321 3.967061 0.7729221 NA 0.004622817  
## MC%ofSD SSeff AC.100 psrf  
## g0 1.2 7463 -0.017328258 1.000344  
## g1 1.5 4277 0.087009374 1.000695  
## sigma 0.5 39052 0.004759155 1.000022  
## tau0 0.6 30043 0.001680297 1.000040  
## tau1 0.6 27955 -0.003848790 1.000072

If for some reason, summary(ris.mod1) doesn’t produce results, try add.summary(ris.mod1).

And for comparison

summary(ris1.lmer)

## Linear mixed model fit by REML. t-tests use Satterthwaite's method [  
## lmerModLmerTest]  
## Formula: math ~ 1 + homew + (1 | school.id) + (0 + homew | school.id)  
## Data: nels  
##   
## REML criterion at convergence: 3657.2  
##   
## Scaled residuals:   
## Min 1Q Median 3Q Max   
## -2.26033 -0.66620 0.01046 0.65144 2.73355   
##   
## Random effects:  
## Groups Name Variance Std.Dev.  
## school.id (Intercept) 50.74 7.123   
## school.id.1 homew 13.76 3.709   
## Residual 53.94 7.345   
## Number of obs: 519, groups: school.id, 23  
##   
## Fixed effects:  
## Estimate Std. Error df t value Pr(>|t|)   
## (Intercept) 46.4644 1.6090 22.0320 28.879 <2e-16 \*\*\*  
## homew 1.9745 0.8315 19.4008 2.375 0.028 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Correlation of Fixed Effects:  
## (Intr)  
## homew -0.110

And some graphs from Bayesian analysis

plot(ris.mod1.runjags)

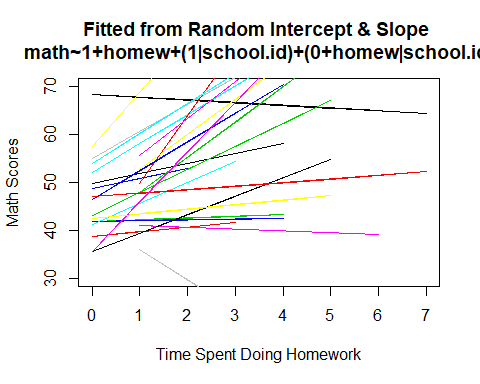
# To obtain estmates of the random effects

Note coefficients used should match what you just got…

#### Better to estimate U within jags, but to get idea across ######  
# Compute conditional regressions  
S<-1  
math.post <- matrix(999,nrow=S,ncol=n)  
homework <- matrix(nels$homew,nrow=n,ncol=1)  
for (s in 1:S) {  
 pointer <- 1   
 for (j in 1:N) {  
 U0 <-rnorm(1, 46.443, sd=7.5859)  
 U1 <-rnorm(1, 1.9677, sd=3.9674)  
 for (i in 1:nj[j]) { # for estimates include + rnorm(1,g0,sigma)  
 math.post[s,pointer] = U0 + U1\*homework[pointer]   
 pointer <- pointer + 1  
 }   
 }  
}

Now we can plot stuff

nels$math.ris12 <- matrix(math.post,nrow=n,ncol=1)  
plot(nels$homew,nels$math, type="n", lwd=2,  
 main="Fitted from Random Intercept & Slope \nmath~1+homew+(1|school.id)+(0+homew|school.id)",  
 xlab="Time Spent Doing Homework", ylab="Math Scores",  
 ylim=c(30,70) )  
for (j in 1:N) {  
 sub <- subset(nels,nels$school.id==j)  
 lines(sub$homew,sub$math.ris1,col=j)  
}



This look a lot like the data, but we can do better by allowing random intercept and random slope to be correlated.

#############################################################  
# Random intercept & slope Wishart #  
# math~ 1 + homew + (1 + homew|school.id) #  
# with correlated random effects: use wishart #  
#############################################################  
ris2.lmer <- lmer(math~ 1 + homew + (1 + homew|school.id), data=nels, REML=TRUE)

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =  
## control$checkConv, : Model failed to converge with max|grad| = 0.00208918  
## (tol = 0.002, component 1)

summary(ris2.lmer)

## Linear mixed model fit by REML. t-tests use Satterthwaite's method [  
## lmerModLmerTest]  
## Formula: math ~ 1 + homew + (1 + homew | school.id)  
## Data: nels  
##   
## REML criterion at convergence: 3635.6  
##   
## Scaled residuals:   
## Min 1Q Median 3Q Max   
## -2.22684 -0.70441 0.00351 0.65896 2.75156   
##   
## Random effects:  
## Groups Name Variance Std.Dev. Corr   
## school.id (Intercept) 62.42 7.900   
## homew 17.73 4.211 -0.83  
## Residual 53.29 7.300   
## Number of obs: 519, groups: school.id, 23  
##   
## Fixed effects:  
## Estimate Std. Error df t value Pr(>|t|)   
## (Intercept) 46.3257 1.7588 22.0013 26.339 <2e-16 \*\*\*  
## homew 1.9801 0.9285 19.9143 2.132 0.0456 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Correlation of Fixed Effects:  
## (Intr)  
## homew -0.824  
## convergence code: 0  
## Model failed to converge with max|grad| = 0.00208918 (tol = 0.002, component 1)

dataList <- list(  
 y = nels$math,  
 hmwk = nels$homew,  
 school.id=nels$school.id,  
 n = n,  
 N = N,  
 sdY = sd(nels$math)  
 )  
   
re.mod2 <- "model {   
 # Likelihood: the data model  
 for (i in 1:n) {  
 y[i] ~ dnorm(meanY[i],precision)  
 meanY[i] <- betaj[school.id[i],1] + betaj[school.id[i],2]\*hmwk[i]  
 }  
 # Random Effects -- note multivariate normal density is used  
 for (j in 1:N) {  
 betaj[j,1:2] ~ dmnorm(mu[1:2],Omega[1:2,1:2])  
 }  
   
 # Priors  
 precision ~ dgamma(0.01,0.01)  
 sigma <- 1/sqrt(precision)   
   
 mu[1] ~ dnorm(0,1/(100\*sdY^2))# or mu[i]<- 0 and add g0 & g1 to meanY   
 mu[2] ~ dnorm(0,1/(100\*sdY^2))# and add g0<- dnorm(0,1,(100\*sdY^2))  
 # & for g1. This will make it easier  
 # to add models for intercept & slopes.  
 # See next model.  
 Omega[1:2,1:2] ~ dwish(R[,],2.1)  
 R[1,1] <- 1/2.1  
 R[1,2] <- 0 # Un-correlated--there are alternative  
 R[2,1] <- 0 # ways of setting this up by just adding  
 R[2,2] <- 1/2.1 # uni-variate distribution for each tau  
 Tau <- inverse(Omega) # This turn into our taus.  
 }"  
  
writeLines(re.mod2, con="re.mod2.txt")  
  
start1 = list("precision"=1, .RNG.name="base::Wichmann-Hill", .RNG.seed=523)   
   
start2 = list("precision"=1, .RNG.name="base::Marsaglia-Multicarry", .RNG.seed=57)  
   
start3 = list("precision"=1, .RNG.name="base::Super-Duper", .RNG.seed=24)  
   
start4 = list("precision"=1, .RNG.name="base::Mersenne-Twister", .RNG.seed=72100)  
  
start <- list(start1,start2,start3,start4)  
  
##########################################################################  
# Fast way to check your model:   
# rjags and then run for sampling runjags with parallel.  
##########################################################################  
re.mod2x <- jags.model(file="re.mod2.txt", # compiles and initializes model  
 data=dataList,  
 inits=start1,  
 n.chains=4,  
 n.adapt=500)

## Compiling model graph  
## Resolving undeclared variables  
## Allocating nodes  
## Graph information:  
## Observed stochastic nodes: 519  
## Unobserved stochastic nodes: 27  
## Total graph size: 1882  
##   
## Initializing model

#########################################################################  
  
re.mod2.runjags <- run.jags(model=re.mod2,   
 method="parallel",   
 monitor=c("mu","sigma","Tau"),  
 data=dataList,  
 sample=20000,   
 n.chains=4,  
 inits=start)

## Calling 4 simulations using the parallel method...  
## Following the progress of chain 1 (the program will wait for all  
## chains to finish before continuing):  
## Welcome to JAGS 4.3.0 on Tue Oct 22 16:21:30 2019  
## 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: 519  
## Unobserved stochastic nodes: 27  
## Total graph size: 1882  
## . Reading parameter file inits1.txt  
## . Initializing model  
## . Adaptation skipped: model is not in adaptive mode.  
## . Updating 4000  
## -------------------------------------------------| 4000  
## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 100%  
## . . . . Updating 20000  
## -------------------------------------------------| 20000  
## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 100%  
## . . . . Updating 0  
## . Deleting model  
## .   
## All chains have finished  
## Note: the model did not require adaptation  
## Simulation complete. Reading coda files...  
## Coda files loaded successfully  
## Calculating summary statistics...  
## Calculating the Gelman-Rubin statistic for 7 variables....  
## Note: Unable to calculate the multivariate psrf  
## Finished running the simulation

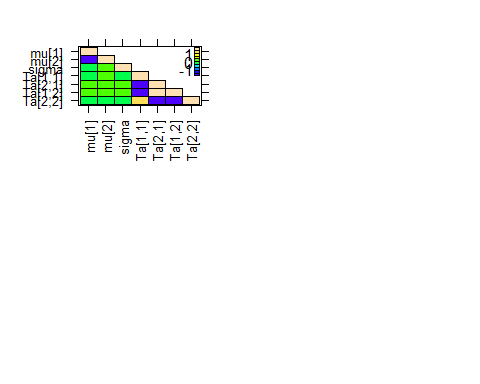
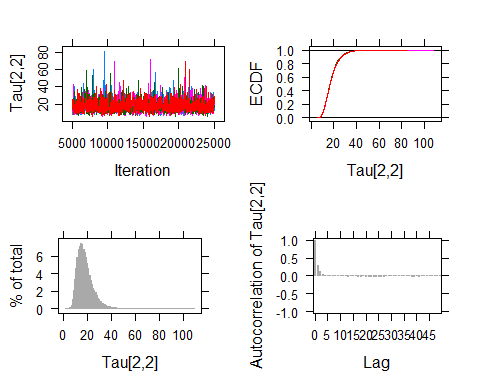
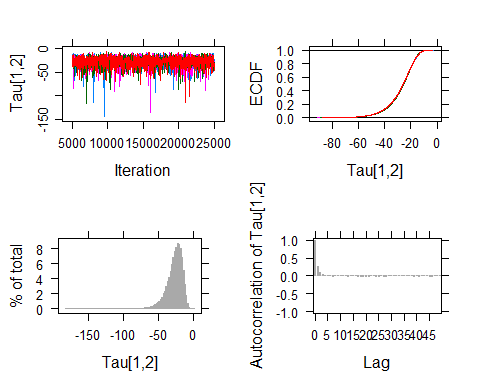
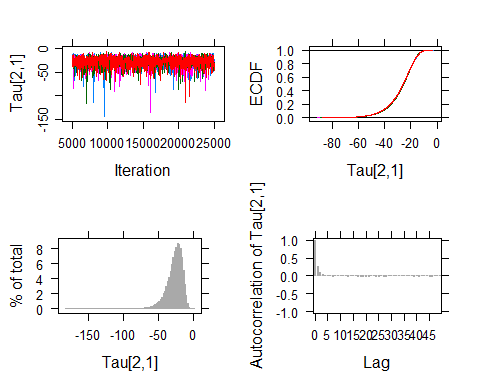
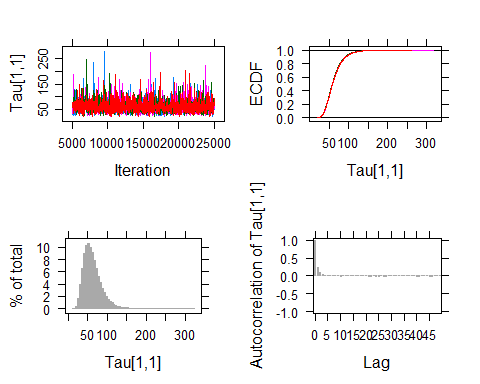
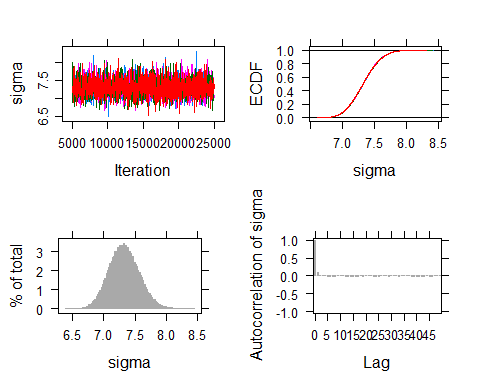
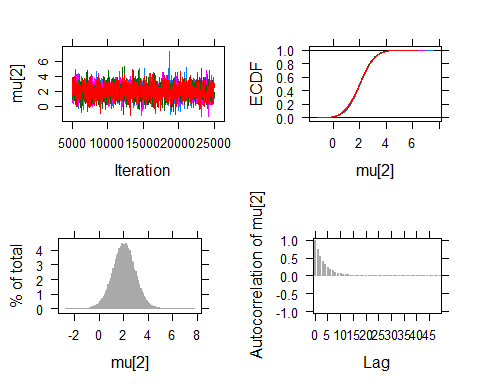
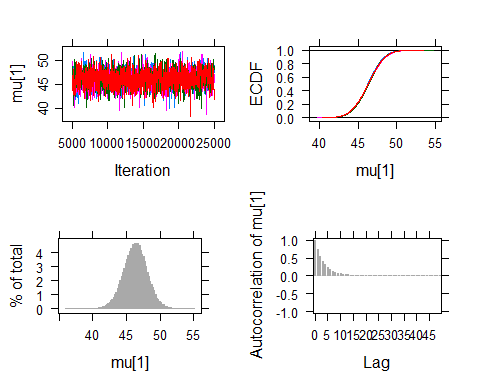
add.summary(re.mod2.runjags)

## Calculating summary statistics...  
## Calculating the Gelman-Rubin statistic for 7 variables....  
## Note: Unable to calculate the multivariate psrf

##   
## JAGS model summary statistics from 80000 samples (chains = 4; adapt+burnin = 5000):  
##   
## Lower95 Median Upper95 Mean SD Mode MCerr MC%ofSD  
## mu[1] 42.763 46.315 49.771 46.297 1.7731 -- 0.016651 0.9  
## mu[2] 0.17778 2.0086 3.8683 2.0082 0.93417 -- 0.0087361 0.9  
## sigma 6.869 7.3201 7.8037 7.3259 0.23938 -- 0.00093184 0.4  
## Tau[1,1] 25.819 58.765 108.79 62.992 23.233 -- 0.11 0.5  
## Tau[2,1] -50.177 -25.674 -9.5123 -27.737 11.38 -- 0.055013 0.5  
## Tau[1,2] -50.177 -25.674 -9.5123 -27.737 11.38 -- 0.055013 0.5  
## Tau[2,2] 7.1918 16.574 31.316 17.825 6.7644 -- 0.033606 0.5  
##   
## SSeff AC.10 psrf  
## mu[1] 11339 0.062008 1.0004  
## mu[2] 11434 0.058839 1.0003  
## sigma 65994 0.0022849 0.99999  
## Tau[1,1] 44609 -0.0037175 1.0001  
## Tau[2,1] 42791 0.002518 1.0001  
## Tau[1,2] 42791 0.002518 1.0001  
## Tau[2,2] 40516 0.0056721 1.0001  
##   
## Total time taken: 10.3 seconds

plot(re.mod2.runjags)

## Generating plots...



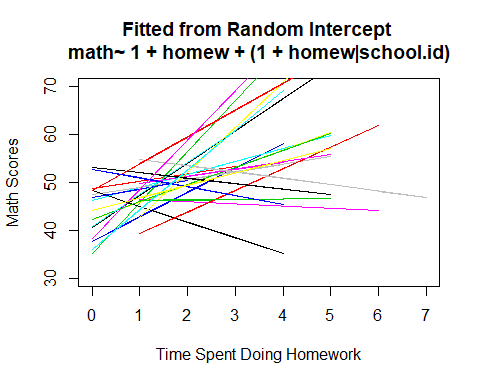
summary(ris2.lmer)

## Linear mixed model fit by REML. t-tests use Satterthwaite's method [  
## lmerModLmerTest]  
## Formula: math ~ 1 + homew + (1 + homew | school.id)  
## Data: nels  
##   
## REML criterion at convergence: 3635.6  
##   
## Scaled residuals:   
## Min 1Q Median 3Q Max   
## -2.22684 -0.70441 0.00351 0.65896 2.75156   
##   
## Random effects:  
## Groups Name Variance Std.Dev. Corr   
## school.id (Intercept) 62.42 7.900   
## homew 17.73 4.211 -0.83  
## Residual 53.29 7.300   
## Number of obs: 519, groups: school.id, 23  
##   
## Fixed effects:  
## Estimate Std. Error df t value Pr(>|t|)   
## (Intercept) 46.3257 1.7588 22.0013 26.339 <2e-16 \*\*\*  
## homew 1.9801 0.9285 19.9143 2.132 0.0456 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Correlation of Fixed Effects:  
## (Intr)  
## homew -0.824  
## convergence code: 0  
## Model failed to converge with max|grad| = 0.00208918 (tol = 0.002, component 1)

The frequenist estimated more ‘failed to converge’. It doesn’t look that bad and we might be able to get it to converge if we change the optimization method.

Let’s look at what fitted values look like conditional on random effects

########### Better to estimate U within jags, but to get idea across ############  
# Compute conditional regressions  
S<- 1  
math.post <- matrix(999,nrow=S,ncol=n)  
homework <- matrix(nels$homew,nrow=n,ncol=1)  
Tau <- matrix(c(63.01,-27.717,-27.717,17.804),nrow=2,ncol=2)  
for (s in 1:S) {  
 pointer <- 1   
 for (j in 1:N) {  
 Uj <- rmvnorm(1,mean=c(46.296,2.0062),sigma=Tau)   
 for (i in 1:nj[j]) {  
 math.post[s,pointer] = Uj[1,1] + Uj[1,2]\*homework[pointer] # for estimates include + rnorm(1,g0,sigma)  
 pointer <- pointer + 1  
 }   
 }  
}   
  
nels$math.ris2 <- matrix(math.post,nrow=n,ncol=1)  
  
plot(nels$homew,nels$math,  
 type="n",   
 col="blue",  
 lwd=2,  
 main="Fitted from Random Intercept \nmath~ 1 + homew + (1 + homew|school.id)",  
 xlab="Time Spent Doing Homework",  
 ylab="Math Scores",  
 ylim=c(30,70)  
 )  
for (j in 1:N) {  
 sub <- subset(nels,nels$school.id==j)  
 lines(sub$homew,sub$math.ris2,col=j)  
}



Now for model 3:

#############################################################################  
# Random intercept & slope with more predictors including a cross-level #  
# interaction #  
# math~ 1 + homew + ses + public + homew\*public +(1 + homew|school.id) #  
# #  
#############################################################################  
# Turn into dummy codes  
nels$public <- ifelse(nels$schtype==1,1,0)  
  
ris3.lmer <- lmer(math~ 1 + homew + ses + public + homew\*public   
 + (1 + homew|school.id), data=nels, REML=TRUE)  
summary(ris3.lmer)

## Linear mixed model fit by REML. t-tests use Satterthwaite's method [  
## lmerModLmerTest]  
## Formula: math ~ 1 + homew + ses + public + homew \* public + (1 + homew |   
## school.id)  
## Data: nels  
##   
## REML criterion at convergence: 3598.3  
##   
## Scaled residuals:   
## Min 1Q Median 3Q Max   
## -2.39584 -0.67895 -0.02471 0.63891 2.99297   
##   
## Random effects:  
## Groups Name Variance Std.Dev. Corr   
## school.id (Intercept) 53.68 7.326   
## homew 15.50 3.936 -0.87  
## Residual 51.40 7.169   
## Number of obs: 519, groups: school.id, 23  
##   
## Fixed effects:  
## Estimate Std. Error df t value Pr(>|t|)   
## (Intercept) 47.5427 2.8217 21.5090 16.849 7.13e-14 \*\*\*  
## homew 2.3253 1.4742 18.6130 1.577 0.132   
## ses 2.6621 0.5071 496.4468 5.250 2.26e-07 \*\*\*  
## public -0.8166 3.4981 21.7687 -0.233 0.818   
## homew:public -0.7525 1.8289 18.8434 -0.411 0.685   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Correlation of Fixed Effects:  
## (Intr) homew ses public  
## homew -0.856   
## ses -0.068 0.000   
## public -0.812 0.691 0.135   
## homew:publc 0.692 -0.806 -0.026 -0.856

dataList <- list(  
 math = nels$math,  
 hmwk = nels$homew,  
 ses = nels$ses,  
 public=nels$public,  
 school.id=nels$school.id,  
 n = n,  
 N = N,  
 sdY = sd(nels$math)  
 )   
   
re.mod3 <- "model {   
 # Likelihood: the data model  
 for (i in 1:n) {  
 math[i] ~ dnorm(mmath[i],precision)  
 mmath[i] <- g0 + Uj[school.id[i],1] + g3\*public[i]   
 + (g10 + Uj[school.id[i],2] + g11\*public[i])\*hmwk[i] + g2\*ses[i]   
 }   
 # Random Effects -- note multivariate normal density is used  
 for (j in 1:N) {  
 Uj[j,1:2] ~ dmnorm(mu[1:2],Omega[1:2,1:2])  
 }  
   
 # Priors  
 precision ~ dgamma(0.01,0.01)  
 sigma <- 1/sqrt(precision)  
   
 g0 ~ dnorm(0,1/(100\*sdY^2))  
 g2 ~ dnorm(0,1/(100\*sdY^2))  
 g3 ~ dnorm(0,1/(100\*sdY^2))  
 g10 ~ dnorm(0,1/(100\*sdY^2))  
 g11 ~ dnorm(0,1/(100\*sdY^2))  
   
 mu[1] <- 0 # This identifies the model   
 mu[2] <- 0 #   
   
 Omega[1:2,1:2] ~ dwish(R[,],2.1)  
 R[1,1] <- 1/2.1  
 R[1,2] <- 0  
 R[2,1] <- 0  
 R[2,2] <- 1/2.1  
 Tau <- inverse(Omega)  
 }"  
  
writeLines(re.mod3, con="re.mod3.txt")  
  
start1 = list("precision"=1, "g0"=rnorm(1,0,10),"g2"=rnorm(1,0,10), "g3"=rnorm(1,0,10),"g10"=rnorm(1,0,10), "g11"=rnorm(1,0,10), .RNG.name="base::Wichmann-Hill", .RNG.seed=523)   
start2 = list("precision"=0.5, "g0"=rnorm(1,0,10),"g2"=rnorm(1,0,10), "g3"=rnorm(1,0,10),"g10"=rnorm(1,0,10), "g11"=rnorm(1,0,10), .RNG.name="base::Marsaglia-Multicarry", .RNG.seed=57)  
start3 = list("precision"=0.05,"g0"=rnorm(1,0,10),"g2"=rnorm(1,0,10), "g3"=rnorm(1,0,10),"g10"=rnorm(1,0,10), "g11"=rnorm(1,0,10),.RNG.name="base::Super-Duper", .RNG.seed=24)  
start4 = list("precision"=.002,"g0"=rnorm(1,0,10),"g2"=rnorm(1,0,10), "g3"=rnorm(1,0,10),"g10"=rnorm(1,0,10), "g11"=rnorm(1,0,10),.RNG.name="base::Mersenne-Twister", .RNG.seed=72100)  
start <- list(start1,start2,start3,start4)

If you want to check whether model is OK, then run

#############################################################################  
# Fast way to check your model: rjags and then run for sampling runjags with parallel.  
 re.mod3 <- jags.model(file="re.mod3.txt", # compiles and initializes model  
 data=dataList,  
 inits=start1,  
 n.chains=4,  
 n.adapt=500)   
#############################################################################

If model is OK, then run jags

re.mod3.runjags <- run.jags(model=re.mod3,   
 method="parallel",   
 monitor=c("g0", "g10", "g11","g2","g3","sigma","Tau"),  
 data=dataList,  
 sample=10000,   
 n.chains=4,  
 inits=start,  
 thin=10)

## Calling 4 simulations using the parallel method...  
## Following the progress of chain 1 (the program will wait for all  
## chains to finish before continuing):  
## Welcome to JAGS 4.3.0 on Tue Oct 22 16:21:54 2019  
## 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: 519  
## Unobserved stochastic nodes: 30  
## Total graph size: 3589  
## . Reading parameter file inits1.txt  
## . Initializing model  
## . Adaptation skipped: model is not in adaptive mode.  
## . Updating 4000  
## -------------------------------------------------| 4000  
## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 100%  
## . . . . . . . . Updating 100000  
## -------------------------------------------------| 100000  
## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 100%  
## . . . . Updating 0  
## . Deleting model  
## .   
## All chains have finished  
## Note: the model did not require adaptation  
## Simulation complete. Reading coda files...  
## Coda files loaded successfully  
## Calculating summary statistics...  
## Calculating the Gelman-Rubin statistic for 10 variables....  
## Note: Unable to calculate the multivariate psrf  
## Finished running the simulation

#  
# run these in class but not for Rmd   
# plot(re.mod3.runjags)   
#summary(ris3.lmer)

## Model evaluation

I will just put in large chunks of code and we’ll dicuss in class what they do

#############################################################################  
#############################################################################  
# Model evaluation  
#############################################################################  
#############################################################################  
  
dataList <- list( math = nels$math,  
 hmwk = nels$homew,  
 ses = nels$ses,  
 public=nels$public,  
 school.id=nels$school.id,  
 n = n,  
 N = N,  
 sdY = sd(nels$math)  
 )   
   
re.mod4 <- "model {   
 # Likelihood: the data model  
 for (i in 1:n) {  
 math[i] ~ dnorm(mmath[i],precision)  
 mmath[i] <- g00 + Uj[school.id[i],1] + g3\*public[i]   
 + (g10 + Uj[school.id[i],2] + g11\*public[i])\*hmwk[i]

+ g2\*ses[i]   
  
 intercept.blup[i] <- Uj[school.id[i],1] # intercept random

slope.blup[i] <- Uj[school.id[i],2] # slope random effect  
 marg.post[i] ~ dnorm(mmath[i],precision)   
 }   
 # Random Effects -- note multivariate normal density is used  
 for (j in 1:N) {  
 Uj[j,1:2] ~ dmnorm(mu[1:2],Omega[1:2,1:2])  
 }  
   
 # Priors  
 precision ~ dgamma(0.01,0.01)  
 sigma <- 1/sqrt(precision)  
   
 g00 ~ dnorm(0,1/(100\*sdY^2))  
 g2 ~ dnorm(0,1/(100\*sdY^2))  
 g3 ~ dnorm(0,1/(100\*sdY^2))  
 g10 ~ dnorm(0,1/(100\*sdY^2))  
 g11 ~ dnorm(0,1/(100\*sdY^2))  
   
 mu[1] <- 0 # This identifies the model   
 mu[2] <- 0 #   
   
 Omega[1:2,1:2] ~ dwish(R[,],2.1)  
 R[1,1] <- 1/2.1  
 R[1,2] <- 0  
 R[2,1] <- 0  
 R[2,2] <- 1/2.1  
 Tau <- inverse(Omega)  
 }"  
  
writeLines(re.mod4, con="re.mod4.txt")  
  
start1 = list("precision"=1, "g00"=rnorm(1,0,10),"g2"=rnorm(1,0,10), "g3"=rnorm(1,0,10),"g10"=rnorm(1,0,10), "g11"=rnorm(1,0,10), .RNG.name="base::Wichmann-Hill", .RNG.seed=523)   
start2 = list("precision"=0.5, "g00"=rnorm(1,0,10),"g2"=rnorm(1,0,10), "g3"=rnorm(1,0,10),"g10"=rnorm(1,0,10), "g11"=rnorm(1,0,10), .RNG.name="base::Marsaglia-Multicarry", .RNG.seed=57)  
start3 = list("precision"=0.05,"g00"=rnorm(1,0,10),"g2"=rnorm(1,0,10), "g3"=rnorm(1,0,10),"g10"=rnorm(1,0,10), "g11"=rnorm(1,0,10),.RNG.name="base::Super-Duper", .RNG.seed=24)  
start4 = list("precision"=.002,"g00"=rnorm(1,0,10),"g2"=rnorm(1,0,10), "g3"=rnorm(1,0,10),"g10"=rnorm(1,0,10), "g11"=rnorm(1,0,10),.RNG.name="base::Mersenne-Twister", .RNG.seed=72100)  
start <- list(start1,start2,start3,start4)

Now get samples. Note that this will take longer than model without extra things to monitor and output. The output will be very large and R may not compute summary statistics for you.

re.mod4.runjags <- run.jags(model=re.mod4,   
 method="parallel",   
 monitor=c("g00", "g10", "g11", "g2", "g3", "sigma", "Tau", "marg.post", "intercept.blup","slope.blup"),  
 data=dataList,  
 sample=10000,   
 n.chains=4,  
 inits=start,  
 thin=10)

## Calling 4 simulations using the parallel method...  
## Following the progress of chain 1 (the program will wait for all  
## chains to finish before continuing):  
## Welcome to JAGS 4.3.0 on Tue Oct 22 16:22:29 2019  
## 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: 519  
## Unobserved stochastic nodes: 549  
## Total graph size: 4108  
## . Reading parameter file inits1.txt  
## . Initializing model  
## . Adaptation skipped: model is not in adaptive mode.  
## . Updating 4000  
## -------------------------------------------------| 4000  
## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 100%  
## . . . . . . . . . . . Updating 100000  
## -------------------------------------------------| 100000  
## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 100%  
## . . . . Updating 0  
## . Deleting model  
## .   
## All chains have finished  
## Note: the model did not require adaptation  
## Simulation complete. Reading coda files...  
## Coda files loaded successfully  
## Note: Summary statistics were not produced as there are >50  
## monitored variables  
## [To override this behaviour see ?add.summary and ?runjags.options]  
## FALSEFinished running the simulation

#add.summary(re.mod4.runjags) # Run in class but not Rmd  
#plot(re.mod4.runjags)  
#summary(ris4.lmer)

Now we can look at posertior starting with some data manipulation. I will put the results in ffit and then pull out samples that I want.

ffit <- mcmc(re.mod4.runjags)  
names(ffit) # lets you know what is available

## [1] "mcmc" "deviance.table" "deviance.sum"   
## [4] "pd" "end.state" "samplers"   
## [7] "burnin" "sample" "thin"   
## [10] "model" "data" "monitor"   
## [13] "noread.monitor" "modules" "factories"   
## [16] "response" "residual" "fitted"   
## [19] "method" "method.options" "timetaken"   
## [22] "runjags.version" "summary" "HPD"   
## [25] "hpd" "mcse" "psrf"   
## [28] "autocorr" "crosscorr" "stochastic"   
## [31] "trace" "density" "hist"   
## [34] "ecdfplot" "key" "acplot"   
## [37] "ccplot" "summaries" "summary.available"  
## [40] "summary.pars" "dic"

# posterior samples are in ffit$mcmc ---> there are separate columns for each chain, i.e., 10000 x 4.   
g00 <- as.array(ffit$mcmc[,1])  
g10 <- as.array(ffit$mcmc[,2])  
g11 <- as.array(ffit$mcmc[,3])  
g2 <- as.array(ffit$mcmc[,4])  
g3 <- as.array(ffit$mcmc[,5])  
sigma <- as.array(ffit$mcmc[,6])  
tau11 <- as.array(ffit$mcmc[,7])  
tau12 <- as.array(ffit$mcmc[,8])  
tau22 <- as.array(ffit$mcmc[,10])

Some more data manipulations

# separate columns for each chain, i.e. 10000 x 519 x 4  
margin.post <- as.array(ffit$mcmc[,11:529])

intercept.blup <- as.array(ffit$mcmc[,530:1048])   
slope.blup <- as.array(ffit$mcmc[,1049:1567])

# This will give objects that are 40,000 x 519:   
marginal.means <- rbind(margin.post[,1:519,1], margin.post[,1:519,2], margin.post[,1:519,3], margin.post[,1:519,4])

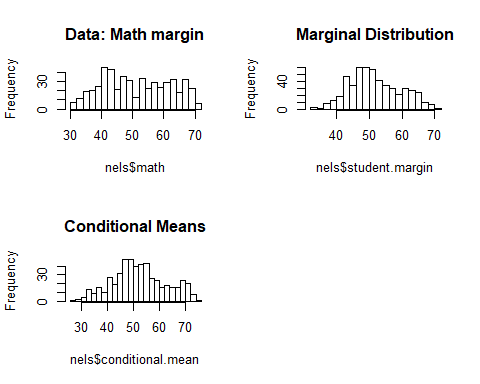
int.blup <- rbind(intercept.blup[,1:519,1], intercept.blup[,1:519,2], intercept.blup[,1:519,3], intercept.blup[,1:519,4])

slp.blup <-rbind(slope.blup[,1:519,1], slope.blup[,1:519,2], slope.blup[,1:519,3], slope.blup[,1:519,4])  
  
  
# Sum over rows to get fitted values (marginal) for each student  
nels$student.margin <- apply(marginal.means,2,"mean")

Using computed values above, plot data and fitted marginal and conditional distributions

# Data & fitted margin  
par(mfrow=c(2,2))  
hist(nels$math,breaks=20,main="Data: Math margin")  
hist(nels$student.margin,breaks=20,main="Marginal Distribution")  
  
   
# to get the conditional margin:   
cond.mean <- marginal.means + int.blup + slp.blup  
nels$conditional.means <- apply(cond.mean,2,"mean")   
   
hist(nels$conditional.mean,breaks=20,main="Conditional Means")   
cor(nels$conditional.mean,nels$math)

## [1] 0.7320163

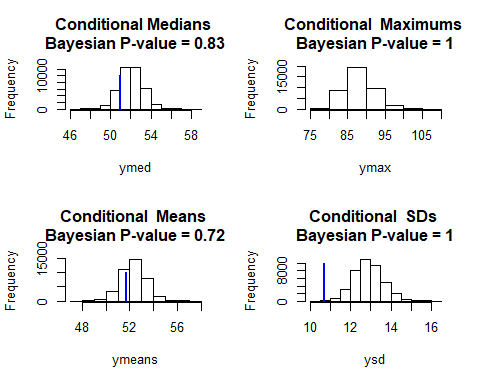


Let’s go the other way: collapse over students

# Compute statistics over columns (students)  
ymeans <- apply(cond.mean,1,"mean")  
ymed <- apply(cond.mean,1,"median")  
ymax <- apply(cond.mean,1,"max")  
ysd <- apply(cond.mean,1,"sd")  
  
# Bayesian p-values  
pmean <- mean(mean(nels$math) < ymeans)  
pmed <- mean(median(nels$math) < ymed)  
pmax <- mean(max(nels$math) < ymax)  
psd <- mean(sd(nels$math) < ysd)

And some graphs for these

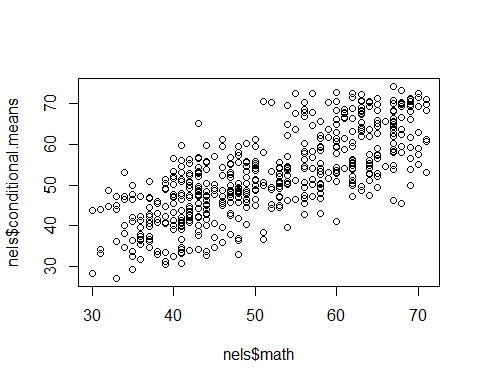
# Graphs of descriptive statistics from posterior predictive distribution   
par(mfrow=c(2,2))  
   
hist(ymed,breaks=10,main=paste("Conditional Medians", "\nBayesian P-value =", round(pmed, 2)))  
lines(c(median(nels$math),median(nels$math)),c(0,10000),col="blue",lwd=2)  
  
hist(ymax,breaks=10,main=paste("Conditional Maximums", "\nBayesian P-value =", round(pmax, 2)))  
lines(c(max(nels$math),max(nels$math)),c(0,10000),col="blue",lwd=2)  
  
hist(ymeans,breaks=10,main=paste("Conditional Means", "\nBayesian P-value =", round(pmean, 2)))  
lines(c(mean(nels$math),mean(nels$math)),c(0,10000),col="blue",lwd=2)  
  
hist(ysd,breaks=10,main=paste("Conditional SDs", "\nBayesian P-value =", round(psd, 2)))  
lines(c(sd(nels$math),sd(nels$math)),c(0,10000),col="blue",lwd=2)



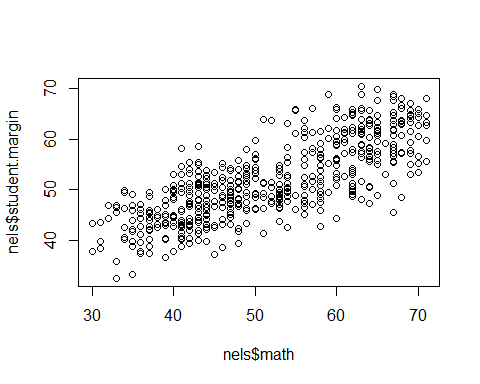
These p-values don’t look at that great.

And if you want to look at the estimated random effects

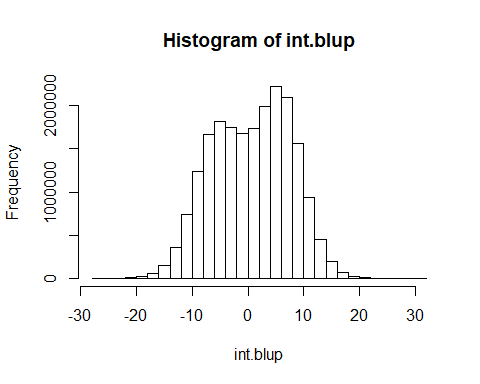
plot(nels$math,nels$conditional.means,type="p")



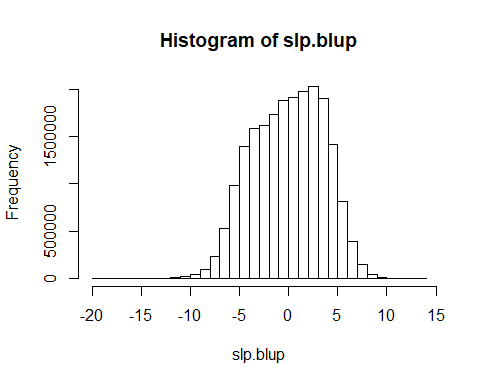
plot(nels$math,nels$student.margin,type="p")



hist(int.blup)



hist(slp.blup)

 ```