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:

$$y_i = (\overline{\alpha} + \eta_{j,[i]}) + \beta x_i$$

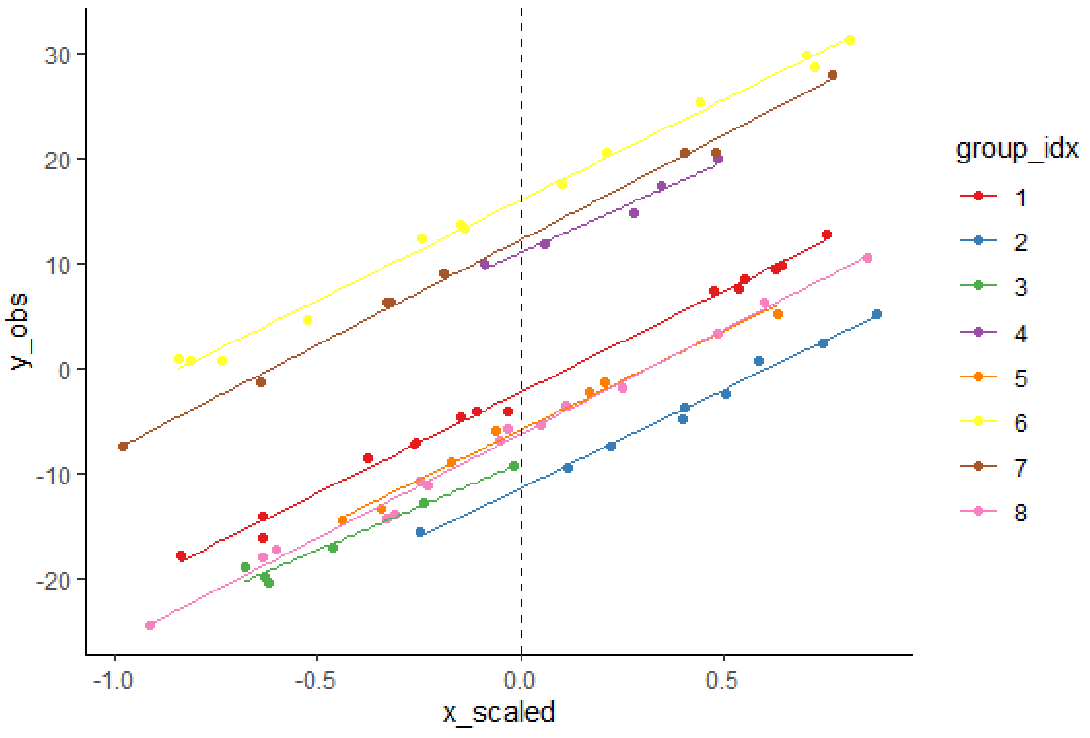
Task 1 (25 points)

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
```

```r
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).

```r
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_r
esid", "log_lik")
```

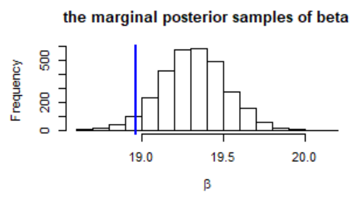
```
test_fit_1 = stan( file = "Intercepts_LinReg.stan",
                 data = stan_data,
                 pars = params_monitor,
                 chains = 1,
                 iter = 10,
                 algorithm="NUTS")

##
## 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 tak
e 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)
```

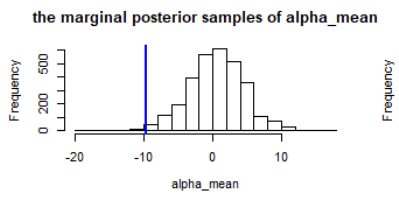
```
## List of 7
##  $ beta       : num [1:3000(1d)] 19.7 19.4 19.4 19.3 19.4 ...
##   ..- attr(*, "dimnames")=List of 1
##   .. ..$ iterations: NULL
##  $ alpha_mean : num [1:3000(1d)] 0.329 6.789 3.795 -6.03 -0.922 ...
##   ..- attr(*, "dimnames")=List of 1
##   .. ..$ iterations: NULL
##  $ eta_alpha  : 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
```

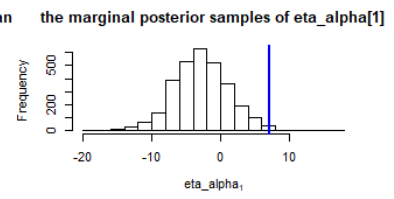
3.Construct at least two visualizations to validate that your model was able to adequately recover the true parameters. Note: A table can be considered a visualization, because it organizes the quantitative information in a visually accessible manner (5 points).

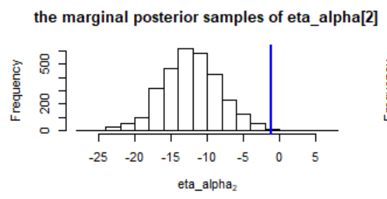
```
#first we visualize tabularly, the 95% CI for estimated parameters and compar
e these the the true values
model_sum = summary(model_fit_1)$summary
credint = model_sum[c(1:12),c(4,6,8)]
credint2 = cbind(credint, c(beta, alpha_mean, alpha_sigma, alpha_eta_vec[1:8]
,sigma_resid))
colnames(credint2)[4] <- "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)
```

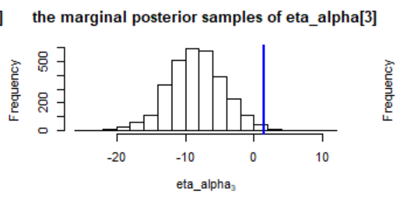
```
colnames(credint2)[5] <- "corrected"
credint2

##                     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 alph
a_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 = ex
pression(beta))
abline(v = beta, lty = 1, lwd = 2, col = "blue")

hist(model_out_1$alpha_mean,main="the marginal posterior samples of alpha_mea
n",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_al
pha[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_al
pha[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_al
pha[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_al
pha[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_al
pha[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_al
pha[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_al
pha[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_al
pha[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 = ex
pression(beta))
abline(v = beta, lty = 1, lwd = 2, col = "blue")

hist(model_out_1$alpha_sigma,main="the marginal posterior samples of alpha_si
gma",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_re
sid",xlab = expression(sigma_resid))
abline(v = sigma_resid, lty = 1, lwd = 2, col = "blue")
```

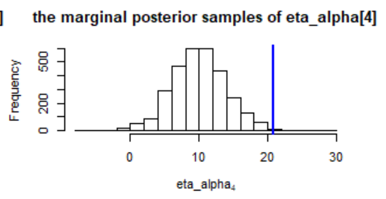


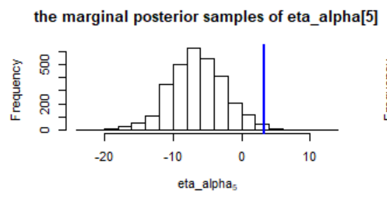


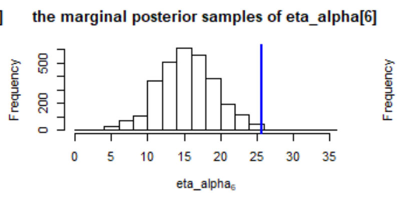


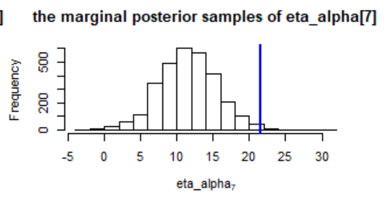


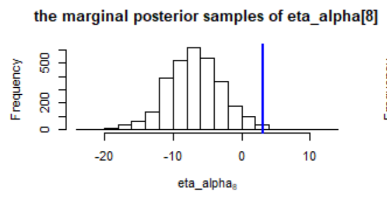


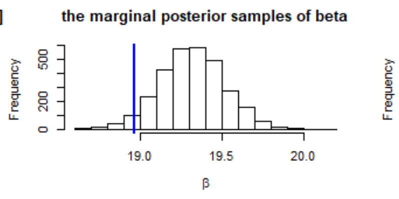


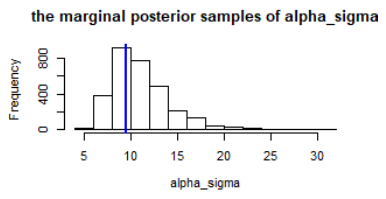


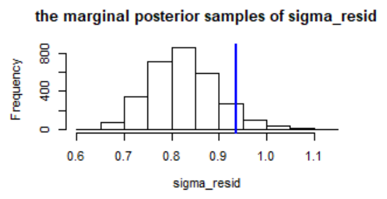












4.Construct a Stan code file that allows for both random intercepts and random slopes. Fit this new, more complex model to your same data set (8 points).

```
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 tak
e 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:
```

```r
## 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)$summary
#first we visualize tabularly, the 95% CI for estimated parameters and compar
e these the the true values
model_sum = summary(model_fit_2)$summary
credint = model_sum[c(1:21),c(4,6,8)]
credint2 = cbind(credint, c(beta_mean, alpha_mean, beta_sigma, alpha_sigma, b
eta_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_resi
d)
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 alph
a_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)

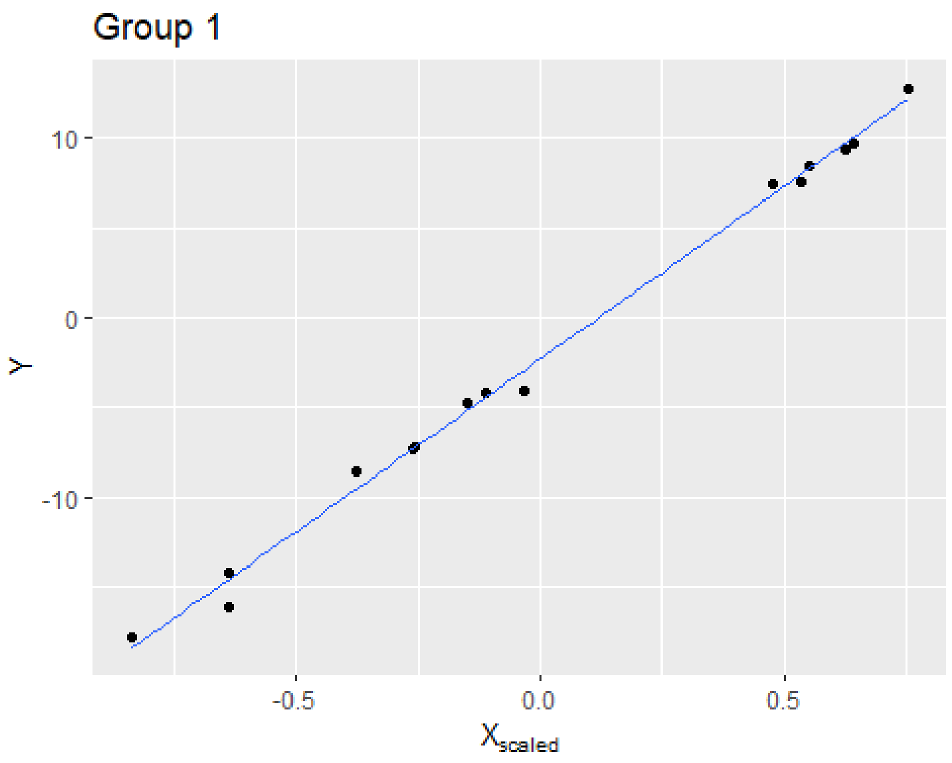
1.Subset your data to include only one of the groups.

```r
# 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)

```r
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)
```

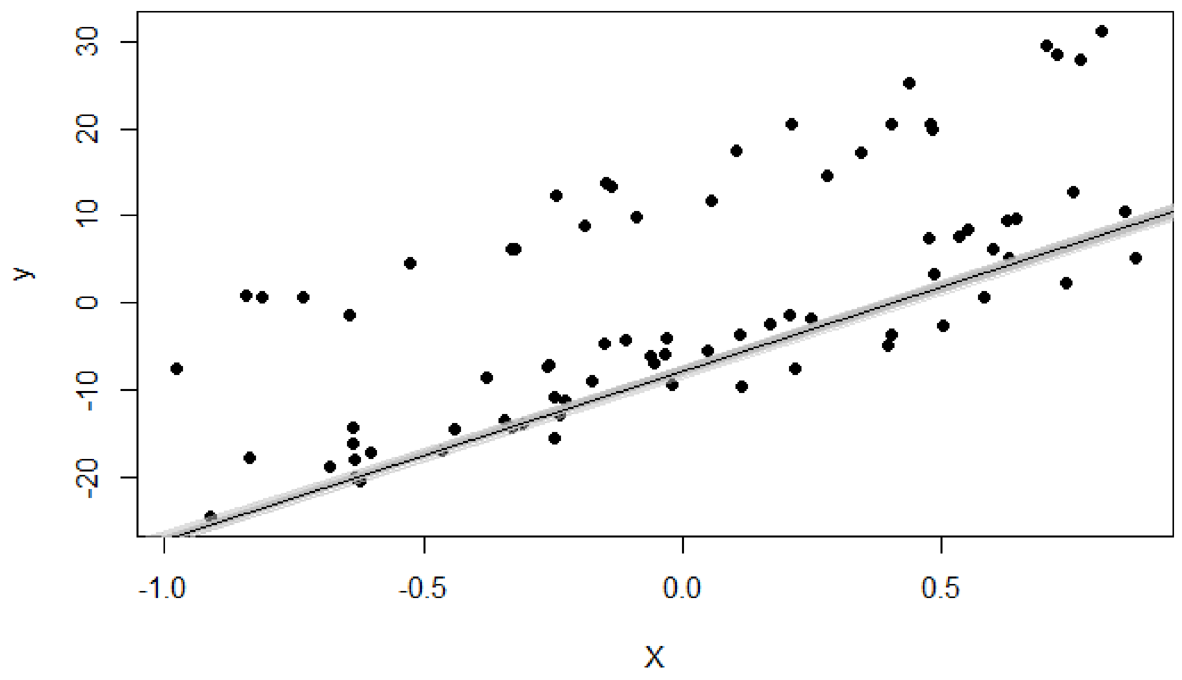


3.On the scatterplot, overlay the median model-fit (i.e., the median estimated line).

```r
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::a
lpha("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).