Lab7

Jun Rao

10/8/2019

# Hierarchical linear regression

In this lab we will test our understanding of random effects (mixed-effects) modeling.

#--------------------------------------  
#--------------------------------------  
# Import/Load the rstan library:  
library(tidyverse)  
library(rstan)  
library(loo)  
rstan\_options(auto\_write = TRUE)  
options(mc.cores = parallel::detectCores())  
#--------------------------------------  
#--------------------------------------  
## Simulating the data

## Simulating the data

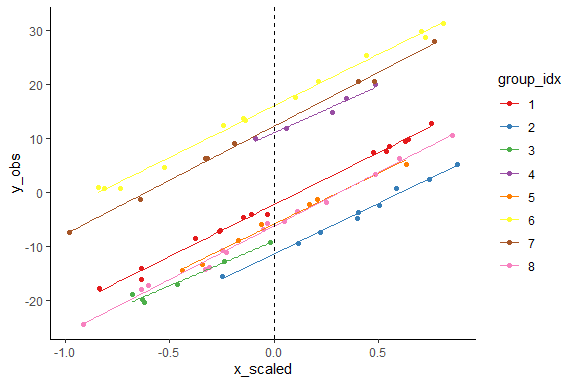
Our first task is to simulate data following the assumptions of random-effects modeling. First, we will assume a true model in which groups have an equal slope, but vary in their intercept:

# Task 1 (25 points)

*Complete the following:*

*1.Add comments to this code-chunk to convey your understanding (2 points).*

#creating simulations or random objects that can be reproduced.  
set.seed(7)  
  
#We have 8 groups  
n\_group = 8  
  
#how many data points in each group  
n\_samp\_per\_group = ceiling(runif(n\_group, 4, 15))  
  
group\_idx = rep(c(1:n\_group), times = n\_samp\_per\_group)  
  
#total data points in all groups  
n\_sample = sum(n\_samp\_per\_group)  
  
#Each Parameter in the model neea a prior  
alpha\_mean = rnorm(1, 0, 10)  
alpha\_sigma = abs(rnorm(1, 0, 10))  
alpha\_eta\_vec = rnorm(n\_group, 0, alpha\_sigma)  
   
beta = rnorm(1, 0, 10)  
  
sigma\_resid = abs(rnorm(1, 0, 2))  
  
#input value  
x\_raw = runif(n\_sample, 0, 3500)  
x\_mean = mean(x\_raw)  
x\_sd = sd(x\_raw)  
  
#center and standardize:  
x\_scaled = (x\_raw - x\_mean) / (2 \* x\_sd)  
  
epsilon = rnorm(n\_sample, 0, sigma\_resid)  
  
  
y\_hat = NULL  
for(i in 1:n\_sample){  
   
 y\_hat[i] = (alpha\_mean + alpha\_eta\_vec[group\_idx[i]]) + x\_scaled[i] \* beta  
   
}  
  
#Output Value  
y\_obs = y\_hat + epsilon  
  
  
data\_df =   
 data.frame(y\_obs, x\_scaled, group\_idx) %>%  
 mutate(group\_idx = factor(group\_idx))  
  
#draw the graphics for each group  
ggplot(data\_df, aes(x = x\_scaled, y = y\_obs, color = group\_idx)) +  
 geom\_point(shape = 19) +  
 geom\_smooth(method = "lm", se = FALSE, size = 0.2) +  
 scale\_color\_brewer(palette = "Set1") +  
 geom\_vline(xintercept = 0, linetype = 2) +  
 theme\_classic()



## Fitting the Stan models

*2.Use the supplied Stan code to fit the random effects model to your generated data.:*

*Remember to monitor your variable so that you can complete sub-task 5, below (5 points).*

stan\_data = list(n\_sample = n\_sample,  
 n\_group = n\_group,  
 y\_vec = y\_obs,  
 x\_vec = x\_scaled,  
 group\_idx = group\_idx)  
  
params\_monitor = c("beta", "alpha\_mean", "eta\_alpha", "alpha\_sigma", "sigma\_resid", "log\_lik")  
  
test\_fit\_1 = stan( file = "Intercepts\_LinReg.stan",  
 data = stan\_data,  
 pars = params\_monitor,  
 chains = 1,  
 iter = 10,  
 algorithm="NUTS")

##   
## SAMPLING FOR MODEL 'Intercepts\_LinReg' NOW (CHAIN 1).  
## Chain 1:   
## Chain 1: Gradient evaluation took 0 seconds  
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.  
## Chain 1: Adjust your expectations accordingly!  
## Chain 1:   
## Chain 1:   
## Chain 1: WARNING: No variance estimation is  
## Chain 1: performed for num\_warmup < 20  
## Chain 1:   
## Chain 1: Iteration: 1 / 10 [ 10%] (Warmup)  
## Chain 1: Iteration: 2 / 10 [ 20%] (Warmup)  
## Chain 1: Iteration: 3 / 10 [ 30%] (Warmup)  
## Chain 1: Iteration: 4 / 10 [ 40%] (Warmup)  
## Chain 1: Iteration: 5 / 10 [ 50%] (Warmup)  
## Chain 1: Iteration: 6 / 10 [ 60%] (Sampling)  
## Chain 1: Iteration: 7 / 10 [ 70%] (Sampling)  
## Chain 1: Iteration: 8 / 10 [ 80%] (Sampling)  
## Chain 1: Iteration: 9 / 10 [ 90%] (Sampling)  
## Chain 1: Iteration: 10 / 10 [100%] (Sampling)  
## Chain 1:   
## Chain 1: Elapsed Time: 0.001 seconds (Warm-up)  
## Chain 1: 0 seconds (Sampling)  
## Chain 1: 0.001 seconds (Total)  
## Chain 1:

## Now we will run our full model:  
# How many samples do we want of each parameter, from each chain?  
n\_mc\_samples = 1000  
# How much burn-in?  
n\_burn = 500  
# How much thinning? (take the ith value of the chain)  
n\_thin = 3  
# Total iterations needed:  
n\_iter\_total = (n\_mc\_samples \* n\_thin) + n\_burn  
model\_fit\_1 =  
stan(fit = test\_fit\_1, # So it knows we're already compiled  
 file = "Intercepts\_LinReg.stan",  
 data = stan\_data,  
 pars = params\_monitor,  
 chains = 3,  
 warmup = n\_burn,  
 thin = n\_thin,  
 iter = n\_iter\_total,  
 algorithm="NUTS")  
model\_out\_1 = rstan::extract(model\_fit\_1)  
str(model\_out\_1)

## List of 7  
## $ beta : num [1:3000(1d)] 19.7 19.4 19.4 19.3 19.4 ...  
## ..- attr(\*, "dimnames")=List of 1  
## .. ..$ iterations: NULL  
## $ alpha\_mean : num [1:3000(1d)] 0.329 6.789 3.795 -6.03 -0.922 ...  
## ..- attr(\*, "dimnames")=List of 1  
## .. ..$ iterations: NULL  
## $ eta\_alpha : num [1:3000, 1:8] -2.37 -8.9 -6.18 3.61 -1.59 ...  
## ..- attr(\*, "dimnames")=List of 2  
## .. ..$ iterations: NULL  
## .. ..$ : NULL  
## $ alpha\_sigma: num [1:3000(1d)] 14.3 8.43 9.38 12.24 8.07 ...  
## ..- attr(\*, "dimnames")=List of 1  
## .. ..$ iterations: NULL  
## $ sigma\_resid: num [1:3000(1d)] 0.727 0.761 0.792 0.95 0.837 ...  
## ..- attr(\*, "dimnames")=List of 1  
## .. ..$ iterations: NULL  
## $ log\_lik : num [1:3000, 1:78] -0.723 -0.678 -0.906 -0.98 -1.024 ...  
## ..- attr(\*, "dimnames")=List of 2  
## .. ..$ iterations: NULL  
## .. ..$ : NULL  
## $ lp\_\_ : num [1:3000(1d)] -50.1 -45.6 -47 -50.3 -45.5 ...  
## ..- attr(\*, "dimnames")=List of 1  
## .. ..$ iterations: NULL

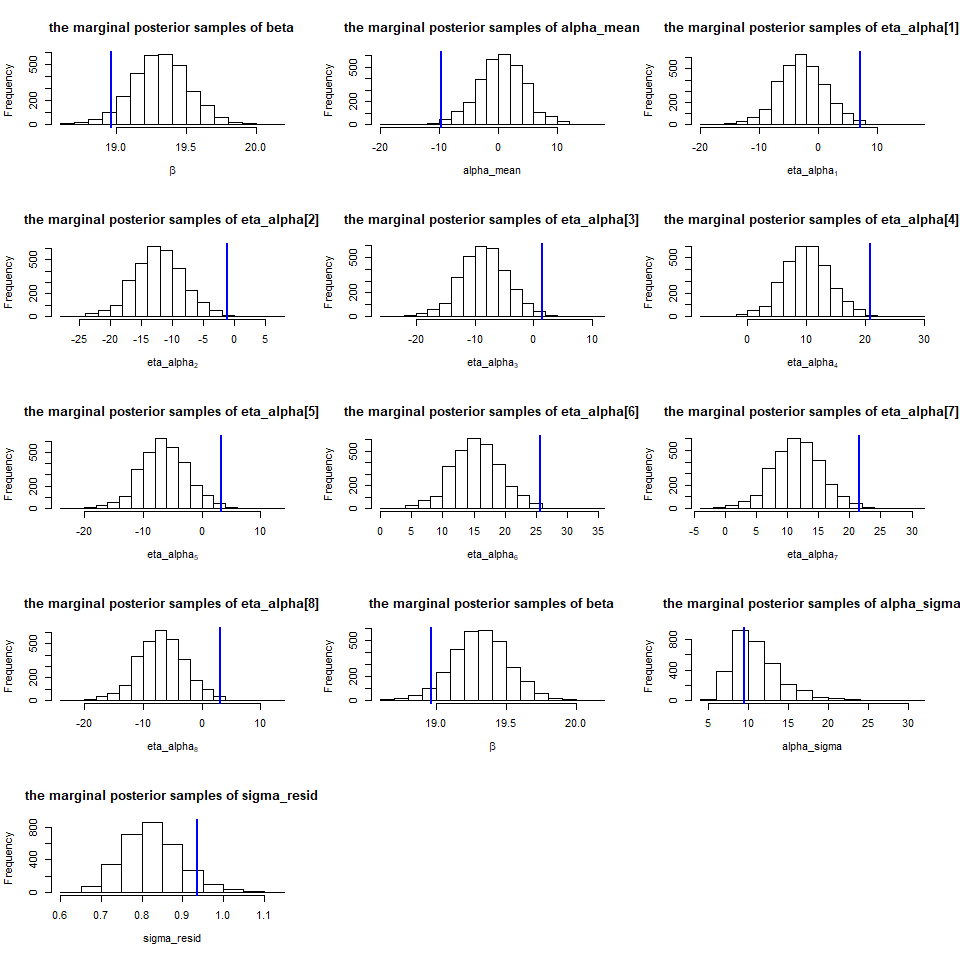
*3.Construct at least two visualizations to validate that your model was able to adequately recover the true parameters. Note: A table can be considered a visualization, because it organizes the quantitative information in a visually accessible manner (5 points).*

#first we visualize tabularly, the 95% CI for estimated parameters and compare these the the true values  
model\_sum = summary(model\_fit\_1)$summary  
credint = model\_sum[c(1:12),c(4,6,8)]  
credint2 = cbind(credint, c(beta, alpha\_mean, alpha\_sigma, alpha\_eta\_vec[1:8],sigma\_resid))  
colnames(credint2)[4] <- "true\_vals"   
  
# correct true eta values  
corrected = c(beta, alpha\_mean + mean(alpha\_eta\_vec), alpha\_sigma, alpha\_eta\_vec + alpha\_mean, sigma\_resid)  
credint2 = cbind(credint2, corrected)  
colnames(credint2)[5] <- "corrected"  
credint2

## 2.5% 50% 97.5% true\_vals corrected  
## beta 18.9236199 19.3179289 19.7174351 18.960671 18.9606707  
## alpha\_mean -7.5914333 0.5827238 8.5616500 -9.706733 0.5393192  
## eta\_alpha[1] -10.8982610 -2.9252298 5.2614329 9.472799 9.4727995  
## eta\_alpha[2] -20.2605034 -12.2285380 -4.1027645 7.086974 -2.6197595  
## eta\_alpha[3] -16.4345432 -8.4601092 -0.3469185 -1.107893 -10.8146268  
## eta\_alpha[4] 1.9911077 9.9272399 18.0476215 1.446095 -8.2606383  
## eta\_alpha[5] -14.4028674 -6.4648197 1.7754864 20.745223 11.0384900  
## eta\_alpha[6] 7.3278001 15.3823418 23.5730000 3.381659 -6.3250744  
## eta\_alpha[7] 3.5685271 11.5971335 19.7853115 25.735245 16.0285114  
## eta\_alpha[8] -14.9054012 -6.8510395 1.2482065 21.611737 11.9050031  
## alpha\_sigma 6.7007690 10.4672809 18.5512442 3.069382 -6.6373518  
## sigma\_resid 0.6984777 0.8212417 0.9790730 0.935361 0.9353610

# we can see that few parameters were adequately recoverd. This included alpha\_mean and all of the eta\_alphas (9 of 12 params)

par(mfrow = c(5,3))  
hist(model\_out\_1$beta,main="the marginal posterior samples of beta",xlab = expression(beta))  
abline(v = beta, lty = 1, lwd = 2, col = "blue")  
  
hist(model\_out\_1$alpha\_mean,main="the marginal posterior samples of alpha\_mean",xlab = expression(alpha\_mean))  
abline(v = alpha\_mean, lty = 1, lwd = 2, col = "blue")  
  
hist(model\_out\_1$eta\_alpha[,1],main="the marginal posterior samples of eta\_alpha[1]",xlab = expression(eta\_alpha[1]))  
abline(v = alpha\_eta\_vec[1], lty = 1, lwd = 2, col = "blue")  
  
hist(model\_out\_1$eta\_alpha[,2],main="the marginal posterior samples of eta\_alpha[2]",xlab = expression(eta\_alpha[2]))  
abline(v = alpha\_eta\_vec[2], lty = 1, lwd = 2, col = "blue")  
  
hist(model\_out\_1$eta\_alpha[,3],main="the marginal posterior samples of eta\_alpha[3]",xlab = expression(eta\_alpha[3]))  
abline(v = alpha\_eta\_vec[3], lty = 1, lwd = 2, col = "blue")  
  
hist(model\_out\_1$eta\_alpha[,4],main="the marginal posterior samples of eta\_alpha[4]",xlab = expression(eta\_alpha[4]))  
abline(v = alpha\_eta\_vec[4], lty = 1, lwd = 2, col = "blue")  
  
hist(model\_out\_1$eta\_alpha[,5],main="the marginal posterior samples of eta\_alpha[5]",xlab = expression(eta\_alpha[5]))  
abline(v = alpha\_eta\_vec[5], lty = 1, lwd = 2, col = "blue")  
  
hist(model\_out\_1$eta\_alpha[,6],main="the marginal posterior samples of eta\_alpha[6]",xlab = expression(eta\_alpha[6]))  
abline(v = alpha\_eta\_vec[6], lty = 1, lwd = 2, col = "blue")  
  
hist(model\_out\_1$eta\_alpha[,7],main="the marginal posterior samples of eta\_alpha[7]",xlab = expression(eta\_alpha[7]))  
abline(v = alpha\_eta\_vec[7], lty = 1, lwd = 2, col = "blue")  
  
hist(model\_out\_1$eta\_alpha[,8],main="the marginal posterior samples of eta\_alpha[8]",xlab = expression(eta\_alpha[8]))  
abline(v = alpha\_eta\_vec[8], lty = 1, lwd = 2, col = "blue")  
  
hist(model\_out\_1$beta,main="the marginal posterior samples of beta",xlab = expression(beta))  
abline(v = beta, lty = 1, lwd = 2, col = "blue")  
  
hist(model\_out\_1$alpha\_sigma,main="the marginal posterior samples of alpha\_sigma",xlab = expression(alpha\_sigma))  
abline(v = alpha\_sigma, lty = 1, lwd = 2, col = "blue")  
  
hist(model\_out\_1$sigma\_resid,main="the marginal posterior samples of sigma\_resid",xlab = expression(sigma\_resid))  
abline(v = sigma\_resid, lty = 1, lwd = 2, col = "blue")



*4.Construct a Stan code file that allows for both random intercepts and random slopes. Fit this new, more complex model to your same data set (8 points).*

beta\_mean = rnorm(1, 0, 10)  
beta\_sigma = abs(rnorm(1, 0, 10))  
beta\_eta\_vec = rnorm(n\_group, 0, alpha\_sigma)  
  
params\_monitor = c("beta\_mean", "alpha\_mean", "eta\_beta", "eta\_alpha", "beta\_sigma", "alpha\_sigma", "sigma\_resid", "log\_lik")  
  
test\_fit\_2 = stan( file = "Lab7.stan",  
 data = stan\_data,  
 pars = params\_monitor,  
 chains = 1,  
 iter = 10,  
 algorithm="NUTS")

##   
## SAMPLING FOR MODEL 'Lab7' NOW (CHAIN 1).  
## Chain 1:   
## Chain 1: Gradient evaluation took 0 seconds  
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.  
## Chain 1: Adjust your expectations accordingly!  
## Chain 1:   
## Chain 1:   
## Chain 1: WARNING: No variance estimation is  
## Chain 1: performed for num\_warmup < 20  
## Chain 1:   
## Chain 1: Iteration: 1 / 10 [ 10%] (Warmup)  
## Chain 1: Iteration: 2 / 10 [ 20%] (Warmup)  
## Chain 1: Iteration: 3 / 10 [ 30%] (Warmup)  
## Chain 1: Iteration: 4 / 10 [ 40%] (Warmup)  
## Chain 1: Iteration: 5 / 10 [ 50%] (Warmup)  
## Chain 1: Iteration: 6 / 10 [ 60%] (Sampling)  
## Chain 1: Iteration: 7 / 10 [ 70%] (Sampling)  
## Chain 1: Iteration: 8 / 10 [ 80%] (Sampling)  
## Chain 1: Iteration: 9 / 10 [ 90%] (Sampling)  
## Chain 1: Iteration: 10 / 10 [100%] (Sampling)  
## Chain 1:   
## Chain 1: Elapsed Time: 0.001 seconds (Warm-up)  
## Chain 1: 0.002 seconds (Sampling)  
## Chain 1: 0.003 seconds (Total)  
## Chain 1:

## Now we will run our full model:  
# How many samples do we want of each parameter, from each chain?  
n\_mc\_samples = 1000  
# How much burn-in?  
n\_burn = 500  
# How much thinning? (take the ith value of the chain)  
n\_thin = 3  
# Total iterations needed:  
n\_iter\_total = (n\_mc\_samples \* n\_thin) + n\_burn  
model\_fit\_2 =  
stan(fit = test\_fit\_2, # So it knows we're already compiled  
 file = "Intercepts\_LinReg.stan",  
 data = stan\_data,  
 pars = params\_monitor,  
 chains = 3,  
 warmup = n\_burn,  
 thin = n\_thin,  
 iter = n\_iter\_total,  
 algorithm="NUTS")  
model\_out\_2 = rstan::extract(model\_fit\_2)  
str(model\_out\_2)

## List of 9  
## $ beta\_mean : num [1:3000(1d)] 19.1 19.2 19.5 19.5 19 ...  
## ..- attr(\*, "dimnames")=List of 1  
## .. ..$ iterations: NULL  
## $ alpha\_mean : num [1:3000(1d)] -0.16 2.51 0.477 9.372 -0.17 ...  
## ..- attr(\*, "dimnames")=List of 1  
## .. ..$ iterations: NULL  
## $ eta\_beta : num [1:3000, 1:8] 0.114 -0.32 -0.286 -0.198 -0.056 ...  
## ..- attr(\*, "dimnames")=List of 2  
## .. ..$ iterations: NULL  
## .. ..$ : NULL  
## $ eta\_alpha : num [1:3000, 1:8] -1.91 -4.58 -2.35 -11.76 -2.51 ...  
## ..- attr(\*, "dimnames")=List of 2  
## .. ..$ iterations: NULL  
## .. ..$ : NULL  
## $ beta\_sigma : num [1:3000(1d)] 0.623 0.211 0.113 0.443 0.245 ...  
## ..- attr(\*, "dimnames")=List of 1  
## .. ..$ iterations: NULL  
## $ alpha\_sigma: num [1:3000(1d)] 9.82 6.73 9.3 10.84 16.73 ...  
## ..- attr(\*, "dimnames")=List of 1  
## .. ..$ iterations: NULL  
## $ sigma\_resid: num [1:3000(1d)] 0.746 0.778 0.821 0.676 0.881 ...  
## ..- attr(\*, "dimnames")=List of 1  
## .. ..$ iterations: NULL  
## $ log\_lik : num [1:3000, 1:78] -0.627 -0.682 -0.731 -0.76 -0.967 ...  
## ..- attr(\*, "dimnames")=List of 2  
## .. ..$ iterations: NULL  
## .. ..$ : NULL  
## $ lp\_\_ : num [1:3000(1d)] -48 -36.6 -38 -44.2 -48.1 ...  
## ..- attr(\*, "dimnames")=List of 1  
## .. ..$ iterations: NULL

#summary(model\_fit\_2)$summary  
#first we visualize tabularly, the 95% CI for estimated parameters and compare these the the true values  
model\_sum = summary(model\_fit\_2)$summary  
credint = model\_sum[c(1:21),c(4,6,8)]  
credint2 = cbind(credint, c(beta\_mean, alpha\_mean, beta\_sigma, alpha\_sigma, beta\_eta\_vec[1:8], alpha\_eta\_vec[1:8],sigma\_resid))  
colnames(credint2)[4] <- "true\_vals"   
  
# correct true eta values  
corrected = c(beta\_mean+mean(beta\_eta\_vec), alpha\_mean + mean(alpha\_eta\_vec), alpha\_sigma, beta\_eta\_vec + beta\_mean, alpha\_eta\_vec + alpha\_mean, sigma\_resid)  
credint2 = cbind(credint2, corrected)  
colnames(credint2)[5] <- "corrected"  
credint2

## 2.5% 50% 97.5% true\_vals corrected  
## beta\_mean 18.39846591 19.24987218 19.8150943 14.9048326 11.1025994  
## alpha\_mean -7.62067516 0.48512666 9.0095442 -9.7067334 0.5393192  
## eta\_beta[1] -0.67366840 0.02003857 0.9514246 2.2660105 9.4727995  
## eta\_beta[2] -1.15214690 -0.05167200 0.8601764 9.4727995 11.4673746  
## eta\_beta[3] -2.05418282 -0.16017574 0.5474489 -3.4374580 24.1947858  
## eta\_beta[4] -1.69526090 -0.06030707 0.8778490 9.2899531 12.9210723  
## eta\_beta[5] -1.03501377 -0.02993664 0.9084060 -1.9837603 -0.6105675  
## eta\_beta[6] -0.77816191 -0.02397415 0.8192952 -15.5154002 6.0191753  
## eta\_beta[7] -0.35750953 0.22614854 1.6054094 -8.8856573 9.3427149  
## eta\_beta[8] -0.36445860 0.20230848 1.4198086 -5.5621177 14.3337057  
## eta\_alpha[1] -11.29123067 -2.78622721 5.3183983 -0.5711269 11.1525341  
## eta\_alpha[2] -20.62970867 -12.03387012 -3.9357379 -3.7522985 -2.6197595  
## eta\_alpha[3] -17.05161298 -8.53412948 -0.5660163 7.0869739 -10.8146268  
## eta\_alpha[4] 1.79700773 10.08681020 18.1409364 -1.1078934 -8.2606383  
## eta\_alpha[5] -15.06070468 -6.40308856 1.6719936 1.4460951 11.0384900  
## eta\_alpha[6] 6.92352381 15.50354839 23.6114134 20.7452234 -6.3250744  
## eta\_alpha[7] 3.14886932 11.69229861 19.7627941 3.3816590 16.0285114  
## eta\_alpha[8] -15.26917912 -6.77913182 1.2036460 25.7352448 11.9050031  
## beta\_sigma 0.03351243 0.40973732 1.5694478 21.6117366 -6.6373518  
## alpha\_sigma 6.69277102 10.43207398 19.7490321 3.0693816 0.9353610  
## sigma\_resid 0.69040320 0.80683548 0.9595048 0.9353610 11.1025994

# we can see that few parameters were adequately recoverd. This included alpha\_mean and all of the eta\_alphas (9 of 12 params)

*5.Calculate the LOO-IC for each of your two models. Comment on which model is more parsimonious, given the data at hand (5 points).*

# Calculate the LOO and WAIC for a single model:  
# Full model  
log\_lik\_1 = extract\_log\_lik(model\_fit\_1)   
loo\_1 = loo(log\_lik\_1)  
loo\_1$estimates

## Estimate SE  
## elpd\_loo -100.136957 6.099760  
## p\_loo 9.529479 1.676108  
## looic 200.273913 12.199520

# reduced, intercept RE model  
log\_lik\_2 = extract\_log\_lik(model\_fit\_2)   
loo\_2 = loo(log\_lik\_2)  
loo\_2$estimates

## Estimate SE  
## elpd\_loo -100.7607 6.127462  
## p\_loo 11.8975 1.947453  
## looic 201.5213 12.254924

# model comparison of loo  
loo::compare(loo\_1, loo\_2)

## elpd\_diff se   
## -0.6 0.9

*Based on the p\_loo values for both models, the estimate is 9.5 which is lower for the reduced, intercept only random effects model. Additionally, the elpd\_loo is lowere but not significant because SE ranges overlap for both model estimates of elpd\_loo.*

*Therefore, we can say the reduced model is more parsimonious but because p\_loo is still greater than the number of parameters, there is a level of model misspecification remaining (other model architectures may improve performance).*

# Task 2 (15 points)

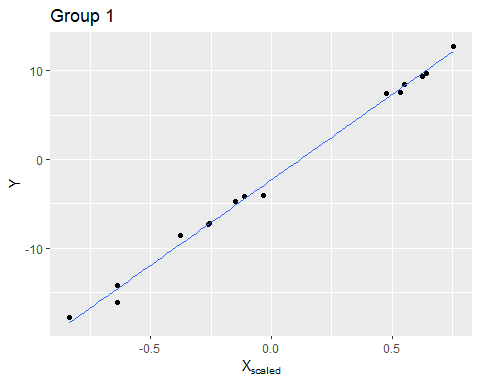
*1.Subset your data to include only one of the groups.*

# choosing Group 1 as the single group for this task  
data\_group1 = data\_df[data\_df$group\_idx == 1, ]

*2.For that specific group, create a scatterplot.*

*(For the following, use your most parsimonious model)*

ggplot(data\_group1, aes(x = x\_scaled, y = y\_obs)) +   
 geom\_point() +  
 labs(title = 'Group 1',  
 x = expression(X[scaled]),  
 y = expression(Y)) +  
 geom\_smooth(method = "lm", se = FALSE, size = 0.2)

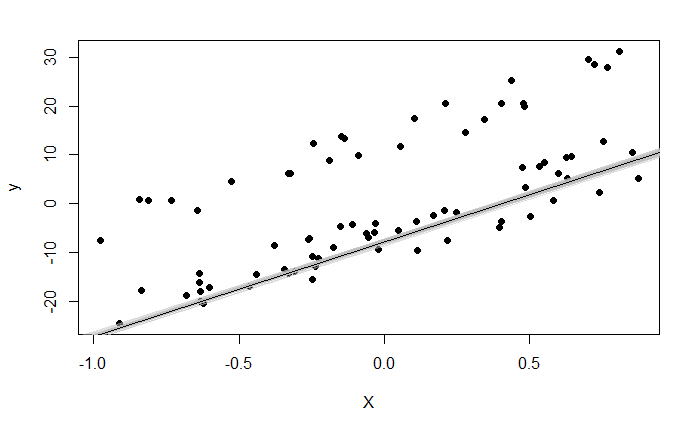


*3.On the scatterplot, overlay the median model-fit (i.e., the median estimated line).*

group\_idx\_rd = sample(c(1:n\_group),1)  
data\_df\_sub = data\_df[group\_idx == group\_idx\_rd,]  
median\_set = summary(model\_fit\_1)$summary[1:12, '50%']  
  
beta\_median = median\_set[1]  
alpha\_mean\_median = median\_set[2]  
eta\_alpha\_media = median\_set[2 + group\_idx\_rd]

*4.Bootstrap from the posterior to overlay many more possible model-fits. Be sure that for each of these model-fits, you are drawing parameter sets from your MCMC samples.: To get a joint draw, first specify which sample you will take from your MCMC chains. For instance:*

plot(data\_df[,2],data\_df[,1],pch=16,xlab="X",ylab="y")  
for (i in 1:50) {  
 temp\_idx = sample(c(1:n\_iter\_total-n\_burn), 1)  
 temp\_alpha\_mean = model\_out\_1$alpha\_mean[temp\_idx]  
 temp\_eta\_alph = model\_out\_1$eta\_alpha[temp\_idx,group\_idx\_rd]  
 temp\_beta = model\_out\_1$beta[temp\_idx]  
 abline(a = temp\_eta\_alph + temp\_alpha\_mean, b = temp\_beta, col = scales::alpha("gray",alpha=0.5))   
}  
  
abline(a = alpha\_mean\_median + eta\_alpha\_media, b= beta\_median,col="black")



*5.Make sure it is easy to visually distinguish between the median model-fit and the other possible model-fits. (You might want to plot the ``other’’ fits before overlaying the median fit).*