Multilevel: Anorexia Data

Carolyn J. Anderson

10/18/2019

In this example, we’ll use jags to fit the following models

* Model 0: Random intercept and no predictors (null HLM)
* Model 1: Random intercept: time
* Model 2: Random intercept: time + treatment
* Model 3: Random intercept: time + treatment + time x treatment
* Model 4: Random slope: time + treatment + time x treatment

# Setup

The packages that we’ll use are

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

Set working directory to where the data live and set up data for initial analyses:

setwd("D:\\Dropbox\\edps 590BAY\\Lectures\\10 Multilevel models")
ano.wide <- read.table("anorexia\_wide.txt",header=TRUE)
ano.wide$change <- ano.wide$weight2 - ano.wide$weight1
(n <- nrow(ano.wide))

## [1] 72

head(ano.wide)

## Rx person weight1 weight2 change
## 1 1 1 80.5 82.2 1.7
## 2 1 2 84.9 85.6 0.7
## 3 1 3 81.5 81.4 -0.1
## 4 1 4 82.6 81.9 -0.7
## 5 1 5 79.9 76.4 -3.5
## 6 1 6 88.7 103.6 14.9

# Examining the data

Let’s do some graphics to look at the data. First we’ll just look at histograms of before treatment, after treatment, and change. We’ll add a 2nd graph of change but looking at is a bit differently. For the 2nd graph of change we need to re-format the data, and create treatment dummy codes

ano <- read.table("anorexia\_long.txt",header=TRUE)
ano <- ano[order(ano$girl),]
ano$Rx1 <- ifelse(ano$Rx==1,1,0)
ano$Rx2 <- ifelse(ano$Rx==2,1,0)
ano$Rx3 <- ifelse(ano$Rx==3,1,0)

and now for the graphs

par(mfrow=c(2,2))
hist(ano.wide$weight1, main='Weight Before Treatment',col="darkseagreen1")
hist(ano.wide$weight2, main='Weight After Treatment',col="darkseagreen1")
hist(ano.wide$change, main='Change in Weight',col="darkseagreen1")
plot(ano$time,ano$weight,
 type="n",
 ylim=c(65,104),
 xlab="Before/After Treatment",
 ylab="Weight in Pounds",
 main="Change for Each Girl"
 )
for (i in 1:n){
 g <- subset(ano,ano$girl==i)
 lines(g$time,g$weight,col=i,lwd=2)
}

 This 4th plot show different intercepts and different slopes. To better see what’s going on, we’ll compute means for each treatement before and after treatement

# means for each treatment
byRx <- aggregate(ano$weight,list(ano$Rx),FUN="mean")

# mean for treatment x time
rx\_by\_time <- aggregate(ano$weight,list(ano$Rx,ano$time),FUN="mean")
names(rx\_by\_time) <-c("rx","time","weight")
rx1 <- subset(rx\_by\_time,rx==1) # Cognitive
rx2 <- subset(rx\_by\_time,rx==2) # Control
rx3 <- subset(rx\_by\_time,rx==3) # Family

Now to look at these…what do you expect based on this?

par(mfrow=c(1,1))
plot(rx2$time,rx2$weight,
 type='b',
 col="cyan",
 main="Weight Change by Treatment",
 ylim=c(80,95),
 xlim=c(1,2),
 xlab="When Weight Measured",
 ylab="Weight",
 xaxt="n"
 )
axis(1, at = seq(1,2,by=1),labels=c("Before","After"))
lines(rx1$time,rx1$weight,type="b",col="blue")
lines(rx3$time,rx3$weight,type="b",col="red")
legend(1.02,95,title="Treatment",legend=c("Control", "Cognitive", "Family"), col=c("cyan","blue","red"),lty=c(1,1,1),cex=0.8)



# Maximum likelihood estimation

Before using Bayesian estimation, we’ll see what restricted maximum likelihhood gives us. I choose REML since these are unbaised; however, if we’re only looking at lmer models from package lme4, I’ld choose MLE. We use the lmer package.

############################################################
# Simple Model using lme4 --- just a random intercept #
############################################################
# Remember: treatment = 1 Cognitive
# = 2 Control
# = 3 Family
model1reml <- lmer(weight ~ time + (1 | girl), data=ano, REML=TRUE)
summary(model1reml)

Here are the fitted values from model 1

girl <- data.frame(seq(1:n))
par(mfrow=c(1,1))

b1 <- data.frame(coef(model1reml)[[1]])
tmp1 <- cbind(b1,girl)
names(tmp1) <- c("b0","b1","girl")
ano <- merge(ano,tmp1,by="girl")
ano$y.1 <- ano$b0 + ano$b1\*ano$time
y <- subset(ano,ano$girl==1)
plot(y$time,y$y.1,
 type="n",
 ylim=c(65,104),
 xlab="Before/After Treatment",
 ylab="Weight in Pounds",
 main="weight ~ time + (1 | girl)",
 xaxt="n"
 )
axis(1, at = seq(1,2,by=1),labels=c("Before","After"))
for (i in 1:n){
 g <- subset(ano,ano$girl==i)
 lines(g$time,g$y.1)
}

 Now make the model a bit more complex by adding in treatment

##############################################################
# #
# Model 2: weight ~ time + treatment + (1 | girl) #
###############################################################
ano$treatment <- as.factor(ano$Rx)

model2reml <- lmer(weight ~ time + treatment + (1 | girl), data=ano, REML=TRUE)
summary(model2reml)

Now graphing conditional values

b2 <- data.frame(coef(model2reml)[[1]])
tmp2 <- cbind(b2,girl)
names(tmp2) <- c("b0.2","b1.2","b2.2","b3.2","girl")
ano <- merge(ano,tmp2,by="girl")
ano$y.2 <- ano$b0.2 + ano$b1.2\*ano$time + ano$b2.2\*ano$Rx2 + ano$b3.2\*ano$Rx3
y <- subset(ano,ano$girl==1)
plot(y$time,y$y.2,
 type="n",
 ylim=c(65,104),
 xlab="Before/After Treatment",
 ylab="Weight in Pounds",
 main="weight ~ time + treatment + (1 | girl)",
 xaxt="n"
 )
axis(1, at = seq(1,2,by=1),labels=c("Before","After"))
for (i in 1:n){
 g <- subset(ano,ano$girl==i)
 lines(g$time,g$y.2)
}

 Doesn’t look much different, so let’s add an interaction between time and treatment. We see this is in the data.

#####################################################################
# Model 3: weight ~ time + treatment + time\*treatment + ( 1 | girl) #
#####################################################################
model3reml <- lmer(weight ~ time + treatment + time\*treatment + ( 1 | girl),data=ano,REML=TRUE)
summary(model3reml)

The fitted values are

b3 <- data.frame(coef(model3reml)[[1]])
tmp3 <- cbind(b3,girl)
names(tmp3) <- c("b0.3","b1.3","b2.3","b3.3","b4.3","b5.3","girl")
ano <- merge(ano,tmp3,by="girl")
ano$y.3 <- ano$b0.3 + ano$b1.3\*ano$time + ano$b2.3\*ano$Rx2 + ano$b3.3\*ano$Rx3 + ano$b4.3\*ano$time\*ano$Rx2 + ano$b5.3\*ano$time\*ano$Rx3
y <- subset(ano,ano$girl==1)
plot(y$time,y$y.3,
 type="n",
 ylim=c(65,104),
 xlab="Before/After Treatment",
 ylab="Weight in Pounds",
 main="~time+treatment+time\*treatment+( 1 | girl)",
 xaxt="n"
 )
axis(1, at = seq(1,2,by=1),labels=c("Before","After"))
for (i in 1:n){
 g <- subset(ano,ano$girl==i)
 lines(g$time,g$y.3)
}

 Based on our initial look at the data, we might want a random slope but NO random intercept.

#####################################################################
# Model 4: weight ~ time + treatment + time\*treatment + ( 0 +time|girl) #
#####################################################################
model4reml <- lmer(weight ~ time + treatment + time\*treatment + ( 1 | girl) + ( 0 + time |girl),data=ano,REML=TRUE)

## boundary (singular) fit: see ?isSingular

model4reml <- lmer(weight ~ time + treatment + time\*treatment + ( 0 + time |girl),data=ano,REML=TRUE)
summary(model4reml)

What are fitted values now? These “look” like the data.

b4 <- data.frame(coef(model4reml)[[1]])
tmp4 <- cbind(b4,girl)
names(tmp4) <- c("b0.4","b1.4","b2.4","b3.4","b4.4","b5.4","girl")
ano <- merge(ano,tmp4,by="girl")
ano$y.4 <- ano$b0.4 + ano$b1.4\*ano$time + ano$b2.4\*ano$Rx2 + ano$b3.4\*ano$Rx3 + ano$b4.4\*ano$time\*ano$Rx2 + ano$b5.4\*ano$time\*ano$Rx3
plot(ano$time,ano$y.4,
 type="n",
 ylim=c(65,104),
 xlab="Before/After Treatment",
 ylab="Weight in Pounds",
 main="~ ... +time\*treatment+(0+time|girl)",
 xaxt="n"
 )
axis(1, at = seq(1,2,by=1),labels=c("Before","After"))
for (i in 1:n){
 g <- subset(ano,ano$girl==i)
 lines(g$time,g$y.4)
}



# Bayesian estimation

The random intercept model

##################################################################################
# Bayesian Model 0 : random intercept no predictors (null hlm) #
##################################################################################
dataList <- list(
 y = ano$weight,
 sdY = sd(ano$weight),
 n = length(ano$weight),
 ng= length(ano$weight)/2,
 girl= ano$girl
 )

The model part

model0 <- "model {
 for (i in 1:n) { # likelihood
 y[i] ~ dnorm(mu[i],precision)
 mu[i] <- U0j[girl[i]]
 }
 for (j in 1:ng) { # random intercepts
 U0j[j] ~ dnorm(g0,ptau)
 }
 g0 ~ dnorm(0,1/(100\*sdY^2)) # other priors
 tau ~ dunif(0.0001,200)
 ptau <- 1/tau^2
 sigma ~ dunif(0.0001,2000)
 precision <- 1/sigma^2
}"

writeLines(model0, con="model0.txt")

I used different random number generators to show you what is available. This shouldn’t make a difference in results.

start1 = list("g0"=mean(ano$weight), "sigma"=sd(ano$weight), "tau"=.5, .RNG.name="base::Wichmann-Hill", .RNG.seed=523)
start2 = list("g0"=rnorm(1,0,3), "sigma"=5, "tau"=1, .RNG.name="base::Marsaglia-Multicarry", .RNG.seed=57)
start3 = list("g0"=rnorm(1,3,4), "sigma"=10, "tau"=5, .RNG.name="base::Super-Duper", .RNG.seed=24)
start4 = list("g0"=rnorm(1,-3,10), "sigma"=50, "tau"=20, .RNG.name="base::Mersenne-Twister", .RNG.seed=72100)

start <- list(start1,start2,start3,start4)

And run jags

model0.runjags <- run.jags(model=model0,
 method="parallel",
 monitor=c("g0", "sigma", "tau"),
 data=dataList,
 n.chains=4,
 sample=20000,
 burnin=5000,
 inits=start,
 thin=15) # seems I need to thin

## 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 Sat Oct 19 14:26:00 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: 144
## Unobserved stochastic nodes: 75
## Total graph size: 380
## . Reading parameter file inits1.txt
## . Initializing model
## . Adapting 1000
## -------------------------------------------------| 1000
## ++++++++++++++++++++++++++++++++++++++++++++++++++ 100%
## Adaptation successful
## . Updating 5000
## -------------------------------------------------| 5000
## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 100%
## . . . . Updating 300000
## -------------------------------------------------| 300000
## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 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 3 variables....
## Finished running the simulation

print(model0.runjags)

##
## JAGS model summary statistics from 80000 samples (thin = 15; chains = 4; adapt+burnin = 6000):
##
## Lower95 Median Upper95 Mean SD Mode MCerr MC%ofSD SSeff
## g0 82.506 83.789 85.045 83.787 0.64527 -- 0.0034704 0.5 34572
## sigma 5.1303 6.0997 7.1926 6.1304 0.53281 -- 0.0040381 0.8 17410
## tau 0.85107 3.308 5.1946 3.186 1.0636 -- 0.011919 1.1 7963
##
## AC.150 psrf
## g0 0.0047032 0.99998
## sigma 0.059015 1.0003
## tau 0.16269 1.0011
##
## Total time taken: 39.3 seconds

summary(model1reml)

## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: weight ~ time + (1 | girl)
## Data: ano
##
## REML criterion at convergence: 947.5
##
## Scaled residuals:
## Min 1Q Median 3Q Max
## -1.92295 -0.46924 -0.03478 0.52092 2.24652
##
## Random effects:
## Groups Name Variance Std.Dev.
## girl (Intercept) 13.84 3.720
## Residual 31.87 5.645
## Number of obs: 144, groups: girl, 72
##
## Fixed effects:
## Estimate Std. Error df t value Pr(>|t|)
## (Intercept) 79.6444 1.5509 99.2652 51.353 < 2e-16 \*\*\*
## time 2.7639 0.9409 71.0000 2.938 0.00446 \*\*
## ---
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
## (Intr)
## time -0.910

plot(model0.runjags)

## Generating plots...



Now for our next model where we add in a predictor, “time”:

##################################################################################
# Bayesian Model 1 : random intercept and fixed time #
##################################################################################
dataList <- list(
 y = ano$weight,
 time = ano$time,
 sdY = sd(ano$weight),
 n = length(ano$weight),
 ng= length(ano$weight)/2,
 girl= ano$girl
 )

model1 <- "model {
 for (i in 1:n) {
 y[i] ~ dnorm(mu[i],precision)
 mu[i] <- b0j[girl[i]] + g1\*time[i]
 }

 for (j in 1:ng) {
 b0j[j] ~ dnorm(g0,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(model1, con="model1.txt")

start1 = list("g0"=mean(ano$weight),"g1"=rnorm(1,0,3), "sigma"=sd(ano$weight), "tau"=.5,
 .RNG.name="base::Wichmann-Hill", .RNG.seed=523)
start2 = list("g0"=rnorm(1,0,3), "g1"=rnorm(1,0,3), "sigma"=5, "tau"=1,
 .RNG.name="base::Marsaglia-Multicarry", .RNG.seed=57)
start3 = list("g0"=rnorm(1,3,4), "g1"=rnorm(1,3,4), "sigma"=50, "tau"=3,
 .RNG.name="base::Super-Duper", .RNG.seed=24)
start4 = list("g0"=rnorm(1,-3,10), "g1"=rnorm(1,-3,10), "sigma"=10, "tau"=10,
 .RNG.name="base::Mersenne-Twister", .RNG.seed=72100)

start <- list(start1,start2,start3,start4)

start <- list(start1,start2,start3,start4)

model1.runjags <- run.jags(model=model1,
 method="parallel",
 monitor=c("g0", "g1", "sigma", "tau"),
 data=dataList,
 sample=20000,
 n.chains=4,
 thin=19,
 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 Sat Oct 19 14:26:45 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: 144
## Unobserved stochastic nodes: 76
## Total graph size: 671
## . Reading parameter file inits1.txt
## . Initializing model
## . Adapting 1000
## -------------------------------------------------| 1000
## ++++++++++++++++++++++++++++++++++++++++++++++++++ 100%
## Adaptation successful
## . Updating 4000
## -------------------------------------------------| 4000
## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 100%
## . . . . . Updating 380000
## -------------------------------------------------| 380000
## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 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

print(model1.runjags)

##
## JAGS model summary statistics from 80000 samples (thin = 19; chains = 4; adapt+burnin = 5000):
##
## Lower95 Median Upper95 Mean SD Mode MCerr MC%ofSD SSeff
## g0 76.493 79.611 82.782 79.612 1.6118 -- 0.010852 0.7 22060
## g1 0.88722 2.7814 4.7411 2.7813 0.98229 -- 0.0067432 0.7 21220
## sigma 4.8666 5.7785 6.8845 5.8215 0.51882 -- 0.0035265 0.7 21645
## tau 1.5692 3.6118 5.3758 3.5264 0.94956 -- 0.0087984 0.9 11648
##
## AC.190 psrf
## g0 0.019679 1.0002
## g1 0.022788 1.0003
## sigma 0.029229 1
## tau 0.083317 1
##
## Total time taken: 46.8 seconds

# For comparison
summary(model1reml)

## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: weight ~ time + (1 | girl)
## Data: ano
##
## REML criterion at convergence: 947.5
##
## Scaled residuals:
## Min 1Q Median 3Q Max
## -1.92295 -0.46924 -0.03478 0.52092 2.24652
##
## Random effects:
## Groups Name Variance Std.Dev.
## girl (Intercept) 13.84 3.720
## Residual 31.87 5.645
## Number of obs: 144, groups: girl, 72
##
## Fixed effects:
## Estimate Std. Error df t value Pr(>|t|)
## (Intercept) 79.6444 1.5509 99.2652 51.353 < 2e-16 \*\*\*
## time 2.7639 0.9409 71.0000 2.938 0.00446 \*\*
## ---
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
## (Intr)
## time -0.910

# And checking a bit more whether model1.runjags converged
plot(model1.runjags)

## Generating plots...

 This model is too simple for the data, so we’ll go on to our next model, model2 which adds treatment as a fixed effect.

##################################################################################
# Bayesian Model 2 : random intercept + fixed time + Rx #
##################################################################################
dataList <- list(
 y = ano$weight,
 time = ano$time,
 rx1 = ano$Rx1,
 rx3 = ano$Rx3,
 sdY = sd(ano$weight),
 n = length(ano$weight),
 ng= length(ano$weight)/2,
 girl= ano$girl
 )

model2 <- "model {
 for (i in 1:n) {
 y[i] ~ dnorm(mu[i],precision)
 mu[i] <- b0j[girl[i]] + g1\*time[i] + g2\*rx1[i] + g3\*rx3[i]
 }

 for (j in 1:ng) {
 b0j[j] ~ dnorm(g0,ptau)
 }

 g0 ~ dnorm(0,1/(100\*sdY^2))
 g1 ~ dnorm(0,1/(100\*sdY^2))
 g2 ~ dnorm(0,1/(100\*sdY^2))
 g3 ~ 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(model2, con="model2.txt")

start1 = list("g0"=mean(ano$weight),"g1"=rnorm(1,0,3), "g2"=rnorm(1,0,3),"g3"=rnorm(1,0,3),
 "sigma"=sd(ano$weight), "tau"=.5, .RNG.name="base::Wichmann-Hill", .RNG.seed=523)

start2 = list("g0"=rnorm(1,0,3), "g1"=rnorm(1,0,3), "g2"=rnorm(1,-1,3), "g3"=rnorm(1,0,3),
 "sigma"=5, "tau"=1, .RNG.name="base::Marsaglia-Multicarry", .RNG.seed=57)

start3 = list("g0"=rnorm(1,3,4), "g1"=rnorm(1,3,4), "g2"=rnorm(1,3,4), "g3"=rnorm(1,0,3),
 "sigma"=50, "tau"=3, .RNG.name="base::Super-Duper", .RNG.seed=24)

start4 = list("g0"=rnorm(1,-3,10), "g1"=rnorm(1,-3,10), "g2"=rnorm(1,10,5), "g3"=rnorm(1,0,3),
 "sigma"=10, "tau"=10, .RNG.name="base::Mersenne-Twister", .RNG.seed=72100)

start <- list(start1,start2,start3,start4)

model2.runjags <- run.jags(model=model2,
 method="parallel",
 monitor=c("g0", "g1", "g2","g3","sigma", "tau"),
 data=dataList,
 sample=20000,
 n.chains=4,
 thin=10,
 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 Sat Oct 19 14:27:38 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: 144
## Unobserved stochastic nodes: 78
## Total graph size: 965
## . Reading parameter file inits1.txt
## . Initializing model
## . Adapting 1000
## -------------------------------------------------| 1000
## ++++++++++++++++++++++++++++++++++++++++++++++++++ 100%
## Adaptation successful
## . Updating 4000
## -------------------------------------------------| 4000
## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 100%
## . . . . . . . Updating 200000
## -------------------------------------------------| 200000
## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 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 6 variables....
## Finished running the simulation

print(model2.runjags)

##
## JAGS model summary statistics from 80000 samples (thin = 10; chains = 4; adapt+burnin = 5000):
##
## Lower95 Median Upper95 Mean SD Mode MCerr MC%ofSD SSeff
## g0 73.576 77.146 80.641 77.142 1.7881 -- 0.019064 1.1 8797
## g1 0.85512 2.7795 4.725 2.7819 0.98242 -- 0.0096204 1 10428
## g2 0.13327 2.8753 5.5987 2.8756 1.3914 -- 0.0085793 0.6 26301
## g3 2.3475 5.5404 8.6962 5.5415 1.6102 -- 0.0092625 0.6 30222
## sigma 4.9011 5.7972 6.8183 5.8221 0.49718 -- 0.004364 0.9 12979
## tau 0.74642 2.9926 4.7422 2.8955 0.99558 -- 0.013205 1.3 5684
##
## AC.100 psrf
## g0 0.12171 1
## g1 0.10122 1.0001
## g2 0.020866 1.0002
## g3 0.017546 1
## sigma 0.078552 1.0003
## tau 0.2273 1.0018
##
## Total time taken: 29.7 seconds

# For comparison
summary(model2reml)

## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: weight ~ time + treatment + (1 | girl)
## Data: ano
##
## REML criterion at convergence: 930.8
##
## Scaled residuals:
## Min 1Q Median 3Q Max
## -2.0131 -0.5380 -0.0562 0.6090 2.3744
##
## Random effects:
## Groups Name Variance Std.Dev.
## girl (Intercept) 10.04 3.168
## Residual 31.87 5.645
## Number of obs: 144, groups: girl, 72
##
## Fixed effects:
## Estimate Std. Error df t value Pr(>|t|)
## (Intercept) 80.0473 1.6992 123.5077 47.108 < 2e-16 \*\*\*
## time 2.7639 0.9409 71.0000 2.938 0.00446 \*\*
## treatment2 -2.8604 1.3764 69.0000 -2.078 0.04141 \*
## treatment3 2.6687 1.5567 69.0000 1.714 0.09096 .
## ---
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
## (Intr) time trtmn2
## time -0.831
## treatment2 -0.383 0.000
## treatment3 -0.339 0.000 0.418

# And we should do some more model checking...
plot(model2.runjags)

## Generating plots...



Next model (this will run fine but it is not run here)

##################################################################################
# Bayesian Model 3 : random intercept + fixed time + Rx + time\*Rx #
##################################################################################
dataList <- list(
 y = ano$weight,
 time = ano$time,
 rx1 = ano$Rx1,
 rx3 = ano$Rx3,
 sdY = sd(ano$weight),
 n = length(ano$weight),
 ng= length(ano$weight)/2,
 girl= ano$girl
 )

model3 <- "model {
 for (i in 1:n) {
 y[i] ~ dnorm(mu[i],precision)
 mu[i] <- b0j[girl[i]] + g1\*time[i] + g2\*rx1[i] + g3\*rx3[i] + g4\*time[i]\*rx1[i] + g5\*time[i]\*rx3[i]
 }

 for (j in 1:ng) {
 b0j[j] ~ dnorm(g0,ptau)
 }

 g0 ~ dnorm(0,1/(100\*sdY^2))
 g1 ~ dnorm(0,1/(100\*sdY^2))
 g2 ~ dnorm(0,1/(100\*sdY^2))
 g3 ~ dnorm(0,1/(100\*sdY^2))
 g4 ~ dnorm(0,1/(100\*sdY^2))
 g5 ~ 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(model3, con="model3.txt")

start1 = list("g0"=mean(ano$weight),"g1"=rnorm(1,0,3), "g2"=rnorm(1,0,3),"g3"=rnorm(1,0,3),
 "g4"=rnorm(1,0,3),"g5"=rnorm(1,0,3), "sigma"=sd(ano$weight), "tau"=.5,
 .RNG.name="base::Wichmann-Hill", .RNG.seed=523)

start2 = list("g0"=rnorm(1,0,3), "g1"=rnorm(1,0,3), "g2"=rnorm(1,-1,3), "g3"=rnorm(1,0,3),
 "g4"=rnorm(1,-1,3), "g5"=rnorm(1,0,3), "sigma"=5, "tau"=1,
 .RNG.name="base::Marsaglia-Multicarry", .RNG.seed=57)

start3 = list("g0"=rnorm(1,3,4), "g1"=rnorm(1,3,4), "g2"=rnorm(1,3,4), "g3"=rnorm(1,0,3),
 "g4"=rnorm(1,3,4), "g5"=rnorm(1,0,3), "sigma"=50, "tau"=5,
 .RNG.name="base::Super-Duper", .RNG.seed=24)

start4 = list("g0"=rnorm(1,-3,10),"g1"=rnorm(1,-3,10), "g2"=rnorm(1,10,5), "g3"=rnorm(1,0,3),
 "g4"=rnorm(1,2,5),
 "g5"=rnorm(1,3,3), "sigma"=10, "tau"=10,
 .RNG.name="base::Mersenne-Twister", .RNG.seed=72100)

start <- list(start1,start2,start3,start4)

model3.runjags <- run.jags(model=model3,
 method="parallel",
 monitor=c("g0","g1","g2","g3","g4","g5","sigma","tau"),
 data=dataList,
 sample=20000,
 n.chains=4,
 thin=20,
 inits=start)

print(model3.runjags)
summary(model3.reml)
plot(model3.runjags)

Now we’ll switch to random slope and drop the random intercept. This might be OK. This code will run but not run here.

##################################################################################
# Bayesian Model 4 : random slope + fixed time + Rx + time\*Rx #
##################################################################################
dataList <- list(
 y = ano$weight,
 time = ano$time,
 rx1 = ano$Rx1,
 rx3 = ano$Rx3,
 sdY = sd(ano$weight),
 n = length(ano$weight),
 ng= length(ano$weight)/2,
 girl= ano$girl
 )

model4 <- "model {
 for (i in 1:n) {
 y[i] ~ dnorm(mu[i],precision)
 mu[i] <- g0 + g1\*time[i] + b1j[girl[i]]\*time[i] + g2\*rx1[i] + g3\*rx3[i] + g4\*time[i]\*rx1[i] + g5\*time[i]\*rx3[i]
 }

 for (j in 1:ng) {
 b1j[j] ~ dnorm(0,ptau)
 }

 g0 ~ dnorm(0,1/(100\*sdY^2))
 g1 ~ dnorm(0,1/(100\*sdY^2))
 g2 ~ dnorm(0,1/(100\*sdY^2))
 g3 ~ dnorm(0,1/(100\*sdY^2))
 g4 ~ dnorm(0,1/(100\*sdY^2))
 g5 ~ 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(model4, con="model4.txt")

start1 = list("g0"=mean(ano$weight), "g1"=rnorm(1,1,3), "g2"=rnorm(1,0,3),"g3"=rnorm(1,0,3),
 "g4"=rnorm(1,0,3), "g5"=rnorm(1,0,3), "sigma"=sd(ano$weight), "tau"=.5,
 .RNG.name="base::Wichmann-Hill", .RNG.seed=523)

start2 = list("g0"=rnorm(1,0,3), "g1"=rnorm(1,-2,3), "g2"=rnorm(1,-1,3), "g3"=rnorm(1,0,3),
 "g4"=rnorm(1,-1,3),"g5"=rnorm(1,0,3), "sigma"=5, "tau"=1,
 .RNG.name="base::Marsaglia-Multicarry", .RNG.seed=57)

start3 = list("g0"=rnorm(1,3,4), "g1"=rnorm(1,0,3), "g2"=rnorm(1,3,4), "g3"=rnorm(1,0,3),
 "g4"=rnorm(1,3,4), "g5"=rnorm(1,0,3), "sigma"=50, "tau"=5,
 .RNG.name="base::Super-Duper", .RNG.seed=24)

start4 = list("g0"=rnorm(1,-3,10), "g1"=rnorm(1,5,3),"g2"=rnorm(1,10,5), "g3"=rnorm(1,0,3),
 "g4"=rnorm(1,2,5), "g5"=rnorm(1,3,3), "sigma"=10, "tau"=10,
 .RNG.name="base::Mersenne-Twister", .RNG.seed=72100)

start <- list(start1,start2,start3,start4)

model4.runjags <- run.jags(model=model4,
 method="parallel",
 monitor=c("g0","g1","g2","g3","g4","g5", "sigma", "tau"),
 data=dataList,
 sample=20000,
 n.chains=4,
 thin=5,
 inits=start)

print(model4.runjags)
summary(model4reml)
plot(model4.runjags)

We’ll try a random intercept and slope. With only 2 time points, you can’t fit both random intercept and slope as well as correlation between the random intercept and slope. In this example,

$$τ\_{01}=0$$

. This seems to be just enough to get the model to fit. Let’s see what happens when you try to get estimate both.

##################################################################################
# Bayesian Model 5: random intercept & slope + fixed time + Rx1 + Rx3 #
# density of tau\_0 doesn't look good #
##################################################################################
dataList <- list(
 y = ano$weight,
 time = ano$time,
 rx1 = ano$Rx1,
 rx3 = ano$Rx3,
 sdY = sd(ano$weight),
 n = length(ano$weight),
 ng= length(ano$weight)/2,
 girl= ano$girl
 )

model5 <- "model {
 for (i in 1:n) {
 y[i] ~ dnorm(mu[i],precision)
 mu[i] <- g0 + g1\*time[i] + g2\*rx1[i] + g3\*rx3[i] + U0j[girl[i]] + U1j[girl[i]]\*time[i]
 }

 for (j in 1:ng) {
 U0j[j] ~ dnorm(0,ptau0)
 U1j[j] ~ dnorm(0,ptau1)
 }

 g0 ~ dnorm(0,1/(100\*sdY^2))
 g1 ~ dnorm(0,1/(100\*sdY^2))
 g2 ~ dnorm(0,1/(100\*sdY^2))
 g3 ~ dnorm(0,1/(100\*sdY^2))

 ptau0 ~ dgamma(0.001,0.001)
 tau0 <- 1/sqrt(ptau0)

 tau1 ~ dunif(0.0001,200)
 ptau1 <- 1/tau1^2

 sigma ~ dunif(0.0001,2000)
 precision <- 1/sigma^2
 }"

writeLines(model5, con="model5.txt")

start1 = list("g0"=mean(ano$weight), "g1"=rnorm(1,1,3), "g2"=rnorm(1,1,3), "g3"=rnorm(1,4,3), "sigma"=sd(ano$weight), "ptau0"=.005, "tau1"=2,
 .RNG.name="base::Wichmann-Hill", .RNG.seed=523)

start2 = list("g0"=dnorm(1,0,3), "g1"=rnorm(1,-2,3), "g2"=rnorm(1,0,3), "g3"=rnorm(1,-1,3), "sigma"=5, "ptau0"=.1, "tau1"=3,
 .RNG.name="base::Marsaglia-Multicarry", .RNG.seed=57)

start3 = list("g0"=dnorm(1,3,4), "g1"=rnorm(1,0,3), "g2"=rnorm(1,-1,3), "g3"=rnorm(1,0,4), "sigma"=50, "ptau0"=.045, "tau1"=10,
 .RNG.name="base::Super-Duper", .RNG.seed=24)

start4 = list("g0"=dnorm(1,-3,10), "g1"=rnorm(1,5,3), "g2"=rnorm(1,5,3), "g3"=rnorm(1,3,4), "sigma"=10, "ptau0"=.10, "tau1"=1,
 .RNG.name="base::Mersenne-Twister", .RNG.seed=72100)

start <- list(start1,start2,start3,start4)

model5.runjags <- run.jags(model=model5,
 method="parallel",
 monitor=c("g0","g1","g2","g3","sigma","ptau0","ptau1","tau0", "tau1"),
 data=dataList,
 sample=20000,
 n.chains=4,
 thin=20,
 inits=start)

print(model5.runjags)
plot(model5.runjags)

So, we’ll take Model 4 as our final model and get some additional statistics from the code (i.e., draws from posterior distribution).

##################################################################################
# Bayesian Model 4 refinement : random slope + fixed time + rx3+ time\*Rx3 #
# 1st: drop time\*Rx1
# 2nd: drop Rx1
##################################################################################
dataList <- list(
 y = ano$weight,
 time = ano$time,
 rx3 = ano$Rx3,
 sdY = sd(ano$weight),
 n = length(ano$weight),
 ng= length(ano$weight)/2,
 girl= ano$girl
 )

model4 <- "model {
 for (i in 1:n) {
 y[i] ~ dnorm(mu[i],precision)
 mu[i] <- g0 + g1\*time[i] + b1j[girl[i]]\*time[i] + g3\*rx3[i] + g5\*time[i]\*rx3[i]
 yhat[i] <- g0 + g1\*time[i] + b1j[girl[i]]\*time[i] + g3\*rx3[i] + g5\*time[i]\*rx3[i]
 }

 for (j in 1:ng) {
 b1j[j] ~ dnorm(0,ptau)
 }

 g0 ~ dnorm(0,1/(100\*sdY^2))
 g1 ~ dnorm(0,1/(100\*sdY^2))
 g3 ~ dnorm(0,1/(100\*sdY^2))
 g5 ~ 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(model4, con="model4.txt")

start1 = list("g0"=mean(ano$weight), "g1"=dnorm(1,1,3), "g3"=dnorm(1,0,3),
 "g5"=dnorm(1,0,3), "sigma"=sd(ano$weight), "tau"=.5,
 .RNG.name="base::Wichmann-Hill", .RNG.seed=523)

start2 = list("g0"=dnorm(1,0,3), "g1"=dnorm(1,-2,3), "g3"=dnorm(1,0,3),
 "g5"=dnorm(1,0,3), "sigma"=5, "tau"=1,
 .RNG.name="base::Marsaglia-Multicarry", .RNG.seed=57)

start3 = list("g0"=dnorm(1,3,4), "g1"=dnorm(1,0,3), "g3"=dnorm(1,0,3),
 "g5"=dnorm(1,0,3), "sigma"=50, "tau"=5,
 .RNG.name="base::Super-Duper", .RNG.seed=24)

start4 = list("g0"=dnorm(1,-3,10), "g1"=dnorm(1,5,3),"g3"=dnorm(1,0,3),
 "g5"=dnorm(1,3,3), "sigma"=10, "tau"=10,
 .RNG.name="base::Mersenne-Twister", .RNG.seed=72100)

start <- list(start1,start2,start3,start4)

model4.runjags <- run.jags(model=model4,
 method="parallel",
 monitor=c("g0","g1","g3","g5", "sigma", "tau","yhat"),
 data=dataList,
 sample=20000,
 n.chains=4,
 thin=5,
 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 Sat Oct 19 14:28:16 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: 144
## Unobserved stochastic nodes: 78
## Total graph size: 967
## . 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
## 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

print(model4.runjags)

##
## JAGS model with 80000 samples (thin = 5; chains = 4; adapt+burnin = 5000)
##
## Full summary statistics have not been pre-calculated - use either the summary method or add.summary to calculate summary statistics

summary(model4reml)

## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: weight ~ time + treatment + time \* treatment + (0 + time | girl)
## Data: ano
##
## REML criterion at convergence: 906
##
## Scaled residuals:
## Min 1Q Median 3Q Max
## -2.59369 -0.46354 -0.00561 0.49016 1.64149
##
## Random effects:
## Groups Name Variance Std.Dev.
## girl time 6.891 2.625
## Residual 22.913 4.787
## Number of obs: 144, groups: girl, 72
##
## Fixed effects:
## Estimate Std. Error df t value Pr(>|t|)
## (Intercept) 79.683 1.988 69.000 40.090 <2e-16 \*\*\*
## time 3.007 1.348 104.632 2.230 0.0279 \*
## treatment2 2.325 2.891 69.000 0.804 0.4240
## treatment3 -3.718 3.270 69.000 -1.137 0.2594
## time:treatment2 -3.457 1.961 104.632 -1.763 0.0808 .
## time:treatment3 4.258 2.218 104.632 1.920 0.0576 .
## ---
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
## (Intr) time trtmn2 trtmn3 tm:tr2
## time -0.885
## treatment2 -0.688 0.608
## treatment3 -0.608 0.538 0.418
## tim:trtmnt2 0.608 -0.688 -0.885 -0.370
## tim:trtmnt3 0.538 -0.608 -0.370 -0.885 0.418

Why didn’t {r} print(model4.runjags) give you the table of statistics? There are many of them; in particular, there are an extra 72 X 2 = 143 yhat values, 2 values per girl (weight before and weight after).

plot(model4.runjags)

Let’s do some other things will resultings samples.

# Takes output and put in format so can use coda plots and diagnostics on it.
samp4 <- as.mcmc.list(model4.runjags) # change to mcmc object so that can use coda.
samples <- as.array(samp4) # change from mcmc to array object, one from each chain.
sampled <- rbind(samples[,,1],samples[,,2],samples[, , 3], samples[, , 4])
write(sampled,"sampled\_model4.txt")

g0 <- sampled[,1]
g1 <- sampled[,2]
g3 <- sampled[,3]
g5 <- sampled[,4]
sigma <- sampled[,5]
tau <- sampled[,6]
fitted <- sampled[,7:150]

ano$yhat <- apply(fitted,2,"mean")
plot(ano$time,ano$yhat,
 type="n",
 ylim=c(65,104),
 xlab="Before/After Treatment",
 ylab="Weight in Pounds",
 main="Bayesian refined Model 4",
 xaxt="n"
 )
axis(1, at = seq(1,2,by=1),labels=c("Before","After"))
for (i in 1:n){
 g <- subset(ano,ano$girl==i)
 lines(g$time,g$yhat)
}

