One Chain, Two Chain, Three Chain, More?

Zachary Roman

December 4, 2017

The Problem

The goal of this project is to gain insight into how many chains to select in a Bayesian MCMC analysis. I found myself interested in this due to a confounding situation that occured during a routine MCMC analysis. I conducted ana anlysis with 4 MCMC chains, 1 chain was divergent, which in turn lead to non-convergence. This peaked my curriosity, I decided to re-run the analysis with 2 chains. The second situation provided excellent fit with no divergent chains. This is a dangerous situation; the decision of the number of chains to employ could completely dictate our confidence in convergence and consequently weather or not to interpret the results as meaningful.

Method

To explore the effect of the number of chains on convergence a simulation was conducted. Four major steps were conducted:

1. Data Generation
   - A single dataset is generated following a multi-level data structure.
2. MCMC Analysis
   - An MCMC multi-evel model is cunducted to analyze the data 3 times (under different prior conditions). Each model has 10 MCMC chains.
3. Sampling of Chains (Simulation Stage)
   - Chains are sampled in different amounts (2 through 10) without replacement a total of 1,000 times each.
4. Aggregation & Diagnostics
   - R-hat and Bias are computed for each sample of chains. Comparisons are made accross conditions and the number of chanis sampled.

Data Generation

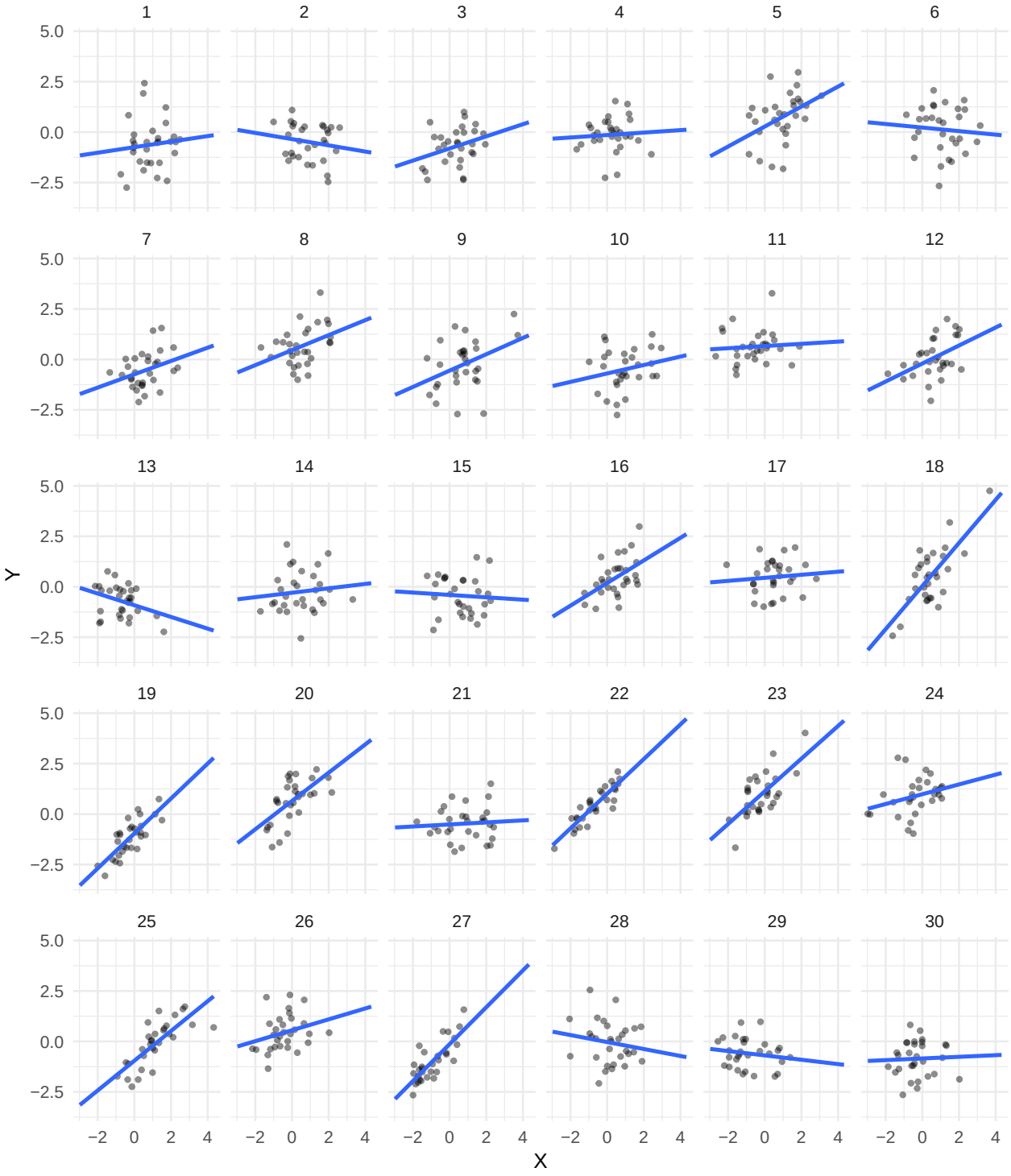
The original analysis that peaked my interest was a realitively complex Multi-Level Model. Therefore, for this simulation I decided to generate similar data.

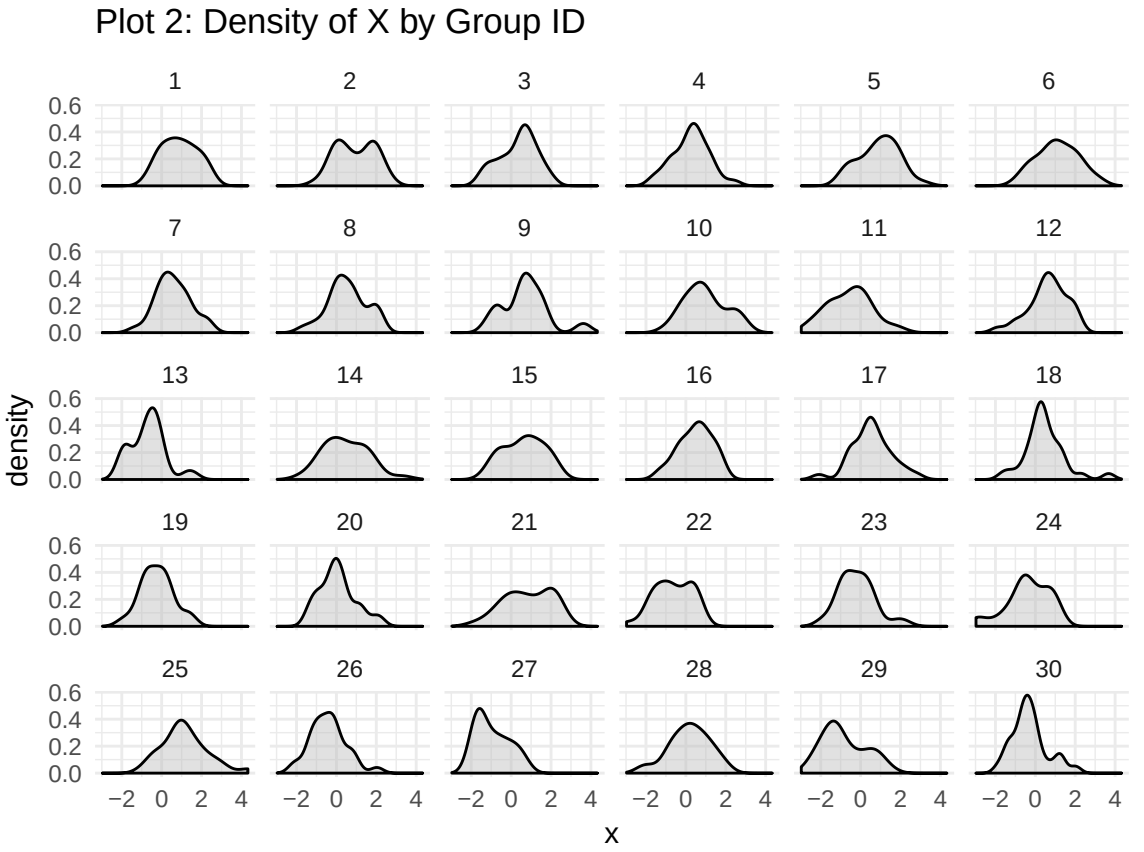
Data were generated to have high slope variances (random slopes), but centered to avoid high variances in the intercept (no random intercept). This was an effort to make parameter estimates more difficult which should encourage more divergent chains. The following code chunk (full version in apendix) generates data described above:

```
cor_thresh <- c(0,0.9)
index <- 1:N2
sig <- c()
for(i in index){
  x <- runif(n = 1,min = cor_thresh[[1]],max = cor_thresh[[2]])
  sig[[i]] <- matrix(c(1,x,
                       x,1),2)
}
```

Data were generated to have 30 groups, with 30 observations per group and no mean differences. These values were chosen to represent realistic values for a typical MLM. Plot 1 through 3 visually describes the multi-level structure of the data.

Plot 1: Raw Simulated Data by Group




Plot 2: Density of X by Group ID
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
density
0.0
0.2
0.4
0.6
−2
0
2
4
x

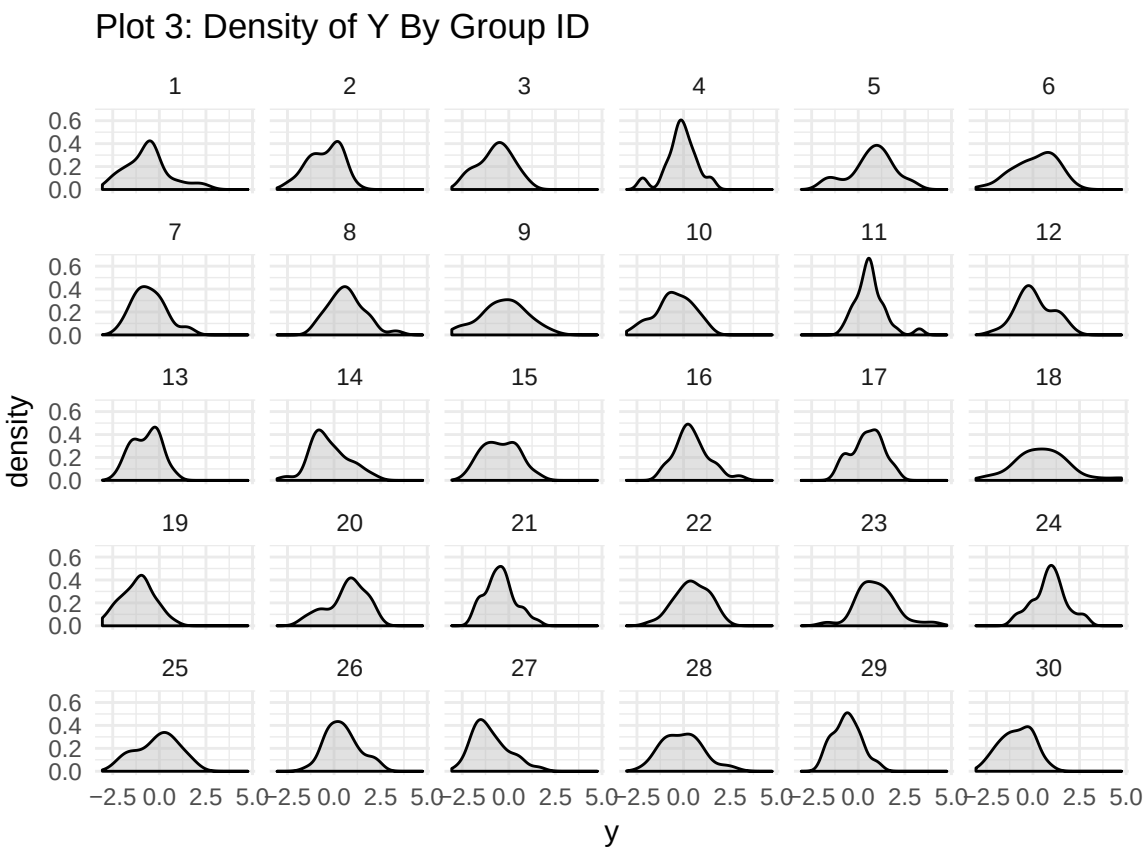

Plot 3: Density of Y By Group ID
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
density
0.0
0.2
0.4
0.6
−2.5
0.0
2.5
5.0
y

Table 1: Table 2: Head of Simulated Data

Y	X	Group ID	ID
-2.75	-0.4239	1	1
-0.2286	2.163	1	2
-0.4276	-0.07353	1	3
-1.893	0.5058	1	4
1.223	1.715	1	5
-1.534	0.899	1	6

MCMC Analysis

The MCMC analysis was conducted 3 times, under 3 different prior conditions: Diffuse, Realistic, and Strong & Incorrect. I choose a realistic prior specification to probe the effect for an average analysis. Diffuse was chosen for the same reason (for some reason some researchers find it meaningful to use diffuse priors). Strong and Incorrect was choosen in an effort to encourage divergent chains, displaying the effect for worst case scenarios. Table 2 summarizes the prior specifications for each condition. Each condtion was ran with 10 chains for 2,000 itterations, 50% was designated burn-in. The model for the analysis is displayed in formula 1 through 4, and is a typical MLM with random slopes.

$$L1: \quad y_{ij} = \beta_{0j} + \beta_{1j}x_{1ij} + \epsilon_{ij} \tag{1}$$

$$L2: \quad \beta_{0j} = \gamma_{00} + u_{0j} \tag{2}$$

$$L2: \quad \beta_{1j} = \gamma_{10} + u_{1j} \tag{3}$$

$$Overall: \quad y_{ij} = \gamma_{00} + u_{0j} + \gamma_{10}x_{1ij} + u_{1j}x_{1ij} + \epsilon_{ij} \tag{4}$$

Table 2: Prior Specifications by Condition

Prior	Values	Parameter	Coefficient
	Diffuse		
Uniform	-100, 100	β	X
LKJ	1	-	Group ID
Student-t	3, 0, 10	SD	Group ID
Uniform	0, 100	SD	ID / G ID
Uniform	-100, 100	Sigma	-
	Strong and Incorrect		
Gaussian	10, 0.1	β	X
LKJ	1	-	Group ID
Student-t	3, 0, 10	SD	-
Student-t	3, 0, 10	SD	ID / G ID
Student-t	3, 0, 10	Sigma	-
	Realistic		
Gaussian	0, 1	β	X
LKJ	1	-	Group ID
Student-t	3, 0, 10	SD	-
Student-t	3, 0, 10	SD	ID / G ID
Student-t	3, 0, 10	Sigma	-

Realistic

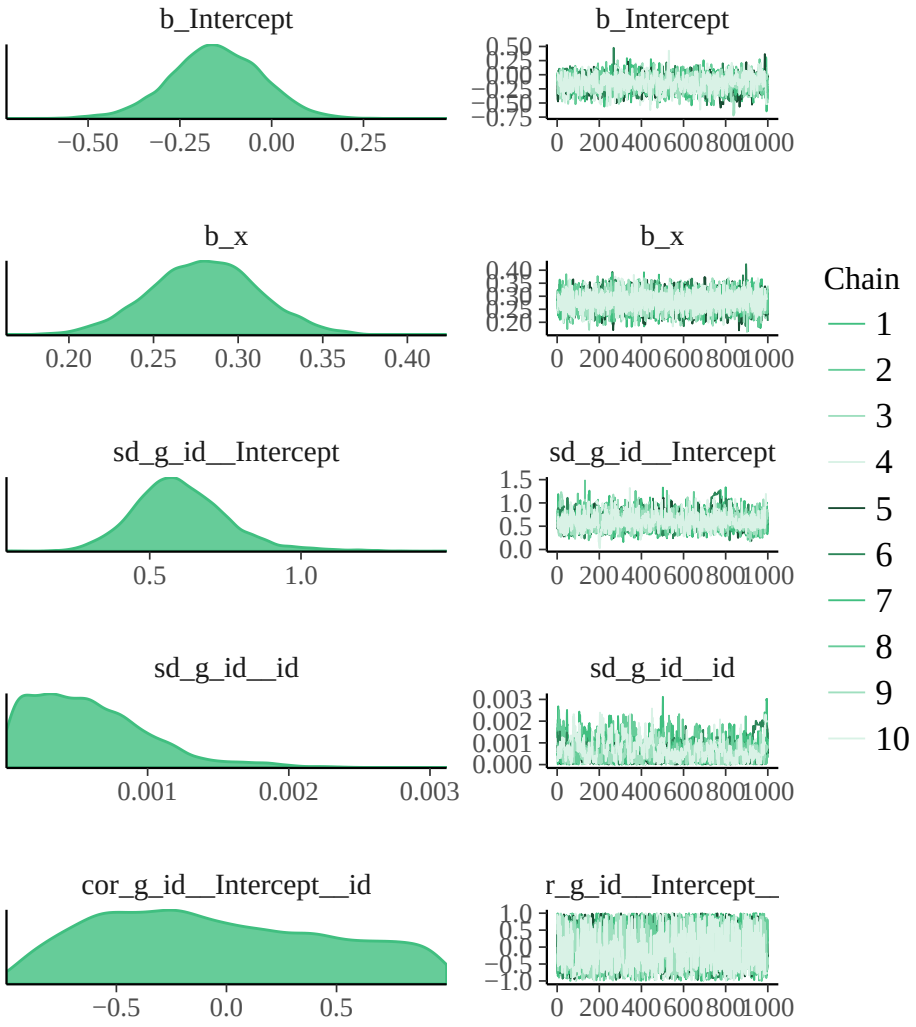


Figure 1: Realistic Estimates & Diagnostics by Parameter

Table 3: Realistic Fit Summary

Parameter	Estimate	Est. Erorr	Low 95 CI	High 95 CI	Eff. n	R-hat
B0	-0.151	0.1259	-0.404	0.09055	1989	1.001
B1	0.2803	0.03223	0.2169	0.3436	10000	1.002
SD B0	0.6043	0.1566	0.3295	0.9624	1193	1.012
SD ID	0.0005914	0.0004279	2.61e-05	0.001676	174.8	1.056
R Intercept & ID	-0.04156	0.5234	-0.8854	0.9262	1436	1.01

The realistic condition provided acceptable fit for the fixed effects: B0 and B1, with near perfect R-hat values. Further, the estimates for these effects are accurate when compared to the population values ($\beta_0^{pop} = 0, \beta_1^{pop} = 0.35$). The random effects also all show no signs of convergence problems with suitable R hat statistics well below the arbitrary cutoff of 1.1.

Diffuse

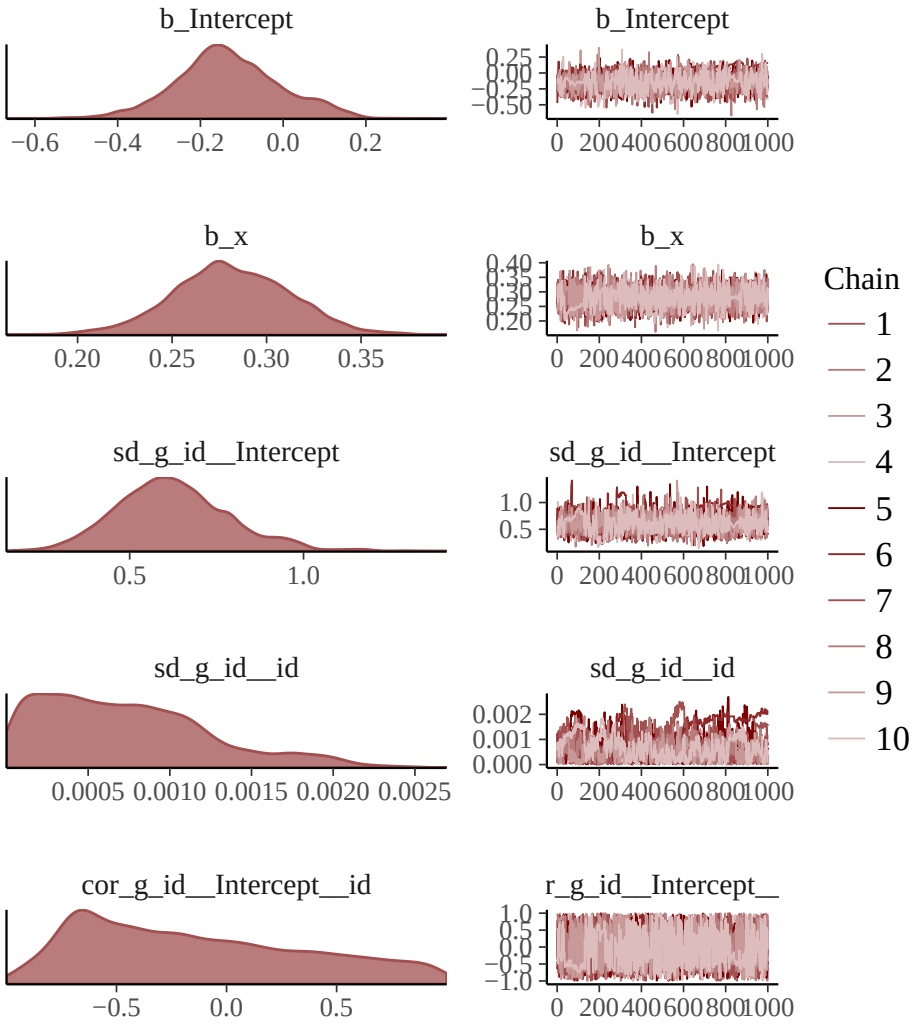


Figure 2: Diffuse Estimates & Diagnostics by Parameter

Table 4: Diffuse Fit Summary

Parameter	Estimate	Est. Erorr	Low 95 CI	High 95 CI	Eff. n	R-hat
B0	-0.132	0.126	-0.3816	0.1226	100.4	1.07
B1	0.2816	0.03127	0.2196	0.3426	873.3	1.014
SD B0	0.6255	0.1634	0.3339	0.9748	78.41	1.072
SD ID	0.0007336	0.0005126	3.451e-05	0.001937	63.36	1.134
R Intercept & ID	-0.1326	0.5133	-0.8809	0.9016	186.3	1.04

The diffuse condition also exhibits acceptable convergence for the fixed effects: B0 and B1, with near perfect R-hat values. Although, the r-hat values are more concerning compared to the realistic condition. The estimates for the fixed effects are accurate when compared to the population values ($\beta_0^{pop} = 0$,$\beta_1^{pop} = 0.35$). The random effects however suffer from the lack of prior information, this resulted in the r-hat value for the SD of ID parameter to be above the acceptable arbitrary threshold of 1.1.

Strong & Incorrect

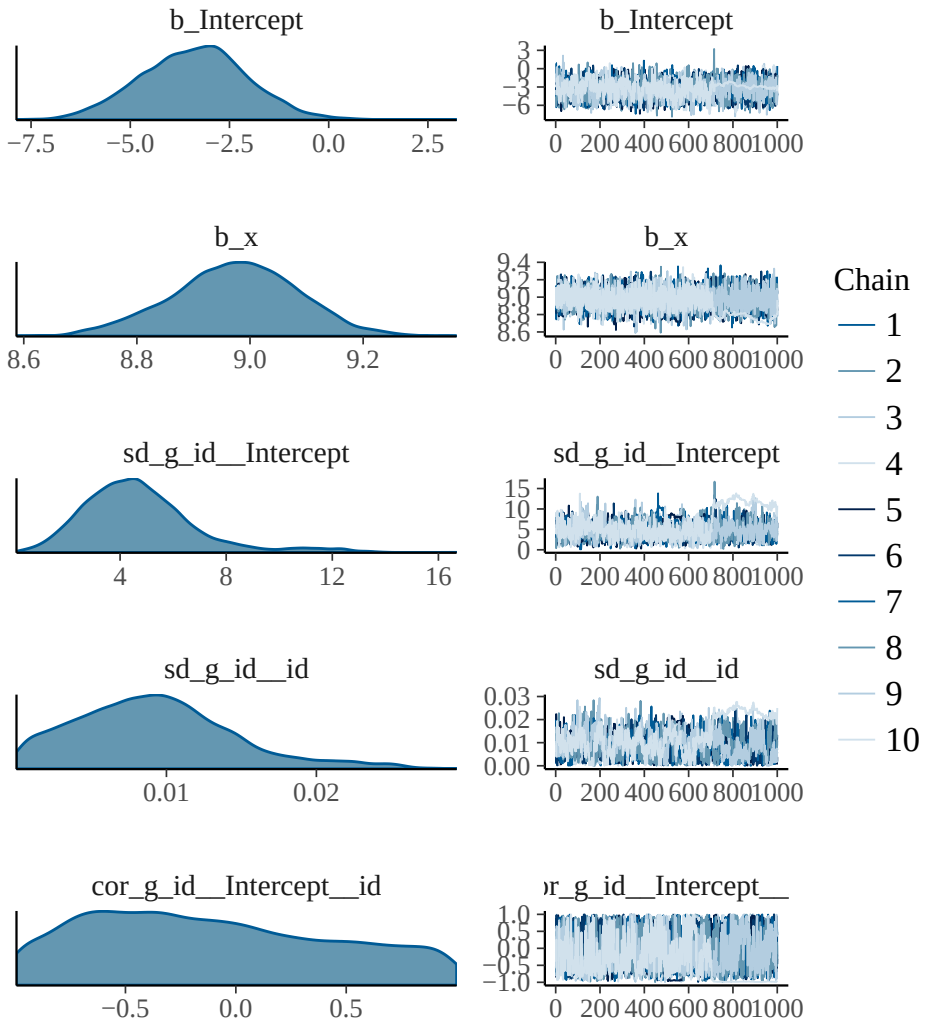


Figure 3: Strong & Incorrect Estimates & Diagnostics by Parameter

Table 5: Strong & Incorrect Fit Summary

Parameter	Estimate	Est. Erorr	Low 95 CI	High 95 CI	Eff. n	R-hat
B0	-3.388	1.332	-5.988	-0.8017	1519	1.007
B1	8.974	0.1104	8.751	9.184	243.6	1.033
SD B0	4.728	2.161	1.417	10.88	78.63	1.106
SD ID	0.009251	0.00527	0.000695	0.02214	98.25	1.091
R Intercept & ID	-0.1034	0.5481	-0.9793	0.9335	258.3	1.031

The Strong and Incorrect condition has even more convergence problems overall. However, the fixed effects are within acceptable r- hat ranges. It is notable that the estimates are extremely biased. The random effects would be considered problematic with r-hat values near 1.1 for the SD estimates for both B0, and ID.

Results

Two derived statistics will be used to evaluate convergence and bias. R-hat describes convergence, values close to 1.0 indicate high convergence between chains. 1.1 is an arbitrary threshold of convergence, 1.1 and greater is considerd a major violation. Bias was calculated with a raw distance formula and is the mean squared summed deviation from the population.

Convergence

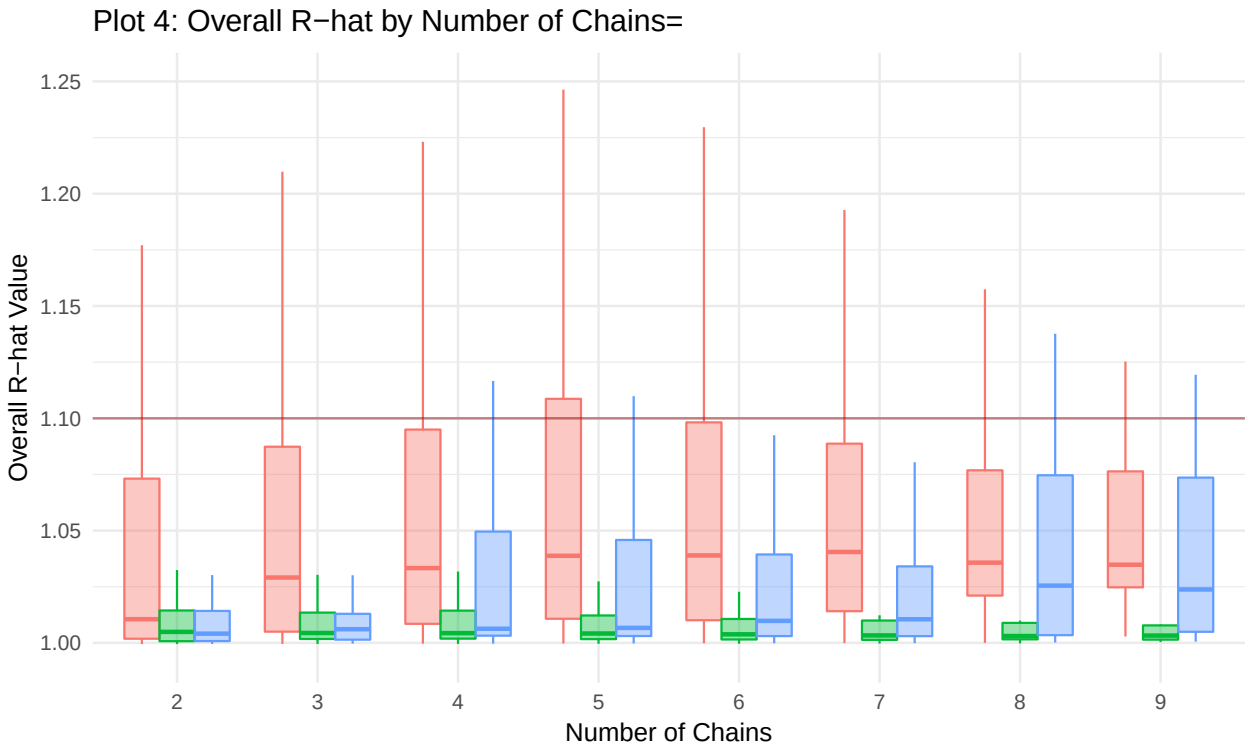
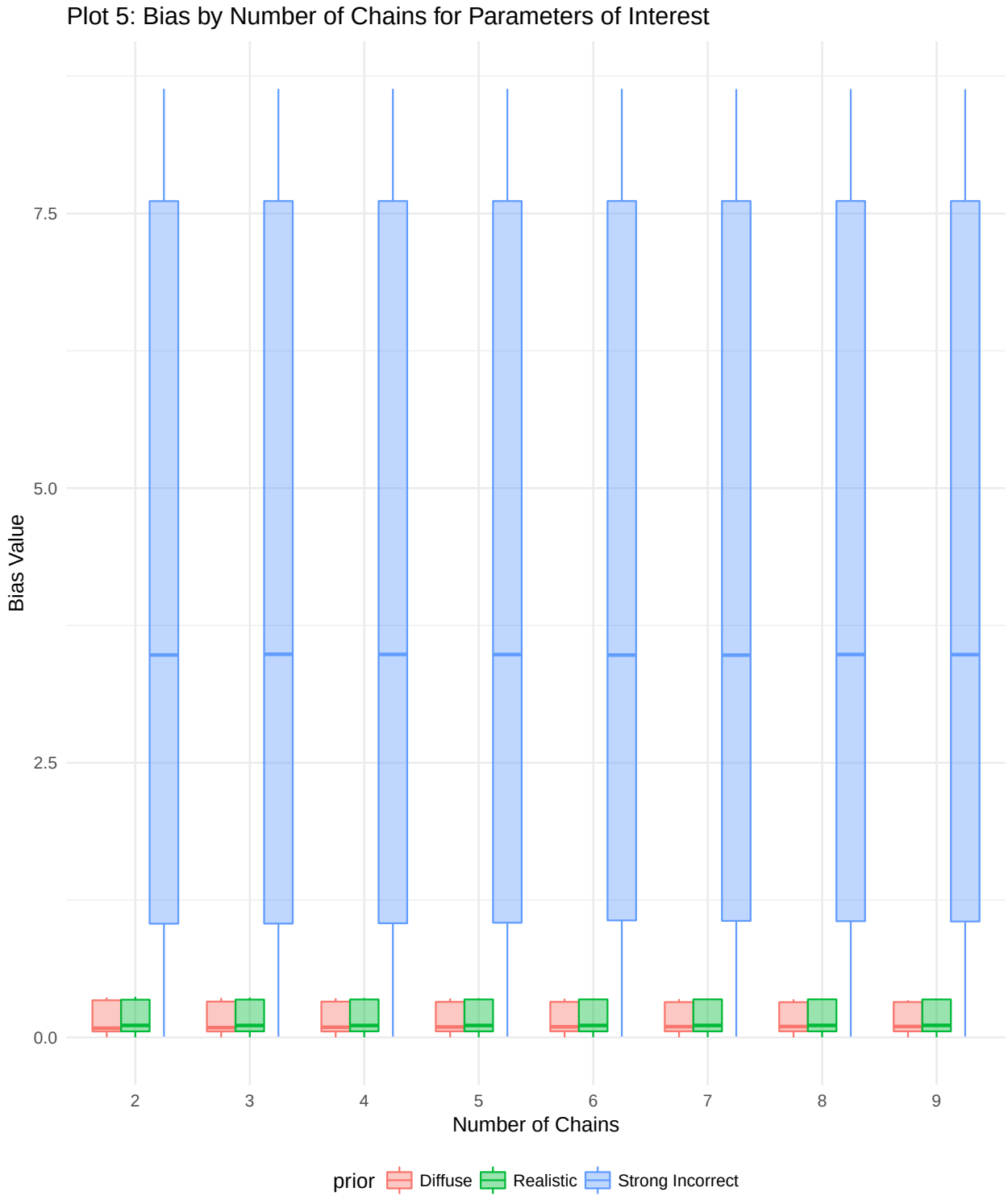


Table 4 displays the results for overall r-hat values by number of chains sampled. It is interesting to note that only the diffuse condition exhibits convergence issues at the low end (2 and 3 chains). This is interesting, the strong incorrect condition exhibits no convergence problems. However, near the high end (8, 9 chains) the strong and incorrect condition exhibits poorer fit, while the realistic condition only gets better.

This result is very explicit, higher chains provide more confidence in our results when priors are realistic or diffuse. However, we see worse convergence rates for the incorrectly specified priors (which is desirable). This provides more confidence in our results overall: if we use high chain numbers, and we reach poor convergence, we can be more confident that the model was misspecified (poor choice of priors).

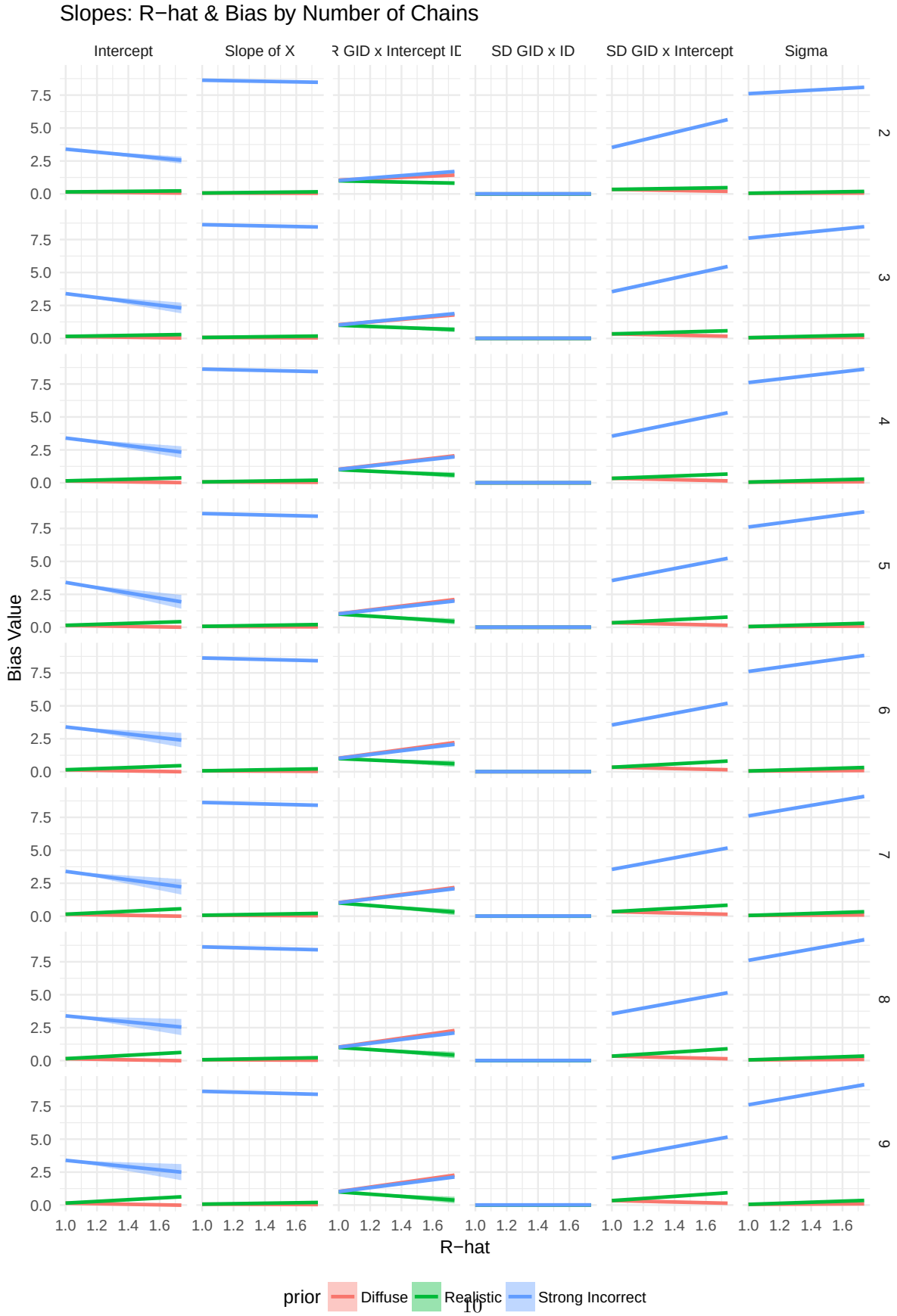
Bias

Bias was calculated as with a raw distance formula and is the mean squared summed deviation from the population. Higher values represent estimates that are farther from the population value. Bias values of 0 represnt perfect estimates.



Plot 5 provides a graphical depiction of the overall bias by chains sampled. The diffuse and realistic conditions provide comperable fit, with no substantial change over number of chains. Strong and incorrect however has a large range of bias, but also sees no difference over number of chains sampled. This is a desirable outcome. It apears that the number of chains has no effect on the accuracy of the estimat, only on the convergence of the model.

Putting it Together: Relationship Between Bias and Convergence



There is only a relationship between bias and convergence under specific conditions. One, the prior condition for strong and incorrect exhibits a relationship between bias and convergence. Two, only for specific paramater, specifically the Intercept, and random effect varainces. For the intercept the relationship is negative, indicating higher R-hat values (lower convergence) corresponds with *less* bias. This is counter intuitive at first, but I interpret this as some chains converging on the strong prior, while others converge on the data space. This would lead to this odd relationship where convergence is only reached when chains converge on the wrong solution. There is however a positive relationship for the random effects variances. It suggests that higher R-hat values (lower convergence) corresponds with more bias. This relationship makes sense, some chains are diverging on an incorrect decision.

Discusion and Recomendation

Under realistic conditions, convergence gets stronger with increased numbers of chains. Under strong and incorrect conditions, they get worse. Under diffuse conditions we expereince an odd, rainbow like effect, high convergence at low and high chain values, but poor convergence in the middle. Applied researchers sometimes are not aware of the accuracy of their priors. Further, some researchers explicitly use diffuse priors. Either of these situations can benefit from high numbers of chains. If convergence was acceptable at say, 10 chains, there is more confidence in the parameter estimates because we are more certain the priors used dont fall towards the incorrect catagory, dominating the data, and pulling chains in a divergent pattern. At low chain numbers the r-hat can be decieiving. Because of this, I urge researchers employing bayesian MCMC analysis to opt for higher chain numbers.

Future Directions

I can see an argument against this study revolving around higher itteration counts providing more convergence information. I plan to explore this further in future work. However, for the time being, one argument for more chains over high itteration counts is efficeincy. STAN allows users to deploy chains accross coomputing cores with ease. On a 4 core system, 4 chains can be ran simultaneously. Therefor, until further work confirms that a high chain number out performs high itteration counts, I contend researchers use the latter in the name of speed.

Apendix

Data Generation

```
set.seed(1856)

library(MASS)
library(brms)
library(lme4)
library(lmerTest)

##Data Gen

N2 <- 30 # Number of L2 Groups
N  <- 30  # N per group

cor_thresh <- c(0.0,.9)


# Makes 100, covariance matricies
# Random value for cov, by group
index <- 1:N2
sig <- c()

for(i in index){
  x <- runif(n = 1,min = cor_thresh[[1]],max = cor_thresh[[2]]) #correlation boundaries
  sig[[i]] <- matrix(c(1,x,
                       x,1),2)
}

# Generate continuos Data
dat1 <- c()
for(i in 1:length(sig)){
  dat1[[i]]<- MASS::mvrnorm(n = N,mu = runif(n = 2,min = -1,max = 1),Sigma = sig[[i]])
}

dat2 <- data.frame(do.call("rbind",dat1))

# Generate Catagorical Data
cat_dat <- c()

for(i in 1:N2){
  cat_dat[[i]] <- rep(i,N)
}

cat_dat2<- unlist(cat_dat)

#put it together
dat2$g <- cat_dat2

dat2$id <- seq(1:nrow(dat2))

colnames(dat2) <- c("y","x","g_id","id")
```

```r
head(dat2)
cor(dat2[,"x"],dat2[,"y"])

#Plot of data by G_id

#pdf(file = "plotty",height = 50,width = 6)
ggplot(data = dat2) +
  geom_point(aes(x = x, y = y)) +
  geom_smooth(aes(x = x, y = y),method = "lm",se = FALSE)+
  facet_grid(g_id~.)


#ICC
mod0<- lmer(formula = y ~ 1 + (1 | g_id), data = dat2)

summary(mod0)

covrand <- as.data.frame(VarCorr(mod0))
sigmau0 <- covrand[1,4]
sigmaeps <- covrand[2,4]

ICC <- sigmau0/(sigmau0+sigmaeps)

# Random intercept and slope

holder <- c()
chainz <- rep(2:10,10)

index <- 1:length(chainz)


form   <- bf(y ~ x + (1 + id | g_id))

#good priors
priors <- c(prior(normal(0,1),class = "b", coef = "x"),
            prior(lkj_corr_cholesky(1), class = "L"),
            prior(student_t(3,0,10), class = "sd", coef = "id", group = "g_id"),
            prior(student_t(3,0,10), class = "sigma"))

#diffuse priors
priors <- c(prior(uniform(-100,100),class = "b", coef = "x"),
            prior(lkj_corr_cholesky(1), class = "L"),
            prior(uniform(0,100), class = "sd", coef = "id", group = "g_id"),
            prior(uniform(0,100), class = "sigma"))

for(i in index){

fit <- brm(formula = form,
            data = dat2, prior = priors,family = gaussian(),
            chains = chainz[[i]],iter = 2000,warmup = 1000)

fit$chainz <- chainz[[i]]
```

```
holder[[i]] <- fit

}
```

Data Aggregation and Sampling

```
#Collector

library(asbio)
library(reshape2)

setwd("/home/z458r456/Documents/Simulation")

index <- list.files("data/fits/")
folder <- c()

for(i in 1:length(index)) {

folder[[i]]<- readRDS(paste0("data/fits/",index[i]))

}

fit_dif   <- folder[[3]][[9]]
fit_str_inc <- folder[[4]][[9]]
fit_str_cor <- folder[[5]][[9]]

rm(folder)

fit_dif_samples      <- fit_dif$fit@sim$samples
fit_str_inc_samples  <- fit_str_inc$fit@sim$samples
fit_str_cor_samples  <- fit_str_cor$fit@sim$samples

#constructing data frames for processing Rhats

index <- c('b_Intercept','b_x','sd_g_id__Intercept','sd_g_id__id','cor_g_id__Intercept__id',
      'sigma','r_g_id[1,Intercept]','r_g_id[2,Intercept]','r_g_id[3,Intercept]',
      'r_g_id[4,Intercept]','r_g_id[5,Intercept]','r_g_id[6,Intercept]',
      'r_g_id[7,Intercept]','r_g_id[8,Intercept]','r_g_id[9,Intercept]',
      'r_g_id[10,Intercept]','r_g_id[11,Intercept]','r_g_id[12,Intercept]',
      'r_g_id[13,Intercept]','r_g_id[14,Intercept]','r_g_id[15,Intercept]',
      'r_g_id[16,Intercept]','r_g_id[17,Intercept]','r_g_id[18,Intercept]',
      'r_g_id[19,Intercept]','r_g_id[20,Intercept]','r_g_id[21,Intercept]',
      'r_g_id[22,Intercept]','r_g_id[23,Intercept]','r_g_id[24,Intercept]',
      'r_g_id[25,Intercept]','r_g_id[26,Intercept]','r_g_id[27,Intercept]',
      'r_g_id[28,Intercept]','r_g_id[29,Intercept]','r_g_id[30,Intercept]',
      'r_g_id[1,id]','r_g_id[2,id]','r_g_id[3,id]','r_g_id[4,id]','r_g_id[5,id]',
      'r_g_id[6,id]','r_g_id[7,id]','r_g_id[8,id]','r_g_id[9,id]','r_g_id[10,id]',
      'r_g_id[11,id]','r_g_id[12,id]','r_g_id[13,id]','r_g_id[14,id]','r_g_id[15,id]',
      'r_g_id[16,id]','r_g_id[17,id]','r_g_id[18,id]','r_g_id[19,id]','r_g_id[20,id]',
      'r_g_id[21,id]','r_g_id[22,id]','r_g_id[23,id]','r_g_id[24,id]','r_g_id[25,id]',
      'r_g_id[26,id]','r_g_id[27,id]','r_g_id[28,id]','r_g_id[29,id]','r_g_id[30,id]',
      'lp__')
```

```
holder <- list()
#Makes list of each parameter estimate by chain.




dat_sets <-list( fit_dif_samples,
              fit_str_inc_samples,
              fit_str_cor_samples)

final_list <- list()


dat <- dat_sets[[3]]

for(j in index){# j = 'b_Intercept'

holder[[j]]   <- data.frame(dat[[1]][[j]],
                            dat[[2]][[j]],
                            dat[[3]][[j]],
                            dat[[4]][[j]],
                            dat[[5]][[j]],
                            dat[[6]][[j]],
                            dat[[7]][[j]],
                            dat[[8]][[j]],
                            dat[[9]][[j]],
                            dat[[10]][[j]])
  }

out_list <- list()
itter <- 100
hats<- data.frame(index)

for(k in 1:itter){#k = 1

  for(j in 2:10){# j = 2

  cols <- sample(x = 1:10,replace = FALSE,size = j)

    for(i in 1:length(index)){# i = 1

    pre_mat<- as.matrix(holder[[i]])
    sample_mat<- pre_mat[,c(cols)]
    hats[i,j] <- R.hat(M = sample_mat,burn.in = 0.5)
    }

  }

colnames(hats) <- c("parm","2_c","3_c","4_c","5_c","6_c","7_c","8_c","9_c","10_c")
out_list[[k]]  <- hats

}
```

```r
out_dat <- do.call("rbind",out_list)
out_parm <- list()

for(j in index){# j = 'b_Intercept'

id <- 1:length(out_dat[which(out_dat$parm== j ),])
out_parm[[j]]  <- data.frame(out_dat[which(out_dat$parm== j ),],id)

}

fin_list <- list()
for(j in 1:length(out_parm)){

t_dat <- melt(data = out_parm[[j]],id.vars = id,measure.vars =
            c("X2_c","X3_c","X4_c","X5_c","X6_c","X7_c","X8_c","X9_c","X10_c"),
            variable.name = "chains")
fin_list[[j]]<- t_dat[c("parm","chains","value")]

}

plot_dat<- do.call("rbind",fin_list)

folder[[5]][[1]]$prior
```