Final

Garrett Carr

April 14, 2022

This is a take-home exam. The final will be due at midnight on Tuesday, April 19. When you finish, email me your exam with the subject 'Stat 451 Final'. Please do your work in a .Rmd file so that I can see your code. Please hand in both your .Rmd file and either an .html file, a .pdf file, or a .doc file. Please name your files with your last name first, so my files would be named: 'FellinghamFinal.Rmd' and 'FellinghamFinal.pdf', 'FellinghamFinal.html' or 'FellinghamFinal.doc'.

As always, I expect your final to be your own work. You may use any notes and any help files available for the programs, but searching for similar code on the web would not be appropriate. You should not ask for help from any person currently alive in the mortal state.

The first question in worth 75 points and should be the focus until you believe you have completed it. The second question is worth 25 points.

```
knitr::opts_chunk$set(fig.align = 'center')
library(rstan)
library(cmdstanr)
library(tidyverse)
library(GGally)
library(bayesplot)
```

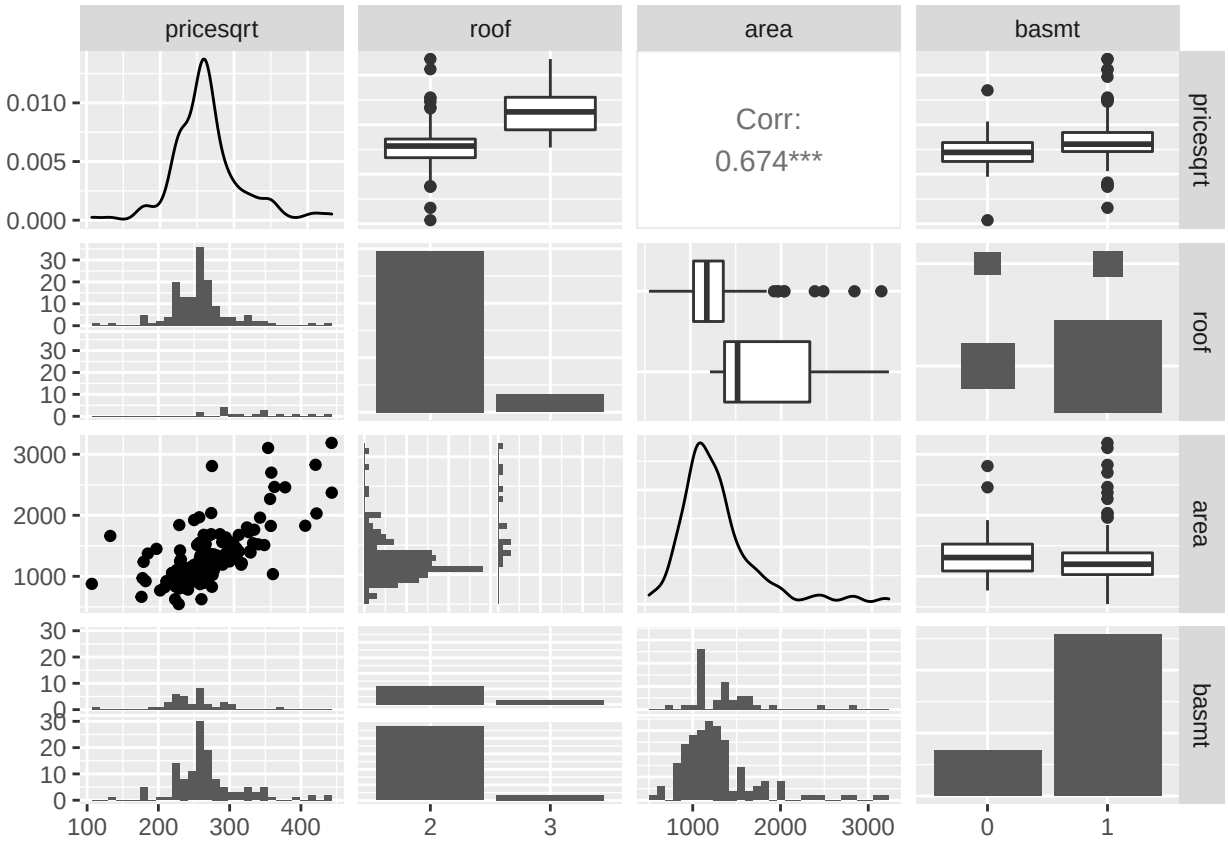
```
## Warning: package 'bayesplot' was built under R version 4.1.3
```

```
iron <- read_table('ironco.txt', skip = 1,
                   col_names = c('obs', 'price', 'lot', 'floors', 'const', 'roof', 'build', 'area', 'yr

# CLEAN-UP
iron$obs <- str_trim(iron$obs) |> str_replace_all('"', "")
iron$basmt <- as_factor(iron$basmt)
iron$roof <- as_factor(iron$roof)

iron
```

```
## # A tibble: 164 x 13
##    obs   price   lot floors const roof  build  area yr.built eff.age baths   gar
##    <chr> <dbl> <dbl>  <dbl> <dbl> <fct> <dbl> <dbl>    <dbl>   <dbl> <dbl> <dbl>
##  1 1     64500  0.17    1       3 2         4  1300     1992       3     2     0
##  2 2     72400  0.26    1       3 2         3  1300     1978      16     2     0
##  3 3     68500  0.2     1.5     3 2         3  1056     1978      15     1     0
##  4 4     71000  0.17    1       3 2         3  1050     1977      16     2     0
##  5 5     67500  0.17    1       3 2         3   994     1981      14     1     1
##  6 6     51000  0.25    1       3 2         3  1841     1978      13     3     1
##  7 7     69000  0.32    1       3 2         3  1100     1977      13     2     0
```

```
##  8 8     55500  0.16    1       3 2         3  1032     1974       14      1      0
##  9 9     60500  0.24    1       3 2         3  1188     1971       12      2      0
## 10 10    55000  0.17    1       3 2         3  1040     1977       15      2      0
## # ... with 154 more rows, and 1 more variable: basmt <fct>
```

```
data_list <- list(N = length(iron$obs), price = iron$price,
                  area = iron$area, roof = iron$roof, basmt = iron$basmt)

# EDA
iron |>
  mutate(pricesqrt = sqrt(price)) |>
  select(pricesqrt, roof, area, basmt) |>
  ggpairs()
```



1. On an annual basis, each county Assessor is required by Utah law to list and value on an assessment roll all property subject to *ad valorem* taxation. Iron County is located in southwest Utah approximately 265 miles south of Salt Lake City, UT and 170 miles north of Las Vegas, NV on the I-15 corridor. The Iron County Assessor's office assesses values on approximately 35,000 parcels of property on approximately 620,000 acres.

The data file *ironco.txt*, will be mailed to you, contains data on selling price for various properties, as well as information of covariates that may be related to selling price. The columns are described below:

- obs - observation number (in quotes)
- price - selling price of the property

- lot - lot acreage
- floors - number of floors (not including basement)
- const - assessed construction quality on a scale of 1 (poor) to 4 (excellent)
- roof - assessed roof condition on a scale of 1 (poor) to 4 (excellent)
- build - assessed home condition on a scale of 1 (poor) to 4 (excellent)
- area - square footage of home
- yr.built - year the home was built
- eff.age - evaluation by the assessor of the home's equivalent market age
- baths - number of full bathrooms
- gar - indicator for presence of a garage
- basmt - indicator of presence of a basement

Make the response the square root of price and write a model to estimate the response based on an intercept, the area of the home, the assessed roof condition, and the indicator for the presence of a basement.

The likelihood is normal.
Make your priors for the β's fairly flat and use a gamma prior for the error variance with parameters shape=3 and rate=.1.

Run the model in JAGS or Stan.

Make sure the chains have converged using the methods we have covered in class.

Show the 95% equal tail interval estimates for β_{area}.

Plot the posterior density of β_{area}.

```
mod <- cmdstan_model('mod1.stan')
mod$print()
```

```
## data {
##  int N;
##   array[N] int roof;
##   array[N] int basmt;
##   array[N] real area;
##   array[N] real price;
## }
##
## transformed data {
##   array[N] real pricesqrt;
##   for (i in 1:N) {
##     pricesqrt[i] = sqrt(price[i]);
##   }
## }
##
## parameters {
## // Betas
##  real beta0;
##   real barea;
##   real broof;
##   real bbasmt;
## // Deviation
##  real <lower=0> serr;
## }
##
## transformed parameters {
```

```
##  real s2err;
##  s2err = serr^2;
##   array[N] real mu;
##   for (i in 1:N) {
##    mu[i] = beta0 + barea*area[i] + broof*roof[i] + bbasmt*basmt[i];
##   }
## }
##
## model {
##   beta0 ~ normal(0, 10);
##   barea ~ normal(0, 10);
##   broof ~ normal(0, 10);
##   bbasmt ~ normal(0, 10);
##  s2err ~ gamma(3,0.1);
##   for (i in 1:N) {
##     pricesqrt[i] ~ normal(mu[i], serr);
##   }
## }
```

```
fit <- mod$sample(
  data = data_list,
  seed = 324,
  chains = 6,
  parallel_chains = 3,
  refresh = 1000,
  sig_figs = 5,
  iter_sampling = 3000
)
```

```
## Running MCMC with 6 chains, at most 3 in parallel...
##
## Chain 3 Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 1 Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 2 Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 2 Iteration: 1000 / 4000 [ 25%]  (Warmup)
## Chain 2 Iteration: 1001 / 4000 [ 25%]  (Sampling)
## Chain 3 Iteration: 1000 / 4000 [ 25%]  (Warmup)
## Chain 1 Iteration: 1000 / 4000 [ 25%]  (Warmup)
## Chain 3 Iteration: 1001 / 4000 [ 25%]  (Sampling)
## Chain 1 Iteration: 1001 / 4000 [ 25%]  (Sampling)
## Chain 2 Iteration: 2000 / 4000 [ 50%]  (Sampling)
## Chain 3 Iteration: 2000 / 4000 [ 50%]  (Sampling)
## Chain 1 Iteration: 2000 / 4000 [ 50%]  (Sampling)
## Chain 2 Iteration: 3000 / 4000 [ 75%]  (Sampling)
## Chain 3 Iteration: 3000 / 4000 [ 75%]  (Sampling)
## Chain 1 Iteration: 3000 / 4000 [ 75%]  (Sampling)
## Chain 2 Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 2 finished in 4.0 seconds.
## Chain 3 Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 3 finished in 4.3 seconds.
## Chain 4 Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 1 Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 1 finished in 4.6 seconds.
## Chain 5 Iteration:    1 / 4000 [  0%]  (Warmup)
```

```
## Chain 6 Iteration:    1 / 4000 [  0%]  (Warmup)
## Chain 6 Iteration: 1000 / 4000 [ 25%]  (Warmup)
## Chain 6 Iteration: 1001 / 4000 [ 25%]  (Sampling)
## Chain 5 Iteration: 1000 / 4000 [ 25%]  (Warmup)
## Chain 5 Iteration: 1001 / 4000 [ 25%]  (Sampling)
## Chain 4 Iteration: 1000 / 4000 [ 25%]  (Warmup)
## Chain 4 Iteration: 1001 / 4000 [ 25%]  (Sampling)
## Chain 6 Iteration: 2000 / 4000 [ 50%]  (Sampling)
## Chain 5 Iteration: 2000 / 4000 [ 50%]  (Sampling)
## Chain 4 Iteration: 2000 / 4000 [ 50%]  (Sampling)
## Chain 6 Iteration: 3000 / 4000 [ 75%]  (Sampling)
## Chain 5 Iteration: 3000 / 4000 [ 75%]  (Sampling)
## Chain 4 Iteration: 3000 / 4000 [ 75%]  (Sampling)
## Chain 6 Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 6 finished in 4.4 seconds.
## Chain 5 Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 5 finished in 5.3 seconds.
## Chain 4 Iteration: 4000 / 4000 [100%]  (Sampling)
## Chain 4 finished in 5.8 seconds.
##
## All 6 chains finished successfully.
## Mean chain execution time: 4.7 seconds.
## Total execution time: 11.0 seconds.
```

```
fit$summary(variables = c('serr', 'beta0', 'barea', 'broof', 'bbasmt', 'lp__'))
```

```
## # A tibble: 6 x 10
##   variable      mean    median      sd     mad        q5      q95  rhat ess_bulk
##   <chr>        <dbl>     <dbl>   <dbl>   <dbl>     <dbl>    <dbl> <dbl>    <dbl>
## 1 serr       24.9      24.9    0.916   0.918     23.4     2.64e+1  1.00   13684.
## 2 beta0      45.4      45.3    7.37    7.34      33.3     5.75e+1  1.00   11553.
## 3 barea       0.0755    0.0754 0.00449 0.00446    0.0681  8.28e-2  1.00   12936.
## 4 broof      41.5      41.5    5.45    5.43      32.6     5.06e+1  1.00   12805.
## 5 bbasmt     41.0      40.9    3.51    3.52      35.2     4.68e+1  1.00   12110.
## 6 lp__     -742.     -742.     1.59    1.41    -745.     -7.40e+2  1.00    7516.
## # ... with 1 more variable: ess_tail <dbl>
```

```
fit$cmdstan_diagnose()
```

```
## Processing csv files: C:/Users/fyref/AppData/Local/Temp/RtmpK0kOQL/mod1-202204191846-1-48ad28.csv, C
##
## Checking sampler transitions treedepth.
## Treedepth satisfactory for all transitions.
##
## Checking sampler transitions for divergences.
## No divergent transitions found.
##
## Checking E-BFMI - sampler transitions HMC potential energy.
## E-BFMI satisfactory.
##
## Effective sample size satisfactory.
##
## Split R-hat values satisfactory all parameters.
```

```
##
## Processing complete, no problems detected.
```

The rhat and effictive sample sizes (in bulk and for the tails) are right where they should be for the above parameters, with an rhat below 1.05, and high effective sample sizes, including for the log-probability. This indicates convergence.

Below are some of the diagnostics we did in class:

```
library(coda)
```

```
##
## Attaching package: 'coda'

## The following object is masked from 'package:rstan':
##
##     traceplot
```

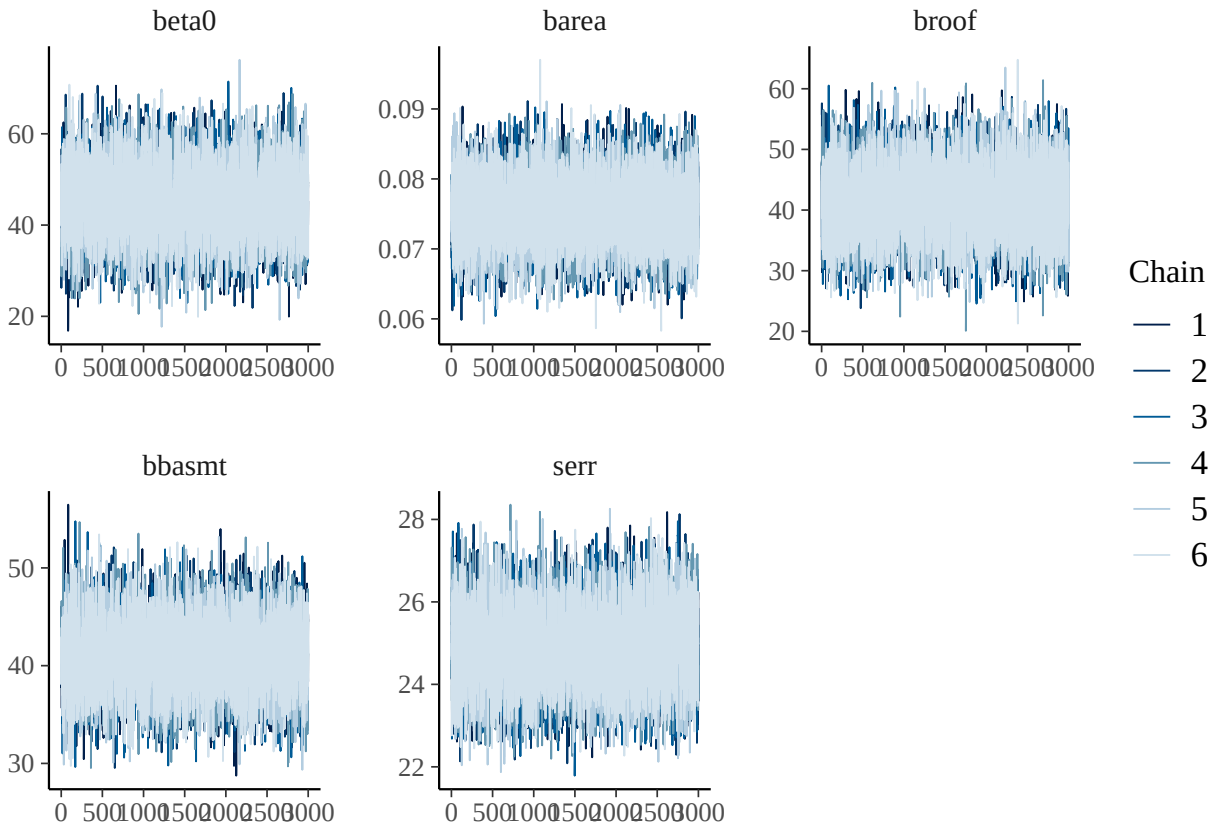
```
stanfit <- rstan::read_stan_csv(fit$output_files())

samps <- rstan::extract(stanfit)
chains <- cbind(samps[[1]], samps[[2]], samps[[3]], samps[[4]], samps[[5]])
colnames(chains) <- c("beta0","barea","broof","bbasmt","serr")
sims <- as.mcmc(chains)
effectiveSize(sims)
```

```
##    beta0    barea    broof   bbasmt     serr
## 18000.00 18000.00 18000.00 18780.32 17595.94
```

```
raftery.diag(sims)
```

```
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.005
## Probability (s) = 0.95
##
##         Burn-in  Total Lower bound  Dependence
##         (M)      (N)   (Nmin)       factor (I)
##  beta0  2        3918  3746         1.050
##  barea  2        3676  3746         0.981
##  broof  2        3795  3746         1.010
##  bbasmt 2        3727  3746         0.995
##  serr   2        3727  3746         0.995
```

```
# Traceplots
mcmc_trace(fit$draws(), pars = c("beta0","barea","broof","bbasmt","serr"))
```



This confirms within chain and global convergence.

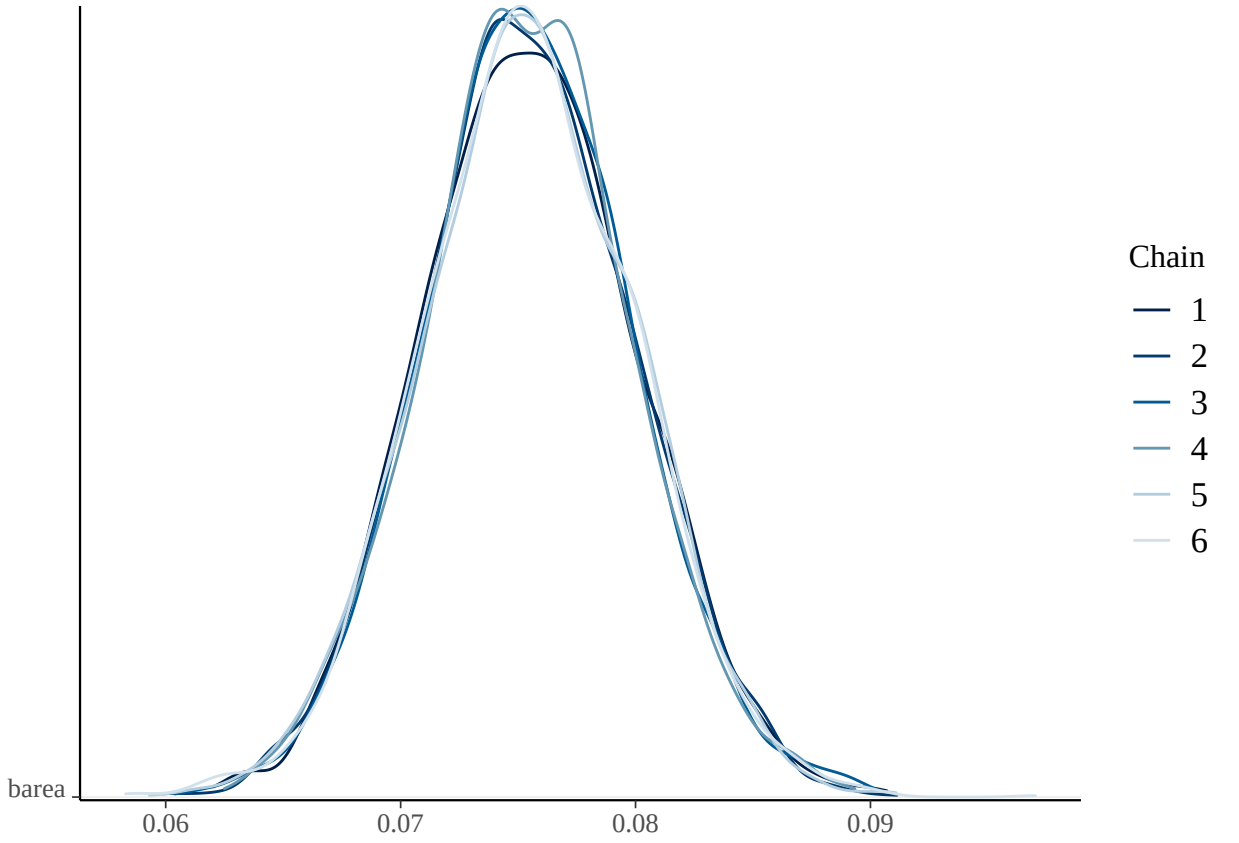
Here is the 95% equal tail estimates for `barea`.

```
fit$draws(variables = 'barea', format = 'draws_matrix') |>
  quantile(c(0.025, 0.975))
```

```
##       2.5%      97.5%
## 0.06672990 0.08439003
```

Here is the associated posterior density for the chains:

```
mcmc_dens_chains(fit$draws('barea'))
```



2. The data we are considering are from the Body Project, a dissonance based eating disorder prevention intervention.
   Female adolescents were randomized to one of four conditions: the dissonance intervention (DI; n=114), a healthy-weight management program (HW; n=117), an expressive writing control (EW, n=123), and an assessment-only control (AO; n=126).
   For the purposes of this exam, we are only considering two treatments, DI and AO. The DI treatment was delivered to groups of participants, (average group size=6.7). The AO treatment was delivered to individuals.
   We would expect the responses in the DI groups to show some within group correlation. The researchers' question of interest was whether the DI treatment lowered the response (called TII2 in the data file).

   The data file (called *bp.dat*) will be emailed to you. There are seven columns in the data file. The first column is a subject id that has issues because we are only looking at two treatments. The second column is a groupid that also had some issues. Ignore the first two columns. The next column is the treatment condition. It is 0 for the DI treatment and 3 for the AO treatment, the only two you care about. The next column is labeled DIS and it also is a treatment indicator, 1 for the DI treatment and 0 for the AO treatment. These two columns have identical information. The next column is TII2, which is the response of interest. The final two columns are subject id and group id that have been fixed up for this data set.

   There are many ways to think about a model in this case. I think the most reasonable one for us is a hierarchical model, as there are multiple sources of variability.Subject variability in both the AO and DI treatments, as well as group variability for the DI treatment. For picking priors you should be aware that all the variance components for this data are fairly small (all less than 1).
   You should write code to evaluate the conjecture that DI is a better treatment than AO.

The most important item you will hand in is your code, either JAGS or Stan. If you get your code to run I would also like to see convergence diagnostics, a summary output table, an equal tail interval for the parameter used to indicate the possible presence of a treatment effect, and a plot of the posterior density of this same parameter.

```
dat <- read_table('bp.dat')
colnames(dat) <- colnames(dat) |> str_replace_all('"', "")
dat <- dat |> select(-ID, -GROUPID, - TXCOND)

# Fixing group ID's for individuals, to be continuous

dat[115:length(dat$DIS),]$grpid1 <- 18:143

data_list <- list(n_DI = sum(dat$DIS),
                  n_AO = length(dat$DIS) - sum(dat$DIS),
                  n_grp = length(table(dat$grpid1)),
                  N = length(dat$TII2),
                  TII2 = dat$TII2,
                  tmt = dat$DIS,
                  n_tmt = 2,
                  id = dat$id,
                  groupid = dat$grpid1)
```

It seems that the individuals who participated in DI treatment participated in groups, while those who did AO on their own were part of their own group. I

```
mod <- cmdstan_model('mod2.stan')

fit <- mod$sample(
  data = data_list,
  seed = 324,
  chains = 6,
  parallel_chains = 3,
  refresh = 1000,
  sig_figs = 5,
  iter_sampling = 2000
)
```

```
## Running MCMC with 6 chains, at most 3 in parallel...
##
## Chain 1 Iteration:    1 / 3000 [  0%]  (Warmup)
## Chain 2 Iteration:    1 / 3000 [  0%]  (Warmup)
## Chain 3 Iteration:    1 / 3000 [  0%]  (Warmup)
## Chain 2 Iteration: 1000 / 3000 [ 33%]  (Warmup)
## Chain 2 Iteration: 1001 / 3000 [ 33%]  (Sampling)
## Chain 3 Iteration: 1000 / 3000 [ 33%]  (Warmup)
## Chain 1 Iteration: 1000 / 3000 [ 33%]  (Warmup)
## Chain 3 Iteration: 1001 / 3000 [ 33%]  (Sampling)
## Chain 1 Iteration: 1001 / 3000 [ 33%]  (Sampling)
## Chain 2 Iteration: 2000 / 3000 [ 66%]  (Sampling)
## Chain 1 Iteration: 2000 / 3000 [ 66%]  (Sampling)
## Chain 3 Iteration: 2000 / 3000 [ 66%]  (Sampling)
## Chain 2 Iteration: 3000 / 3000 [100%]  (Sampling)
```

```
## Chain 2 finished in 4.9 seconds.
## Chain 1 Iteration: 3000 / 3000 [100%]  (Sampling)
## Chain 1 finished in 5.0 seconds.
## Chain 4 Iteration:    1 / 3000 [  0%]  (Warmup)
## Chain 3 Iteration: 3000 / 3000 [100%]  (Sampling)
## Chain 5 Iteration:    1 / 3000 [  0%]  (Warmup)
## Chain 3 finished in 5.5 seconds.
## Chain 6 Iteration:    1 / 3000 [  0%]  (Warmup)
## Chain 4 Iteration: 1000 / 3000 [ 33%]  (Warmup)
## Chain 4 Iteration: 1001 / 3000 [ 33%]  (Sampling)
## Chain 5 Iteration: 1000 / 3000 [ 33%]  (Warmup)
## Chain 6 Iteration: 1000 / 3000 [ 33%]  (Warmup)
## Chain 5 Iteration: 1001 / 3000 [ 33%]  (Sampling)
## Chain 6 Iteration: 1001 / 3000 [ 33%]  (Sampling)
## Chain 6 Iteration: 2000 / 3000 [ 66%]  (Sampling)
## Chain 4 Iteration: 2000 / 3000 [ 66%]  (Sampling)
## Chain 5 Iteration: 2000 / 3000 [ 66%]  (Sampling)
## Chain 4 Iteration: 3000 / 3000 [100%]  (Sampling)
## Chain 6 Iteration: 3000 / 3000 [100%]  (Sampling)
## Chain 4 finished in 5.1 seconds.
## Chain 6 finished in 4.6 seconds.
## Chain 5 Iteration: 3000 / 3000 [100%]  (Sampling)
## Chain 5 finished in 6.6 seconds.
##
## All 6 chains finished successfully.
## Mean chain execution time: 5.3 seconds.
## Total execution time: 12.3 seconds.
```

```
fit$summary()
```

```
## # A tibble: 151 x 10
##    variable       mean  median     sd    mad       q5     q95  rhat ess_bulk
##    <chr>         <dbl>   <dbl>  <dbl>  <dbl>    <dbl>   <dbl> <dbl>    <dbl>
##  1 lp__        -636.   -649.   56.6   51.8   -710.    -526.    1.04     114.
##  2 group_mu[1]    1.09    1.07  0.309  0.273    0.597    1.62  1.01    8154.
##  3 group_mu[2]    1.04    1.03  0.305  0.267    0.552    1.57  1.01   11064.
##  4 group_mu[3]    1.04    1.03  0.319  0.279    0.525    1.58  1.01    7339.
##  5 group_mu[4]    1.02    1.02  0.303  0.264    0.521    1.54  1.01    9967.
##  6 group_mu[5]    1.06    1.05  0.309  0.271    0.572    1.60  1.01    9665.
##  7 group_mu[6]    1.05    1.03  0.309  0.269    0.550    1.57  1.01    9567.
##  8 group_mu[7]    1.05    1.04  0.311  0.269    0.554    1.59  1.01    7691.
##  9 group_mu[8]    1.03    1.02  0.309  0.270    0.529    1.55  1.01    9877.
## 10 group_mu[9]    1.07    1.05  0.313  0.272    0.576    1.61  1.01    8374.
## # ... with 141 more rows, and 1 more variable: ess_tail <dbl>
```

The model I wrote is not ideal, after struggling to come up with something better, I believe the best model would be perhaps a logistic regression model, or perhaps the model would run better by computing the difference between the two means.

I think since there is very little difference between the individuals, the model is having a hard time picking up the differences.

I tried squaring the response to get better scaling, but I think the problem lies with my chosen model.

I tried to account for the different sources of variablity, but it made the model too complicated.

I could do some research on the factors that go into these scores. My analysis isn't very useful anyways because I don't have the knowledge of how these scores were determined, or created.

This is the best I can do with the times and resources I have been given.