Pilot Analysis: Linear Log Odds Model of Probability of Superiority

In this document, we build a linear log odds model of probability of superiority judgments.

The LLO model follows from related work (https://www.frontiersin.org/articles/10.3389/fnins.2012.00001/full) suggesting that the human perception of probability is encoded on a log odds scale. On this scale, the slope of a linear model represents the shape and severity of the function describing bias in probability perception. The greater the deviation of from a slope of 1 (i.e., ideal performance), the more biased the judgments of probability. Slopes less than one correspond to the kind of bias predicted by excessive attention to the mean. On the same log odds scale, the intercept is a crossover-point which should be proportional to the number of categories of possible outcomes among which probability is divided. In our case, the intercept should be about 0.5 since workers are judging the probability of a team getting more points with a new player than without.

In this pilot, we allowed people to respond on a scale of 0-100, and we only have samples where the ground truth is greater than 50%.

Load and Prepare Data

We load worker responses from our pilot and do some preprocessing.

```
# read in data
full_df <- read_csv("pilot-anonymous.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   workerId = col_character(),
##   batch = col_integer(),
##   n_trials = col_integer(),
##   n_data_conds = col_integer(),
##   condition = col_character(),
##   start_means = col_character(),
##   cutoff = col_integer(),
##   max_bonus = col_integer(),
##   numeracy = col_integer(),
##   gender = col_character(),
##   age = col_character(),
##   education = col_character(),
##   chart_use = col_character(),
##   difficulties = col_character(),
##   intervene = col_integer(),
##   outcome = col_character(),
##   pSup = col_integer(),
##   sdDiff = col_integer(),
##   trial = col_character(),
##   trialIdx = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
# preprocessing
responses_df <- full_df %>%
  rename( # rename to convert away from camel case
    worker_id = workerId,
    ground_truth = groundTruth,
    sd_diff = sdDiff,
    p_award_with = pAwardWith,
    p_award_without = pAwardWithout,
    account_value = accountValue,
    p_superiority = pSup,
    start_time = startTime,
    resp_time = respTime,
    trial_dur = trialDur,
    trial_idx = trialIdx
  ) %>%
  # remove practice and mock trials from responses dataframe, leave in full version
  filter(trial_idx != "practice", trial_idx != "mock") %>%
  # mutate rows where intervene == -1 for some reason
  mutate(
    intervene = if_else(intervene == -1,
                        # repair
                        if_else((payoff == (award_value - 1) | payoff == -1),
                                1, # payed for intervention
                                0), # didn't pay for intervention
                        # don't repair
                        as.numeric(intervene) # hack to avoid type error
                        )
  ) %>%
  # set up factors for modeling
  mutate(
    # add a variable to note whether the chart they viewed showed means
    means = as.factor((start_means == "True" & as.numeric(trial) <= (n_trials / 2)) | (s
tart_means == "False" & as.numeric(trial) > (n_trials / 2))),
    start_means = as.factor(start_means == "True"),
    sd_diff = as.factor(sd_diff),
    trial_number = as.numeric(trial)
  )

head(responses_df)
```

```
## # A tibble: 6 x 37
##   worker_id batch n_trials n_data_conds condition baseline es_threshold
##   <chr>     <int>    <int>        <int> <chr>        <dbl>        <dbl>
## 1 290ad208      2       34           18 HOPs           0.5          0.9
## 2 290ad208      2       34           18 HOPs           0.5          0.9
## 3 290ad208      2       34           18 HOPs           0.5          0.9
## 4 290ad208      2       34           18 HOPs           0.5          0.9
## 5 290ad208      2       34           18 HOPs           0.5          0.9
## 6 290ad208      2       34           18 HOPs           0.5          0.9
## # … with 30 more variables: start_means <fct>, award_value <dbl>,
## #   starting_value <dbl>, exchange <dbl>, cutoff <int>, max_bonus <int>,
## #   total_bonus <dbl>, duration <dbl>, numeracy <int>, gender <chr>,
## #   age <chr>, education <chr>, chart_use <chr>, difficulties <chr>,
## #   account_value <dbl>, ground_truth <dbl>, intervene <dbl>,
## #   outcome <chr>, p_award_with <dbl>, p_award_without <dbl>,
## #   p_superiority <int>, payoff <dbl>, resp_time <dbl>, sd_diff <fct>,
## #   start_time <dbl>, trial <chr>, trial_dur <dbl>, trial_idx <chr>,
## #   means <fct>, trial_number <dbl>
```

We need the data in a format where it is prepared for modeling. We censor responses to the range 0.5% to 99.5% where responses at these bounds reflect an intended response at the bound or higher. By rounding responses to the nearest 0.5%, we assume that the response scale has a resolution of 1% in practice while avoiding values of infinity on a log scale that our model cannot handle. Last, we converte both probability of superiority judgments and the ground truth to a logit scale.

```
# create data frame for model
model_df <- responses_df %>%
  mutate(
    # recode responses greater than 99.5% and less than 0.5% to avoid values of +/- Inf
 on a logit scale
    p_superiority = if_else(p_superiority > 99.5,
                            99.5,
                            if_else(p_superiority < 0.5,
                                    0.5,
                                    as.numeric(p_superiority))),
    # apply logit function to p_sup judgments and ground truth
    lo_p_sup = qlogis(p_superiority / 100),
    lo_ground_truth = qlogis(ground_truth),
    # scale and center trial order
    trial = (trial_number - as.numeric(n_trials) / 2) / as.numeric(n_trials)
  )
```

Now, lets apply our exclusion criteria.

```
# determine exclusions
exclude_df <- model_df %>%
  # attention check trials where ground truth = c(0.5, 0.999)
  mutate(failed_check = (ground_truth == 0.5 & intervene != 0) | (ground_truth == 0.999
 & intervene != 1)) %>%
  group_by(worker_id) %>%
  summarise(
    failed_attention_checks = sum(failed_check),
    exclude = failed_attention_checks > 0
    # p_sup_less_than_50 = sum(p_superiority < 50) / n(),
    # exclude = (failed_attention_checks > 0 | p_sup_less_than_50 > 0.3)
  ) %>%
  select(worker_id, exclude)

# apply exclusion criteria and remove attention checks to modeling data set
model_df <- model_df %>%
  left_join(exclude_df, by = "worker_id") %>%
  filter(exclude == FALSE) %>%
  filter(ground_truth > 0.5 & ground_truth < 0.999)

# how many remaining workers per condition
model_df %>%
  group_by(condition, start_means) %>% # between subject manipulations
  summarise(
    n_workers = length(unique(worker_id))
  )
```

```
## # A tibble: 4 x 3
## # Groups:   condition [2]
##   condition start_means n_workers
##   <chr>     <fct>           <int>
## 1 HOPs      FALSE               3
## 2 HOPs      TRUE                8
## 3 intervals FALSE               7
## 4 intervals TRUE                6
```

The table above shows the number of included workers in each between-subjects condition.

Distribution of Probability of Superiority Judgments

We start as simply as possible by just modeling the distribution of probability of superiority judgements on the log odds scale.

Before we fit the model to our data, let's check that our priors seem reasonable. We'll use a weakly informative prior for the intercept parameter since we want the population-level centered intercept to be flexible. We set the expected value of the prior on the intercept equal to the mean value of the ground truth that we sampled (in log odds units).

```
# get mean value of ground truth sampled in log odds units
model_df %>% select(lo_ground_truth) %>% summarize(mean = mean(lo_ground_truth))
```

```
## # A tibble: 1 x 1
##    mean
##   <dbl>
## 1  1.30
```

```
# get_prior(data = model_df, family = "gaussian", formula = lo_p_sup ~ 1)

# starting as simple as possible: learn the distribution of lo_p_sup
prior.lo_p_sup <- brm(data = model_df, family = "gaussian",
              lo_p_sup ~ 1,
              prior = c(prior(normal(1.3, 1), class = Intercept),
                        prior(normal(0, 1), class = sigma)),
              sample_prior = "only",
              iter = 3000, warmup = 500, chains = 2, cores = 2)
```

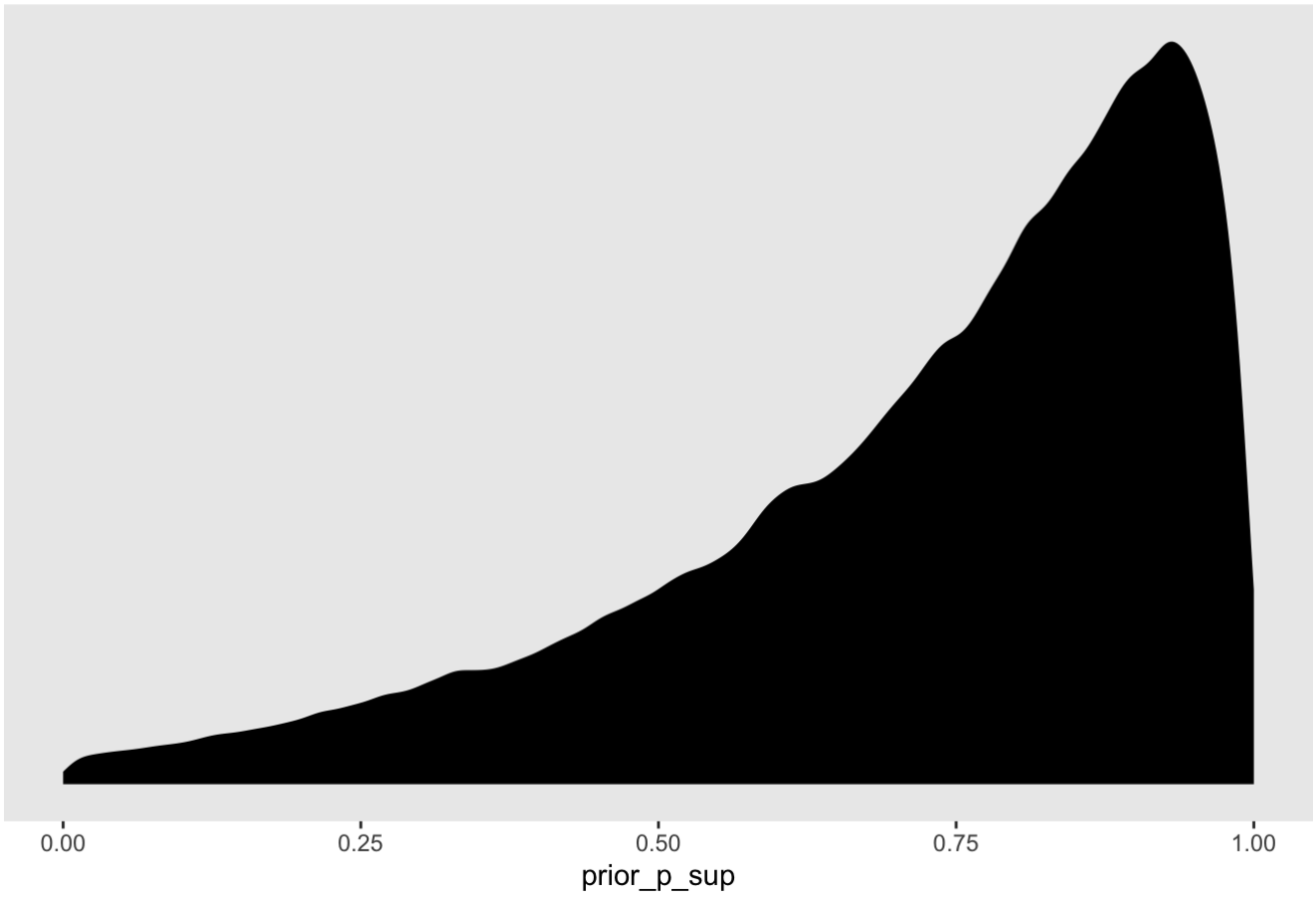
```
## Compiling the C++ model
```

```
## Start sampling
```

Let's look at our prior predictive distribution. For this intercept model, it should be skewwed left because we have located our prior near 74%. We should see a peak near the upper bound of the probability scale.

```
# prior predictive check
model_df %>%
  select() %>%
  add_predicted_draws(prior.lo_p_sup, prediction = "lo_p_sup", seed = 1234) %>%
  mutate(
    # transform to probability units
    prior_p_sup = plogis(lo_p_sup)
  ) %>%
  ggplot(aes(x = prior_p_sup)) +
  geom_density(fill = "black", size = 0) +
  scale_y_continuous(NULL, breaks = NULL) +
  labs(subtitle = "Prior predictive distribution for probability of superiority") +
  theme(panel.grid = element_blank())
```

Prior predictive distribution for probability of superiority



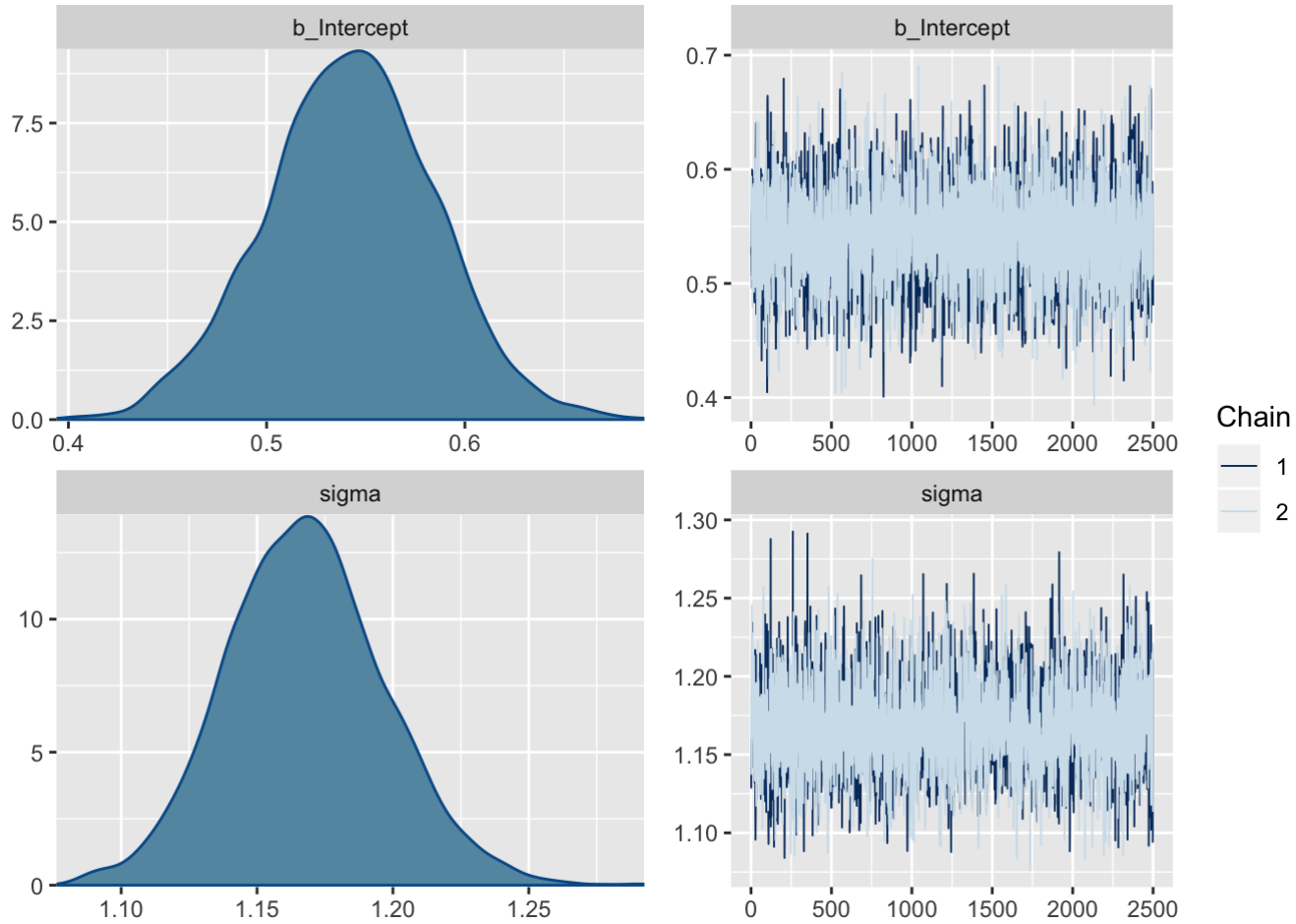
Now, let's fit the model to data. This is just trying to estimate the mean response regardless of the ground truth.

```
# starting as simple as possible: learn the distribution of lo_p_sup
m.lo_p_sup <- brm(data = model_df, family = "gaussian",
              lo_p_sup ~ 1,
              prior = c(prior(normal(1.3, 1), class = Intercept),
                        prior(normal(0, 1), class = sigma)),
              iter = 3000, warmup = 500, chains = 2, cores = 2,
              file = "model-fits/lo_mdl")
```

Check diagnostics:

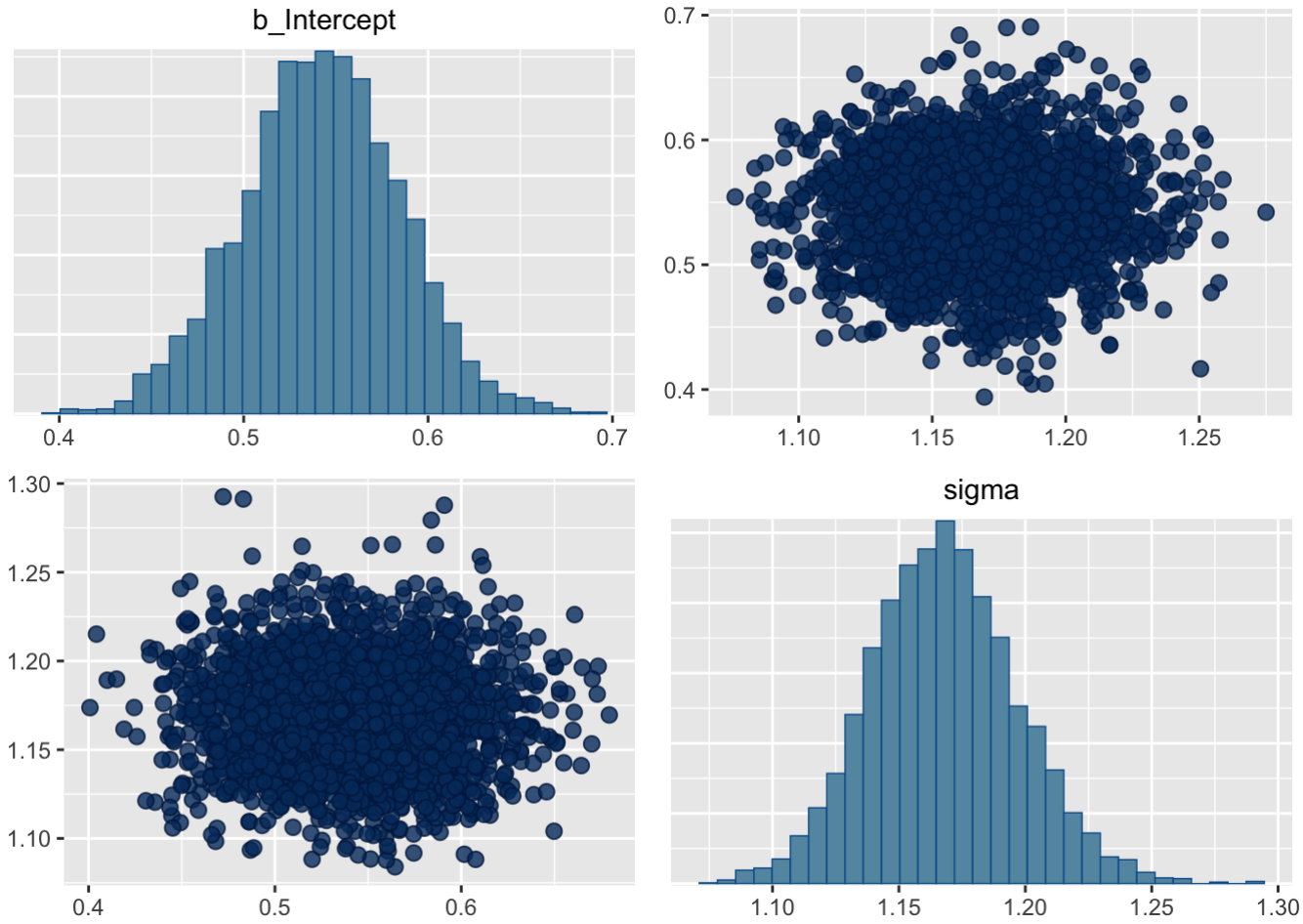
- Trace plots

```
# trace plots
plot(m.lo_p_sup)
```



- Pairs plot

```
# pairs plot
pairs(m.lo_p_sup)
```



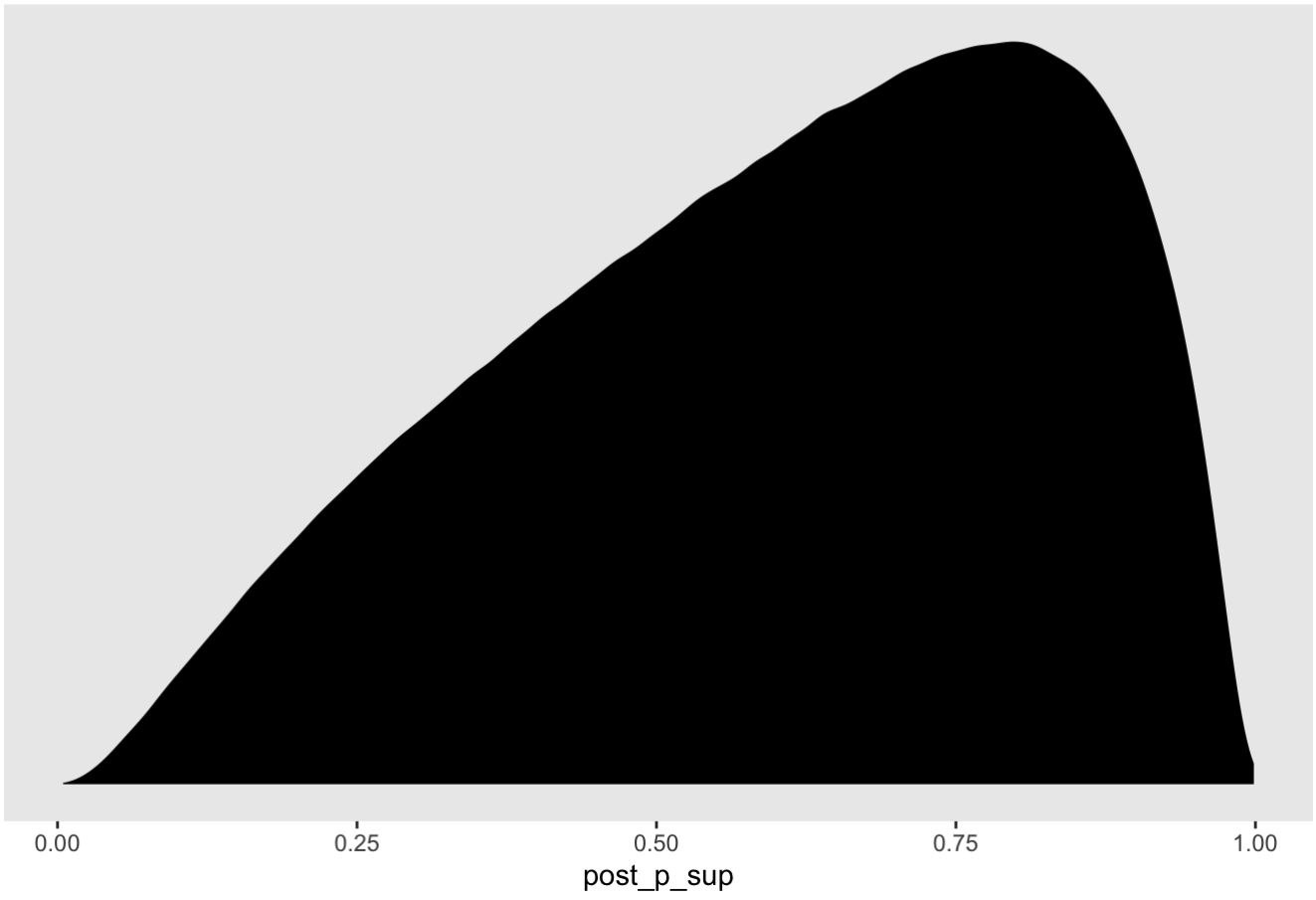
- Summary

```
# model summary
print(m.lo_p_sup)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: lo_p_sup ~ 1
##    Data: model_df (Number of observations: 768)
## Samples: 2 chains, each with iter = 3000; warmup = 500; thin = 1;
##          total post-warmup samples = 5000
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## Intercept     0.54      0.04     0.46     0.63       4943 1.00
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## sigma     1.17      0.03     1.11     1.23       3669 1.00
##
## Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
## is a crude measure of effective sample size, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

Let's check our posterior predictive distribution.

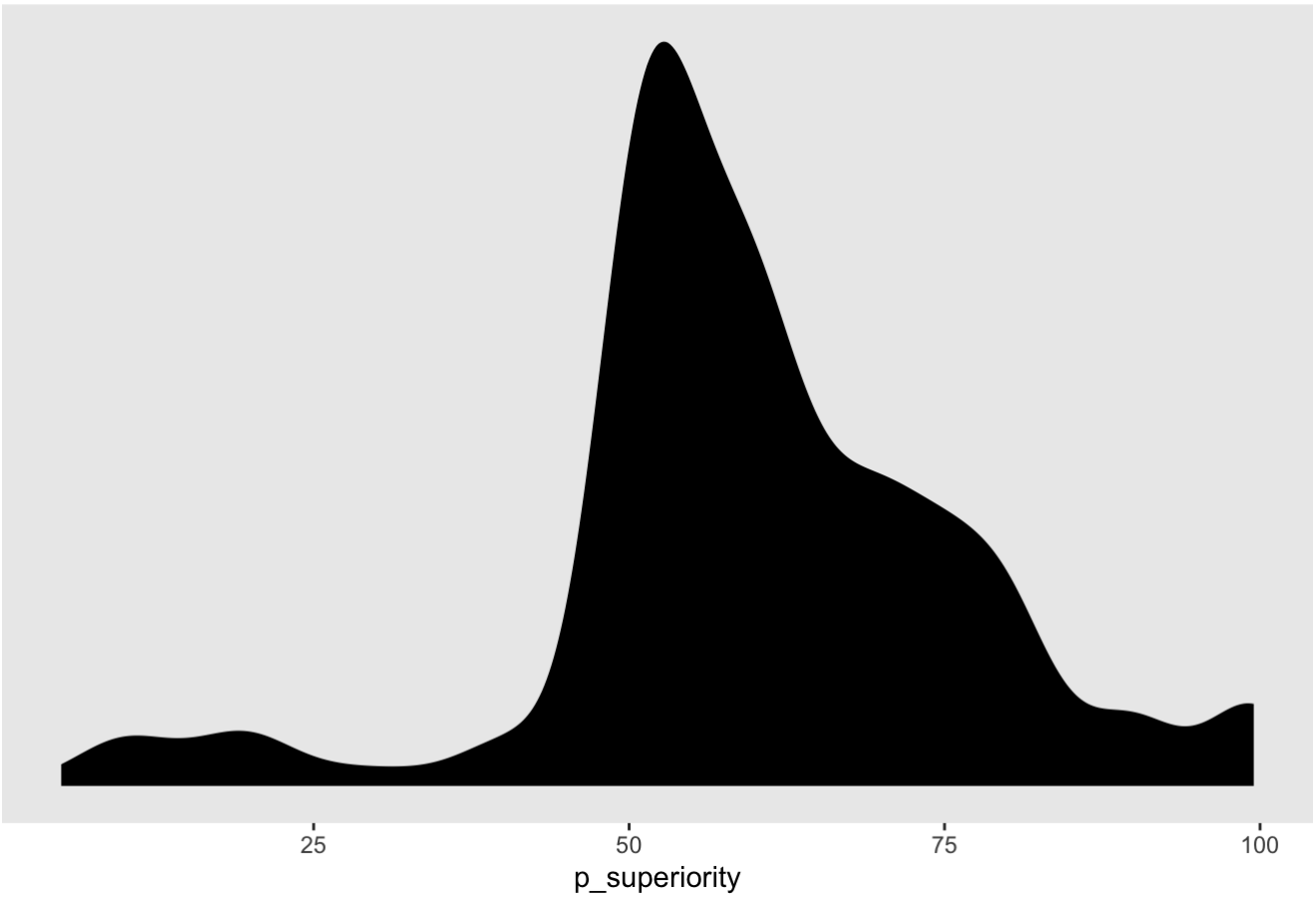
```
# posterior predictive check
model_df %>%
  select() %>%
  add_predicted_draws(m.lo_p_sup, prediction = "lo_p_sup", seed = 1234) %>%
  mutate(
    # transform to probability units
    post_p_sup = plogis(lo_p_sup)
    ) %>%
  ggplot(aes(x = post_p_sup)) +
  geom_density(fill = "black", size = 0) +
  scale_y_continuous(NULL, breaks = NULL) +
  labs(subtitle = "Posterior predictive distribution for probability of superiority",
       post_p_sup = NULL) +
  theme(panel.grid = element_blank())
```



How do these predictions compare to the observed data?

```
# data density
model_df %>%
  ggplot(aes(x = p_superiority)) +
  geom_density(fill = "black", size = 0) +
  scale_y_continuous(NULL, breaks = NULL) +
  labs(subtitle = "Data distribution for probability of superiority") +
  theme(panel.grid = element_blank())
```

Data distribution for probability of superiority



Our model is not sensitive to the ground truth, so we expect to see a mismatch here.

Linear Log Odds Model of Probability of Superiority

Now well add in a slope parameter to make our model sensitive to the ground truth. This is the simplest version of our linear log odds (LLO) model.

Before we fit the model to our data, let's check that our priors seem reasonable. Since we are now including a slope parameter for the ground truth in our model, we can dial down the width of our prior for sigma to avoid over-dispersion of predicted responses.

```
# get_prior(data = model_df, family = "gaussian", formula = lo_p_sup ~ lo_ground_truth)

# simple LLO model
prior.llo_p_sup <- brm(data = model_df, family = "gaussian",
                       lo_p_sup ~ lo_ground_truth,
                       prior = c(prior(normal(1, 0.5), class = b),
                                 prior(normal(1.3, 1), class = Intercept),
                                 prior(normal(0, 0.5), class = sigma)),
                       sample_prior = "only",
                       iter = 3000, warmup = 500, chains = 2, cores = 2)
```

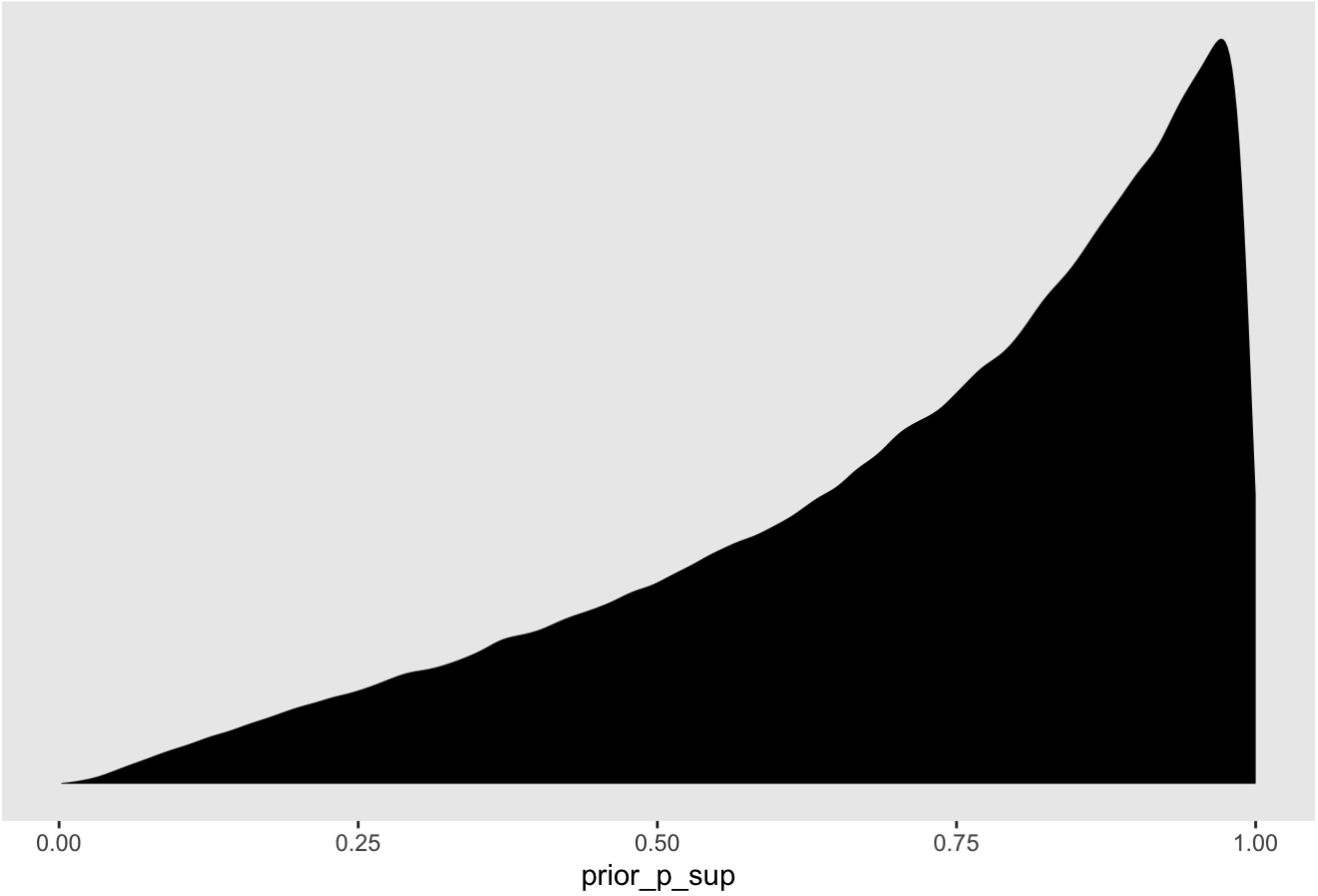
```
## Compiling the C++ model
```

```
## Start sampling
```

Let's look at our prior predictive distribution. For this linear model, we should see density spread more evenly across probability values.

```
# prior predictive check
model_df %>%
  select(lo_ground_truth) %>%
  add_predicted_draws(prior.llo_p_sup, prediction = "lo_p_sup", seed = 1234) %>%
  mutate(
    # transform to probability units
    prior_p_sup = plogis(lo_p_sup)
    ) %>%
  ggplot(aes(x = prior_p_sup)) +
  geom_density(fill = "black", size = 0) +
  scale_y_continuous(NULL, breaks = NULL) +
  labs(subtitle = "Prior predictive distribution for probability of superiority") +
  theme(panel.grid = element_blank())
```

Prior predictive distribution for probability of superiority



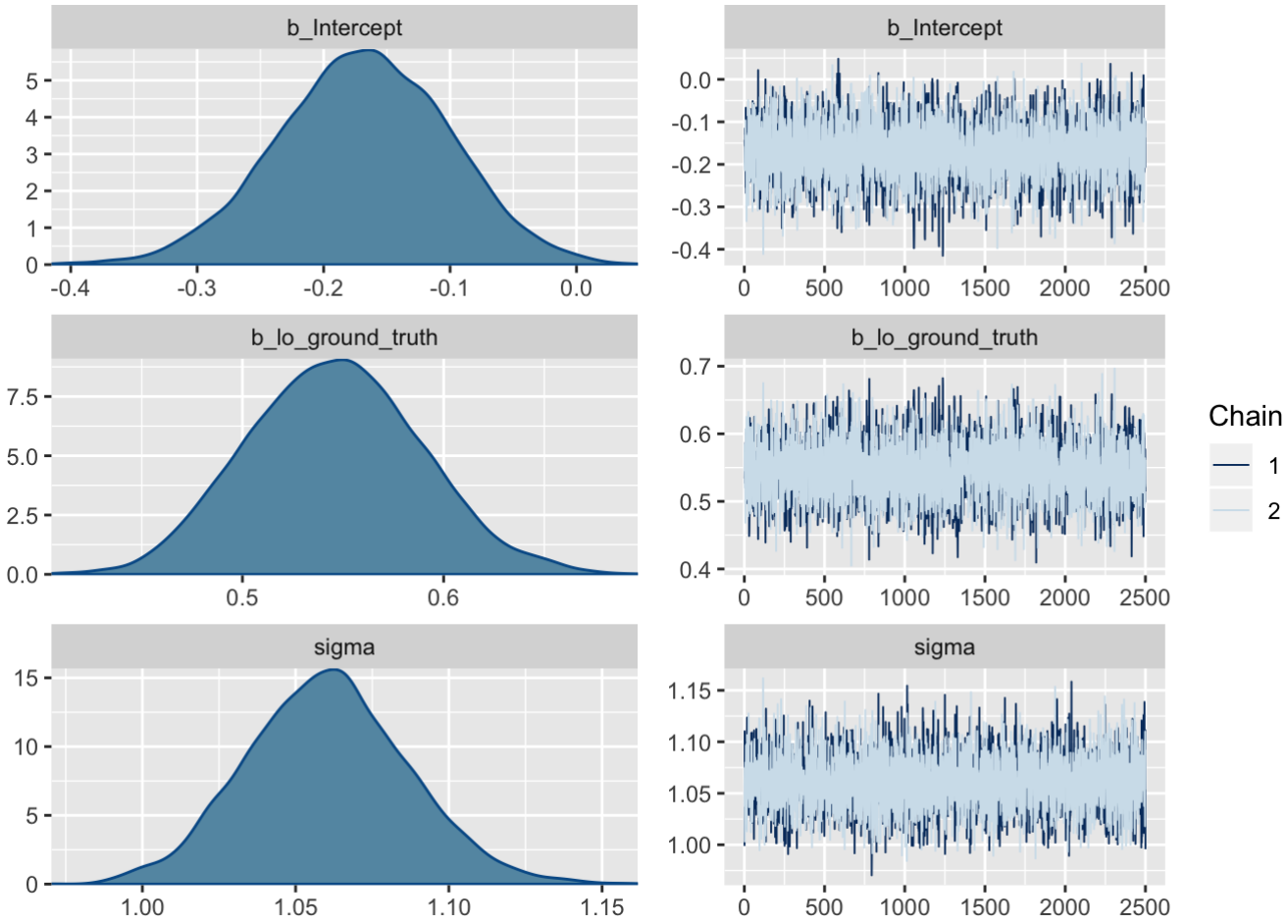
Now let's fit the model to data.

```
# simple LLO model
m.llo_p_sup <- brm(data = model_df, family = "gaussian",
                   lo_p_sup ~ lo_ground_truth,
                   prior = c(prior(normal(1, 0.5), class = b),
                             prior(normal(1.3, 1), class = Intercept),
                             prior(normal(0, 0.5), class = sigma)),
                   iter = 3000, warmup = 500, chains = 2, cores = 2,
                   file = "model-fits/llo_mdl")
```

Check diagnostics:

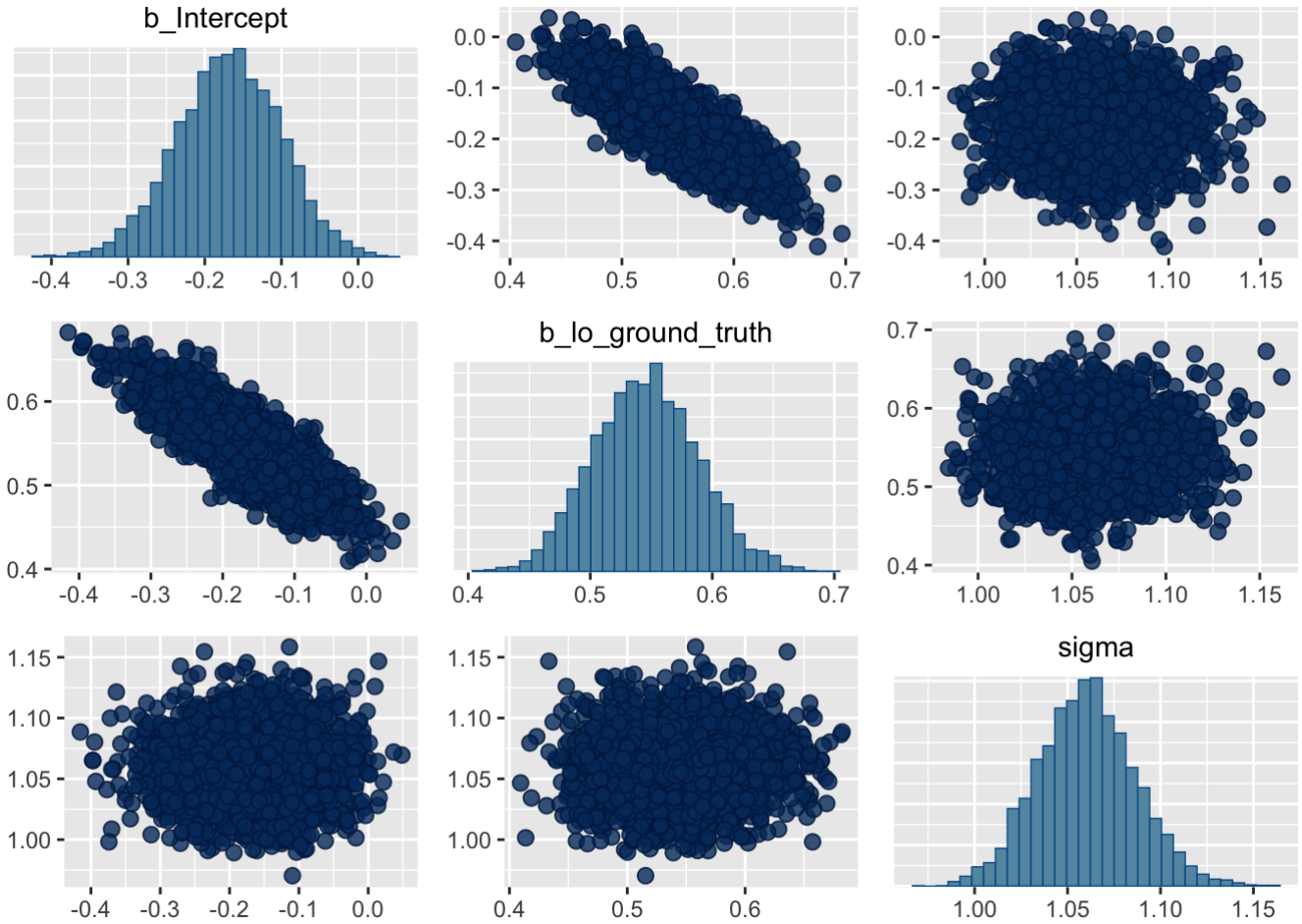
- Trace plots

```
# trace plots
plot(m.llo_p_sup)
```



- Pairs plot

```
# pairs plot
pairs(m.llo_p_sup)
```



Our slope and intercept parameters seem pretty highly correlated. Maybe adding hierarchy to our model will remedy this.

- Summary

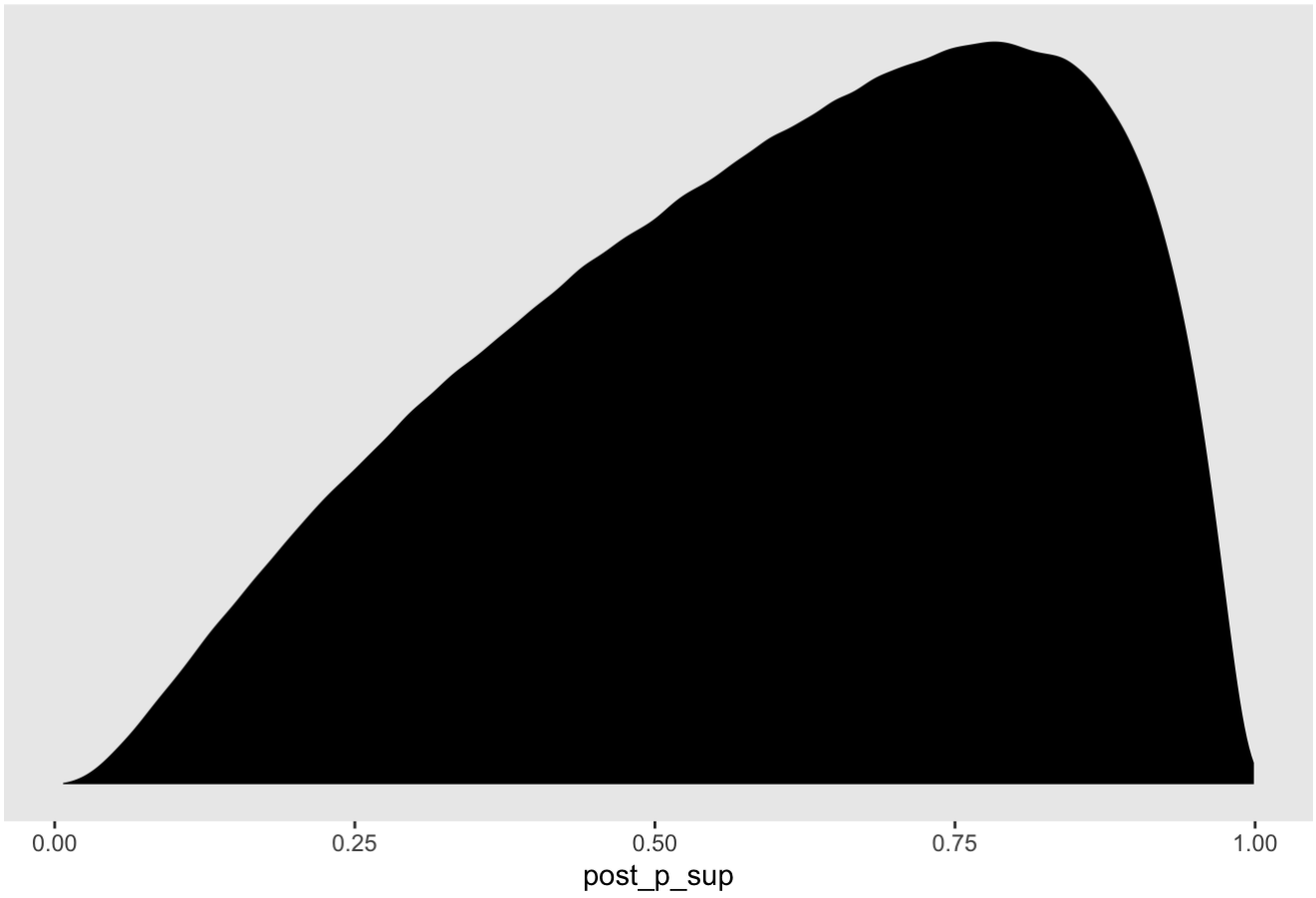
```
# model summary
print(m.llo_p_sup)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: lo_p_sup ~ lo_ground_truth
##    Data: model_df (Number of observations: 768)
## Samples: 2 chains, each with iter = 3000; warmup = 500; thin = 1;
##          total post-warmup samples = 5000
##
## Population-Level Effects:
##                 Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## Intercept          -0.17      0.07    -0.31    -0.04       4682 1.00
## lo_ground_truth     0.55      0.04     0.46     0.63       4313 1.00
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## sigma     1.06      0.03     1.01     1.11       6289 1.00
##
## Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
## is a crude measure of effective sample size, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

Let's check our posterior predictive distribution.

```
# posterior predictive check
model_df %>%
  select(lo_ground_truth) %>%
  add_predicted_draws(m.llo_p_sup, prediction = "lo_p_sup", seed = 1234) %>%
  mutate(
    # transform to probability units
    post_p_sup = plogis(lo_p_sup)
  ) %>%
  ggplot(aes(x = post_p_sup)) +
  geom_density(fill = "black", size = 0) +
  scale_y_continuous(NULL, breaks = NULL) +
  labs(subtitle = "Posterior predictive distribution for probability of superiority") +
  theme(panel.grid = element_blank())
```

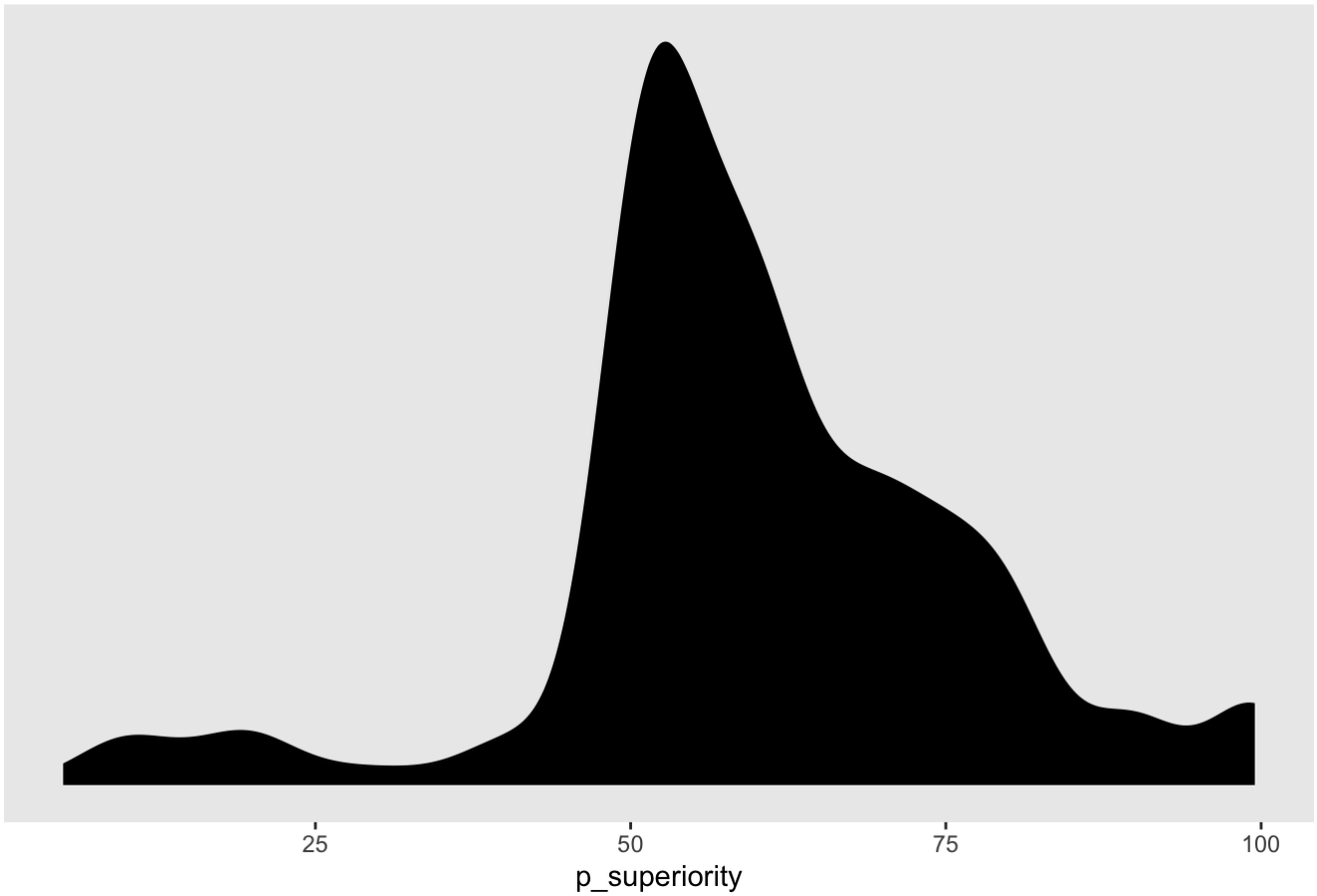
Posterior predictive distribution for probability of superiority



How do these predictions compare to the observed data?

```
# data density
model_df %>%
  ggplot(aes(x = p_superiority)) +
  geom_density(fill = "black", size = 0) +
  scale_y_continuous(NULL, breaks = NULL) +
  labs(subtitle = "Data distribution for probability of superiority") +
  theme(panel.grid = element_blank())
```

Data distribution for probability of superiority



Our model is now sensitive to the ground truth, but it is still having trouble fitting the data. It may be that the model is not capturing the individual variability in response patterns. Next we'll add hierarchy to our model.

Add Hierarchy for Slope, Intercepts, and Sigma

The models we've created thus far fail to account for much of the noise in the data. Here, we attempt to parse some heterogeniety in responses by modeling a random effect of worker on slopes, intercepts, and residual variance. This introduces a hierarchical component to our model in order to account for individual differences in the best fitting linear model for each worker's data.

Before we fit the model to our data, let's check that our priors seem reasonable. We are adding hyperpriors for the standard deviation of slopes, intercepts, and residual variation (i.e., sigma) per worker, as well as the correlation between them. We'll set moderately wide priors on these worker-level slope and intercept effects to avoid overregularizing potentially large individual variability. We'll also narrow the priors on sigma since we are now attributing variability to more sources and we want to avoid overdispersion. We'll set a prior on the correlation between slopes and intercepts per worker that avoids large absolute correlations.

```
# get_prior(data = model_df, family = "gaussian", formula = bf(lo_p_sup ~ (1 + lo_ground
_truth|sharecor|worker_id) + lo_ground_truth, sigma ~ (1|sharecor|worker_id)))

# hierarchical LLO model
prior.wrkr.llo_p_sup <- brm(data = model_df, family = "gaussian",
                            formula = bf(lo_p_sup ~ (1 + lo_ground_truth|sharecor|worker
_id) + lo_ground_truth,
                                         sigma ~ (1|sharecor|worker_id)),
                            prior = c(prior(normal(1, 0.5), class = b),
                                      prior(normal(1.3, 1), class = Intercept),
                                      prior(normal(0, 0.15), class = sd, group = worker_
id),
                                      prior(normal(0, 0.15), class = sd, dpar = sigma),
                                      prior(lkj(4), class = cor)),
                            sample_prior = "only",
                            iter = 3000, warmup = 500, chains = 2, cores = 2)
```

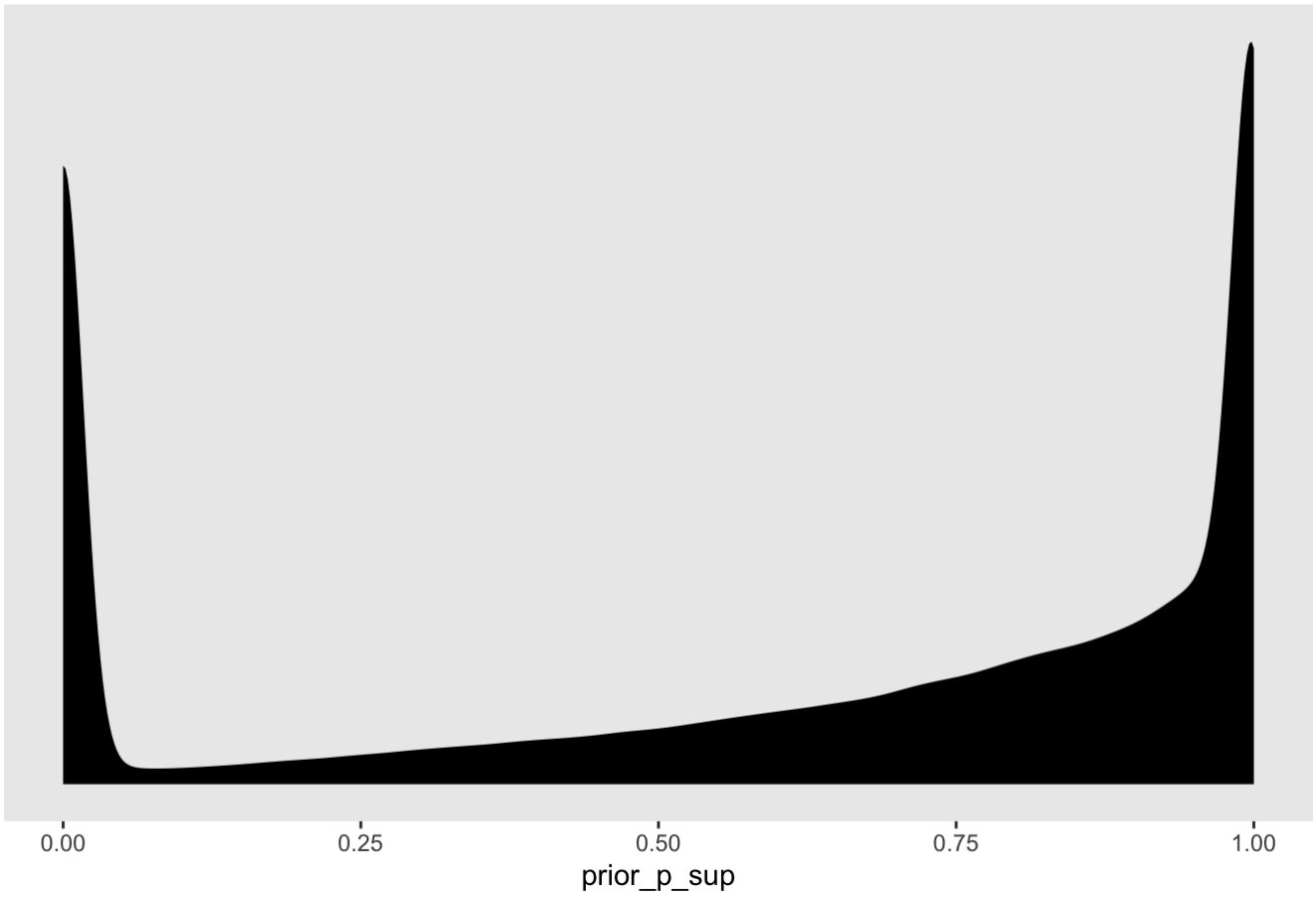
```
## Compiling the C++ model
```

```
## Start sampling
```

Let's look at our prior predictive distribution. This should predict more mass closer to 50% compared to our previous model because it allows more random variation in slopes and intercepts. We end up seeing a lot of mass at the edges of the scale suggesting overdispersion. However, it's probably best to err on the side of not making our priors on individual parameters overly narrow.

```
# prior predictive check
model_df %>%
  select(lo_ground_truth, worker_id) %>%
  add_predicted_draws(prior.wrkr.llo_p_sup, prediction = "lo_p_sup", seed = 1234) %>%
  mutate(
    # transform to probability units
    prior_p_sup = plogis(lo_p_sup)
    ) %>%
  ggplot(aes(x = prior_p_sup)) +
  geom_density(fill = "black", size = 0) +
  scale_y_continuous(NULL, breaks = NULL) +
  labs(subtitle = "Prior predictive distribution for probability of superiority") +
  theme(panel.grid = element_blank())
```

Prior predictive distribution for probability of superiority



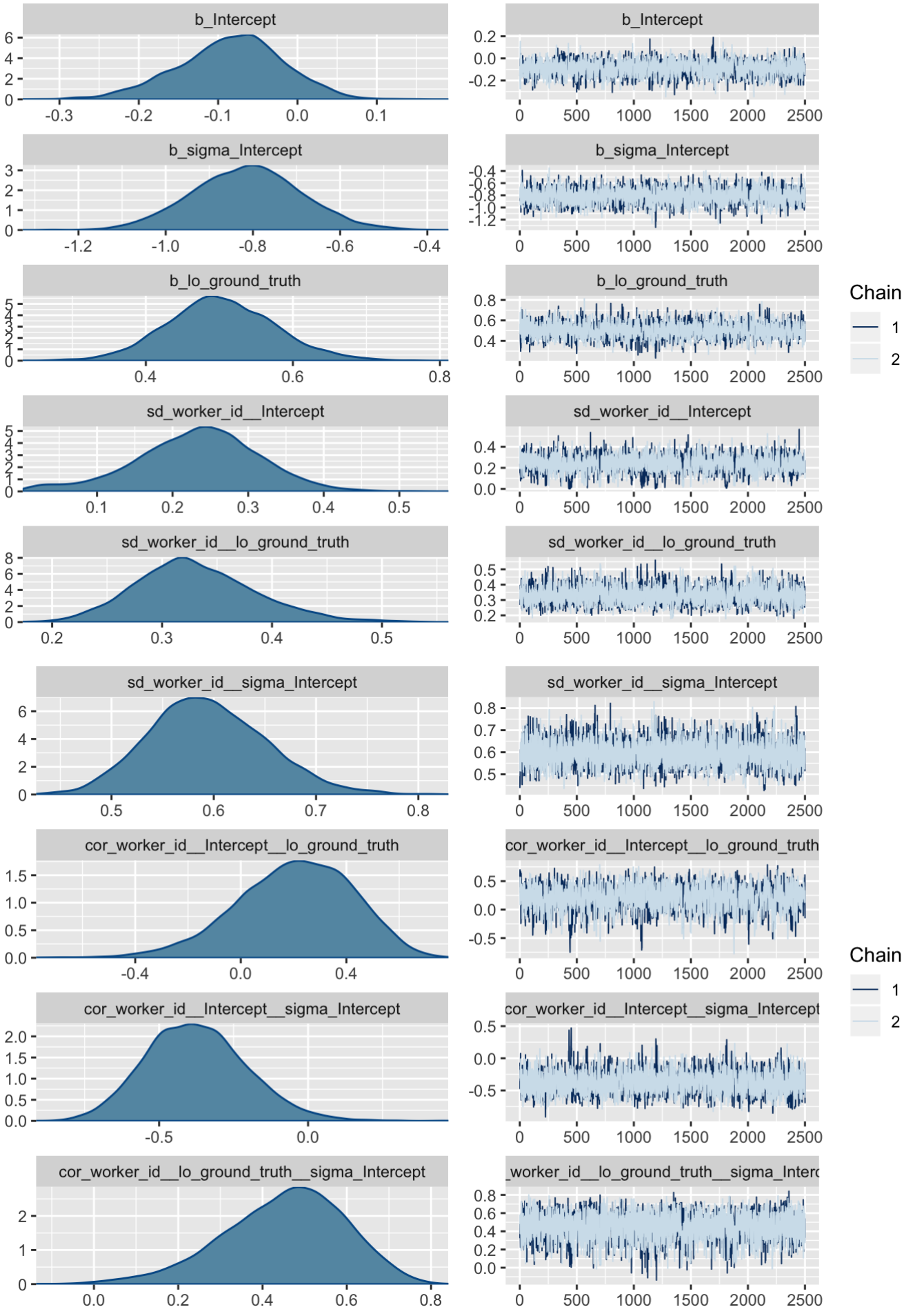
Now, let's fit the model to our data.

```
# hierarchical LLO model
m.wrkr.llo_p_sup <- brm(data = model_df, family = "gaussian",
                        formula = bf(lo_p_sup ~ (1 + lo_ground_truth|sharecor|worker_id)
+ lo_ground_truth,
                                     sigma ~ (1|sharecor|worker_id)),
                        prior = c(prior(normal(1, 0.5), class = b),
                                  prior(normal(1.3, 1), class = Intercept),
                                  prior(normal(0, 0.15), class = sd, group = worker_id),
                                  prior(normal(0, 0.15), class = sd, dpar = sigma),
                                  prior(lkj(4), class = cor)),
                        iter = 3000, warmup = 500, chains = 2, cores = 2,
                        control = list(adapt_delta = 0.99, max_treedepth = 12),
                        file = "model-fits/llo_mdl-wrkr")
```

Check diagnostics:

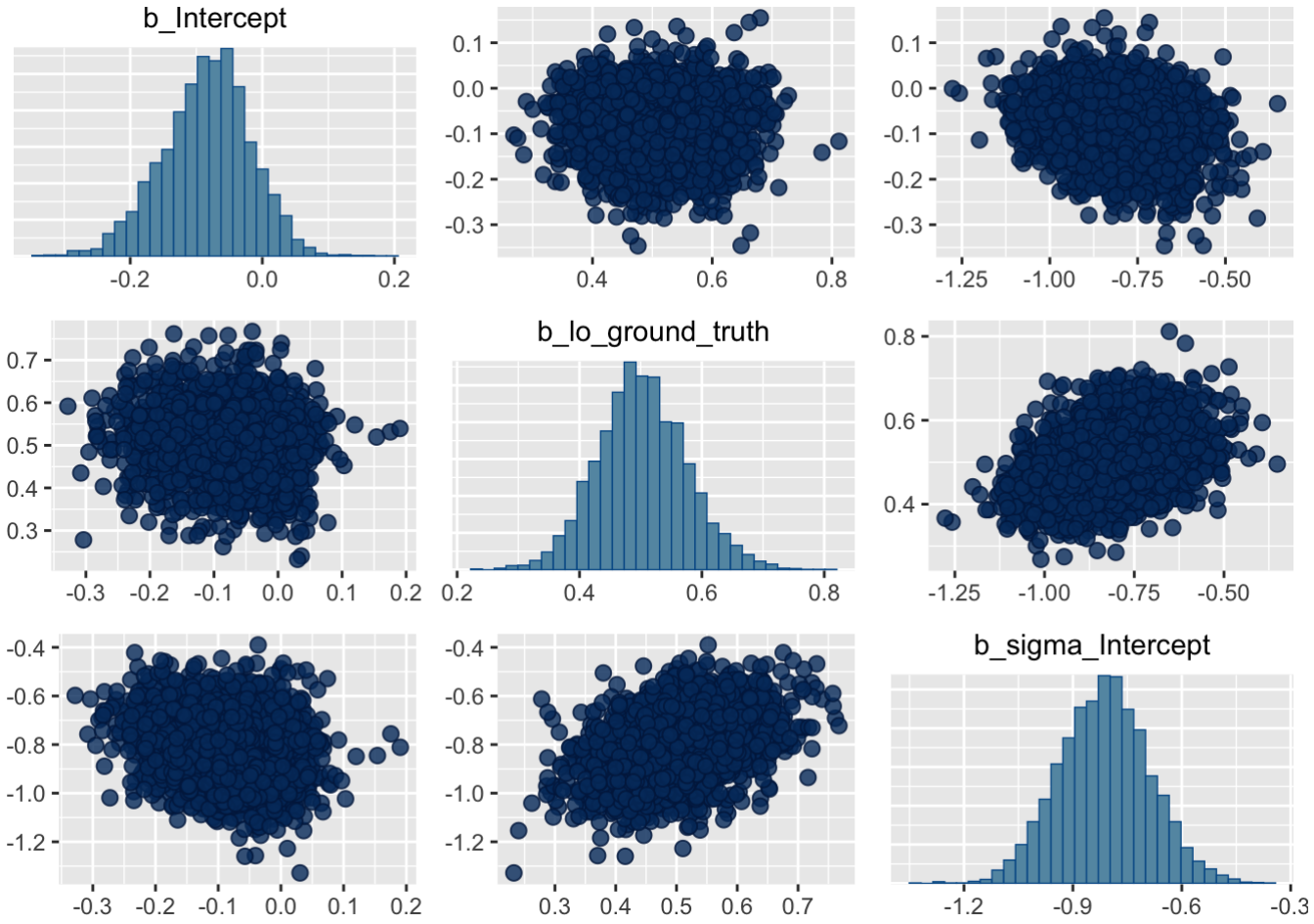
- Trace plots

```
# trace plots
plot(m.wrkr.llo_p_sup)
```

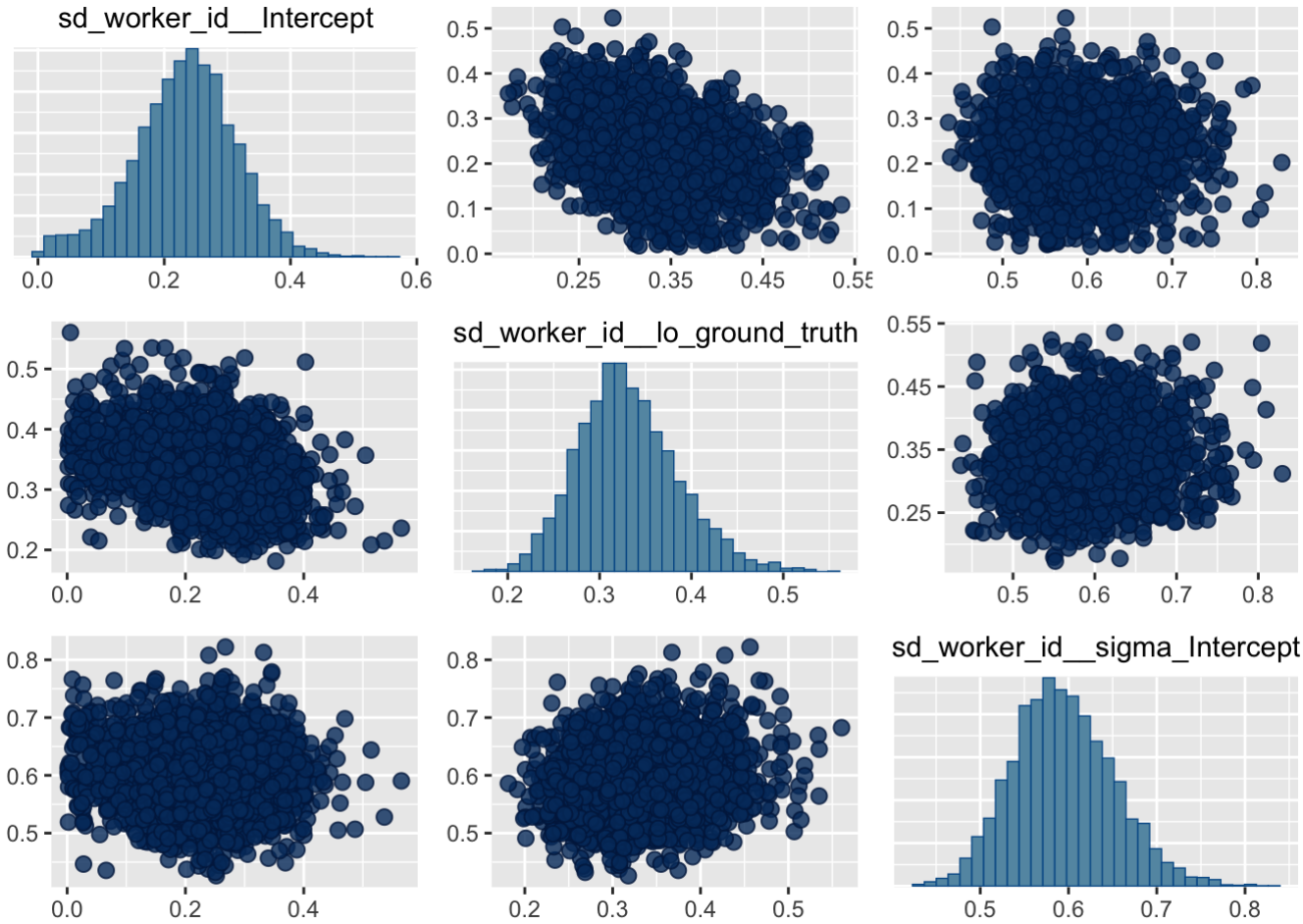

b_Intercept
b_sigma_Intercept
b_lo_ground_truth
sd_worker_id__Intercept
sd_worker_id__lo_ground_truth
sd_worker_id__sigma_Intercept
cor_worker_id__Intercept__lo_ground_truth
cor_worker_id__Intercept__sigma_Intercept
cor_worker_id__lo_ground_truth__sigma_Intercept
Chain
1
2

- Pairs plot

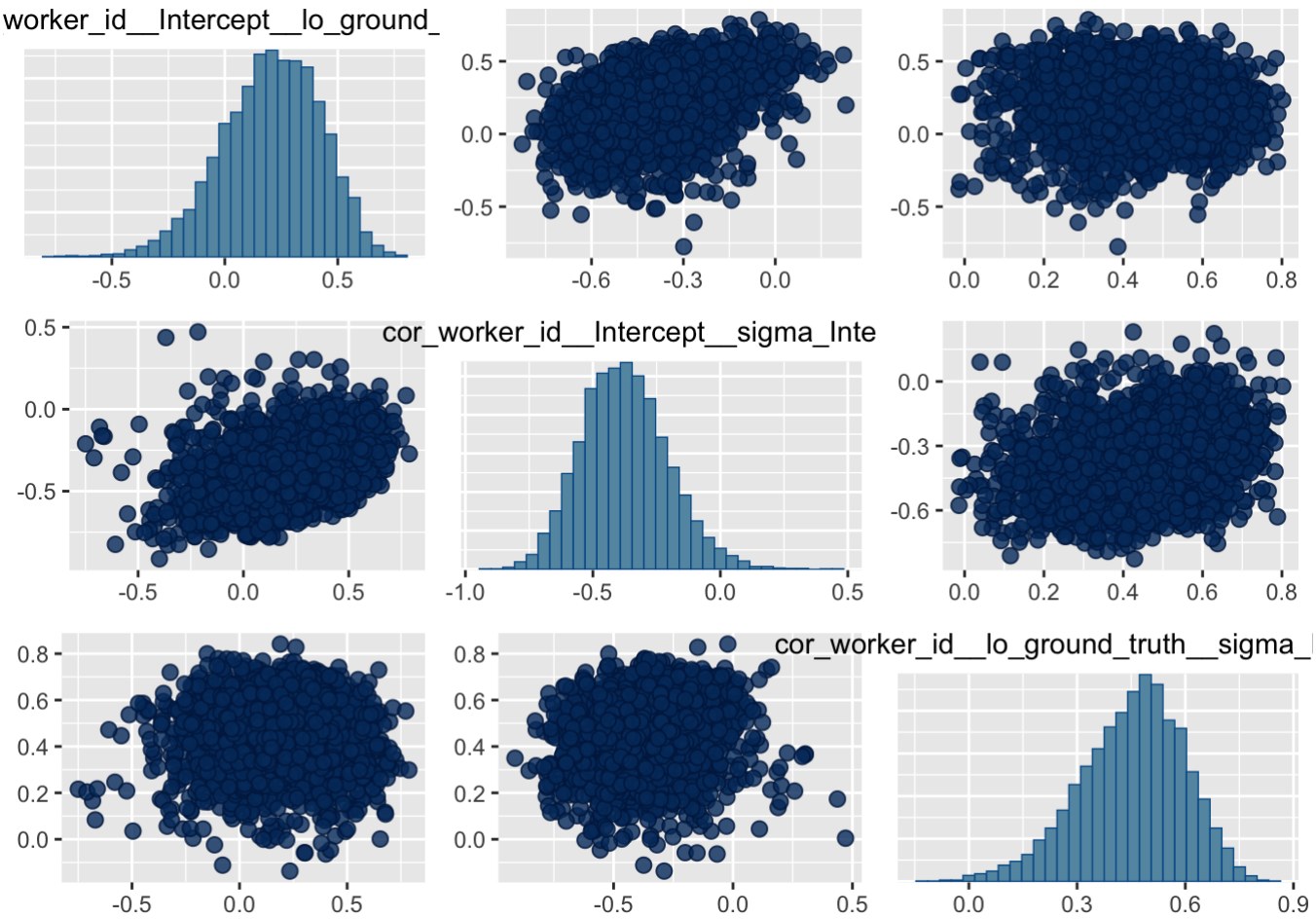
```
# pairs plot (fixed effects)
pairs(m.wrkr.llo_p_sup, exact_match = TRUE, pars = c("b_Intercept", "b_lo_ground_truth",
"b_sigma_Intercept"))
```




```
# pairs plot (random effects)
pairs(m.wrkr.llo_p_sup, exact_match = TRUE, pars = c("sd_worker_id__Intercept", "sd_work
er_id__lo_ground_truth", "sd_worker_id__sigma_Intercept"))
```




```
# pairs plot (random effects covariance)
pairs(m.wrkr.llo_p_sup, pars = c("cor_worker_id__"))
```



- Summary

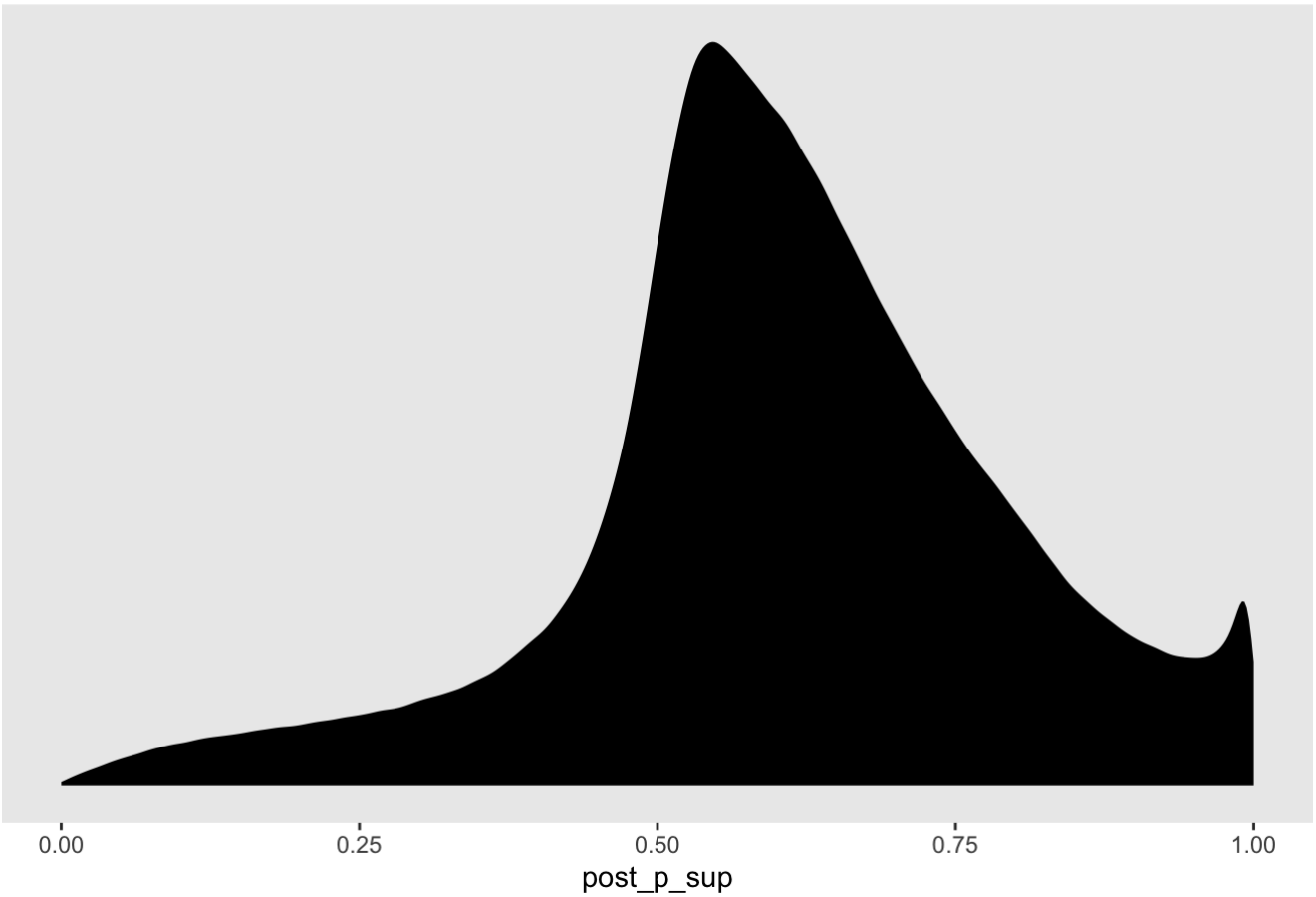
```
# model summary
print(m.wrkr.llo_p_sup)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = log
## Formula: lo_p_sup ~ (1 + lo_ground_truth | sharecor | worker_id) + lo_ground_truth
##          sigma ~ (1 | sharecor | worker_id)
##    Data: model_df (Number of observations: 768)
## Samples: 2 chains, each with iter = 3000; warmup = 500; thin = 1;
##          total post-warmup samples = 5000
##
## Group-Level Effects:
## ~worker_id (Number of levels: 24)
##                                     Estimate Est.Error l-95% CI u-95% CI
## sd(Intercept)                           0.23      0.08     0.05     0.39
## sd(lo_ground_truth)                     0.33      0.05     0.23     0.45
## sd(sigma_Intercept)                     0.59      0.06     0.49     0.71
## cor(Intercept,lo_ground_truth)          0.20      0.22    -0.26     0.58
## cor(Intercept,sigma_Intercept)         -0.37      0.17    -0.68    -0.02
## cor(lo_ground_truth,sigma_Intercept)    0.45      0.14     0.13     0.70
##                                     Eff.Sample Rhat
## sd(Intercept)                              827 1.00
## sd(lo_ground_truth)                       1669 1.00
## sd(sigma_Intercept)                       4301 1.00
## cor(Intercept,lo_ground_truth)            1186 1.00
## cor(Intercept,sigma_Intercept)            2247 1.00
## cor(lo_ground_truth,sigma_Intercept)       1902 1.00
##
## Population-Level Effects:
##                 Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## Intercept          -0.08      0.07    -0.22     0.04       1766 1.00
## sigma_Intercept    -0.81      0.12    -1.06    -0.57       3190 1.00
## lo_ground_truth     0.50      0.07     0.36     0.65       2101 1.00
##
## Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
## is a crude measure of effective sample size, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

Let's check our posterior predictive distribution.

```
# posterior predictive check
model_df %>%
  select(lo_ground_truth, worker_id) %>%
  add_predicted_draws(m.wrkr.llo_p_sup, prediction = "lo_p_sup", seed = 1234) %>%
  mutate(
    # transform to probability units
    post_p_sup = plogis(lo_p_sup)
    ) %>%
  ggplot(aes(x = post_p_sup)) +
  geom_density(fill = "black", size = 0) +
  scale_y_continuous(NULL, breaks = NULL) +
  labs(subtitle = "Posterior predictive distribution for probability of superiority") +
  theme(panel.grid = element_blank())
```

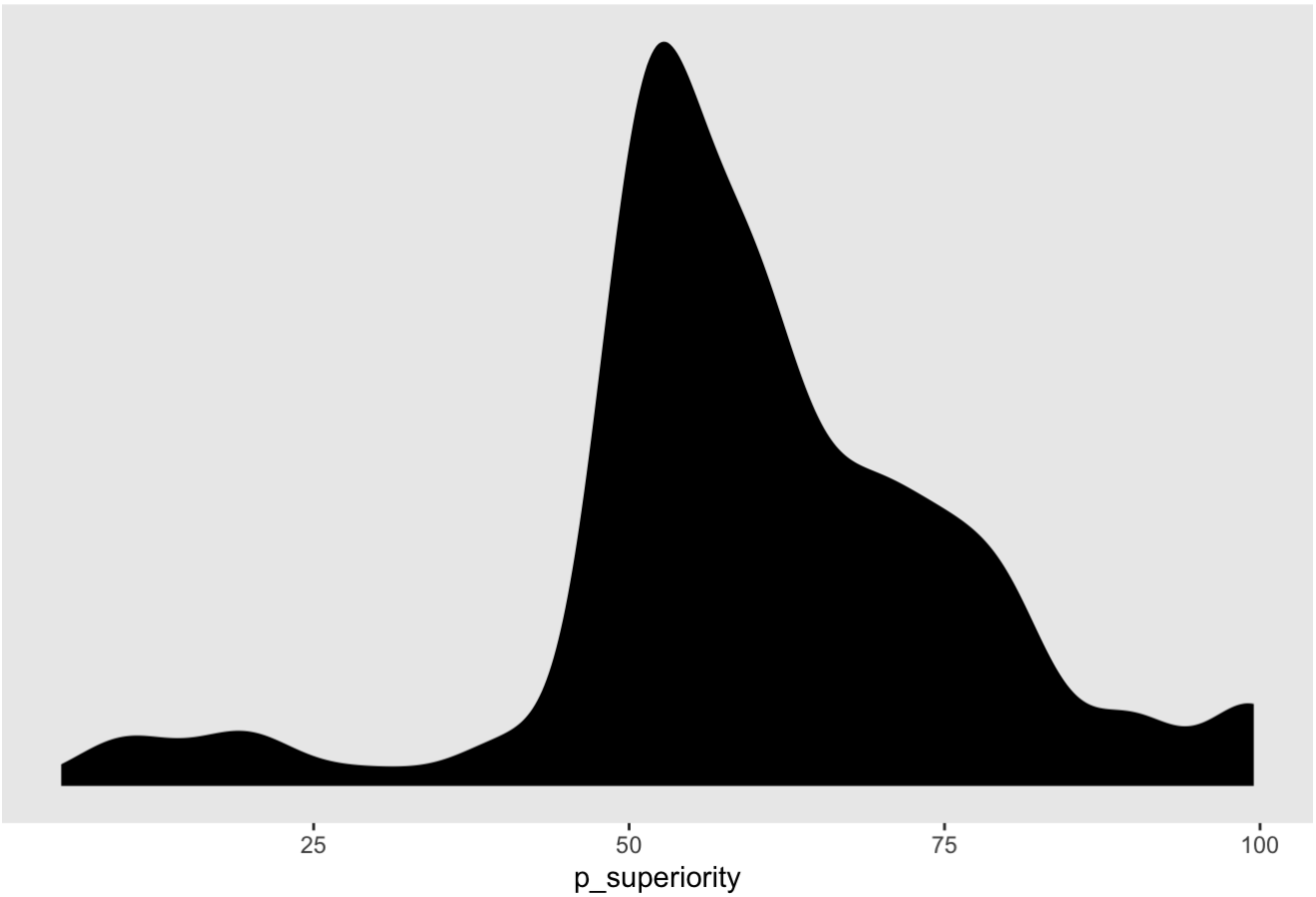
Posterior predictive distribution for probability of superiority



How do these predictions compare to the observed data?

```
# data density
model_df %>%
  ggplot(aes(x = p_superiority)) +
  geom_density(fill = "black", size = 0) +
  scale_y_continuous(NULL, breaks = NULL) +
  labs(subtitle = "Data distribution for probability of superiority") +
  theme(panel.grid = element_blank())
```

Data distribution for probability of superiority



Running a leave one out posterior predictive check, we can see that overall this model has decent predictive validity.

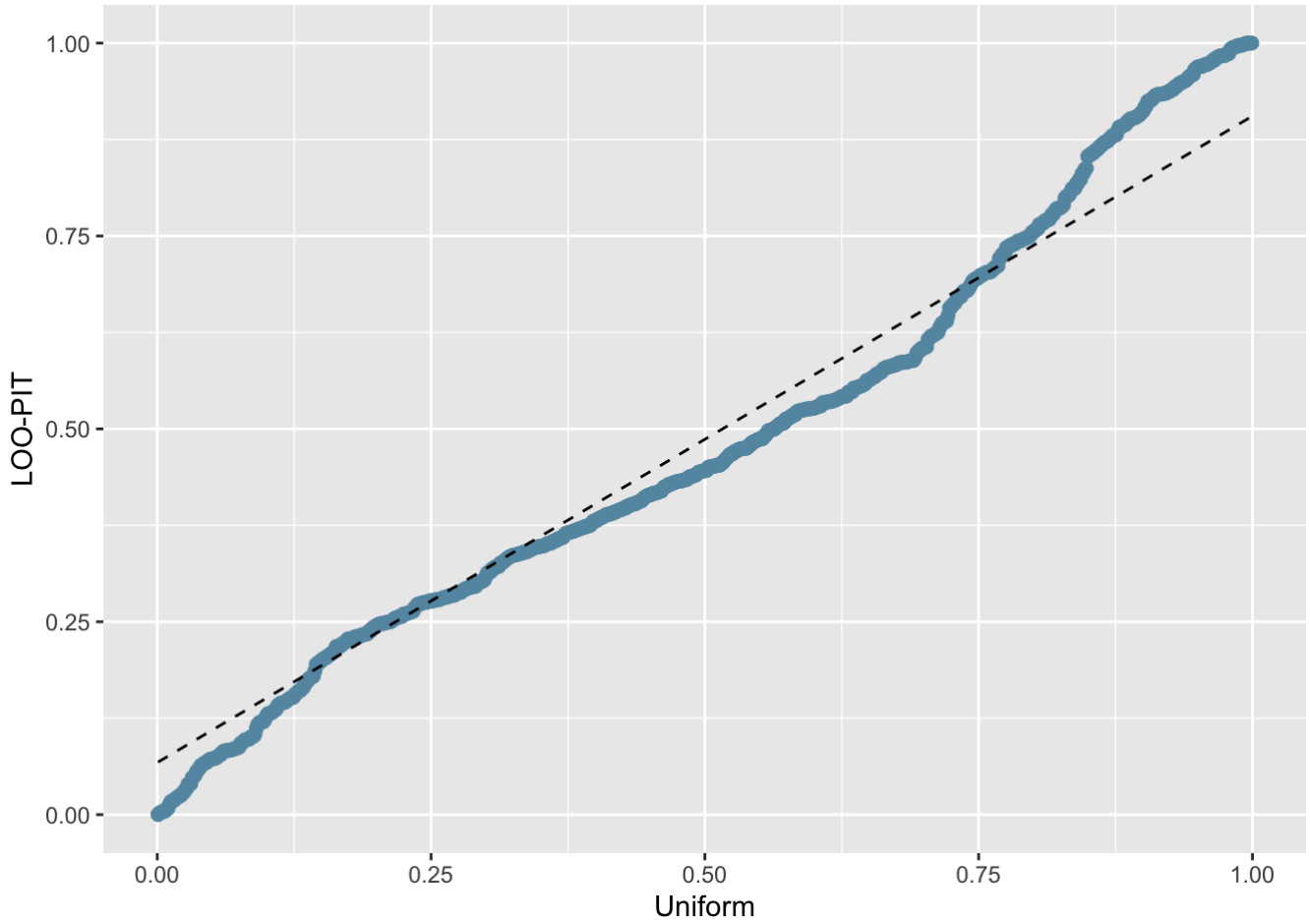
```
# set up data for LOO posterior predictive check
y <- model_df$lo_p_sup
yrep <- posterior_predict(m.wrkr.llo_p_sup)

# run LOO to get weights
loo <- loo(m.wrkr.llo_p_sup, save_psis = TRUE, cores = 2)
```

```
## Warning: Found 3 observations with a pareto_k > 0.7 in model
## 'm.wrkr.llo_p_sup'. It is recommended to set 'reloo = TRUE' in order to
## calculate the ELPD without the assumption that these observations are
## negligible. This will refit the model 3 times to compute the ELPDs for the
## problematic observations directly.
```

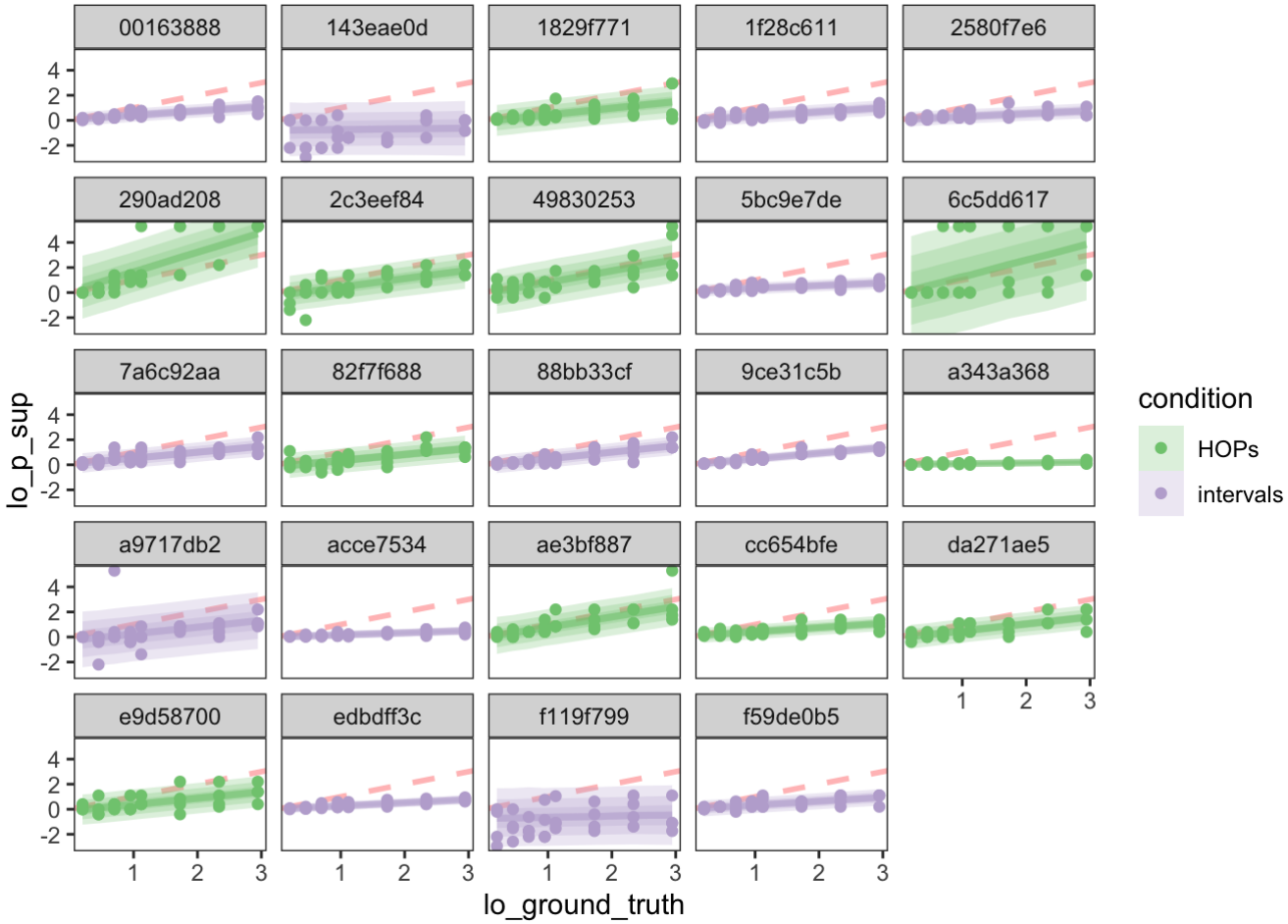
```
psis <- loo$psis_object
lw <- weights(psis)
```

```
ppc_loo_pit_qq(y, yrep, lw = lw)
```



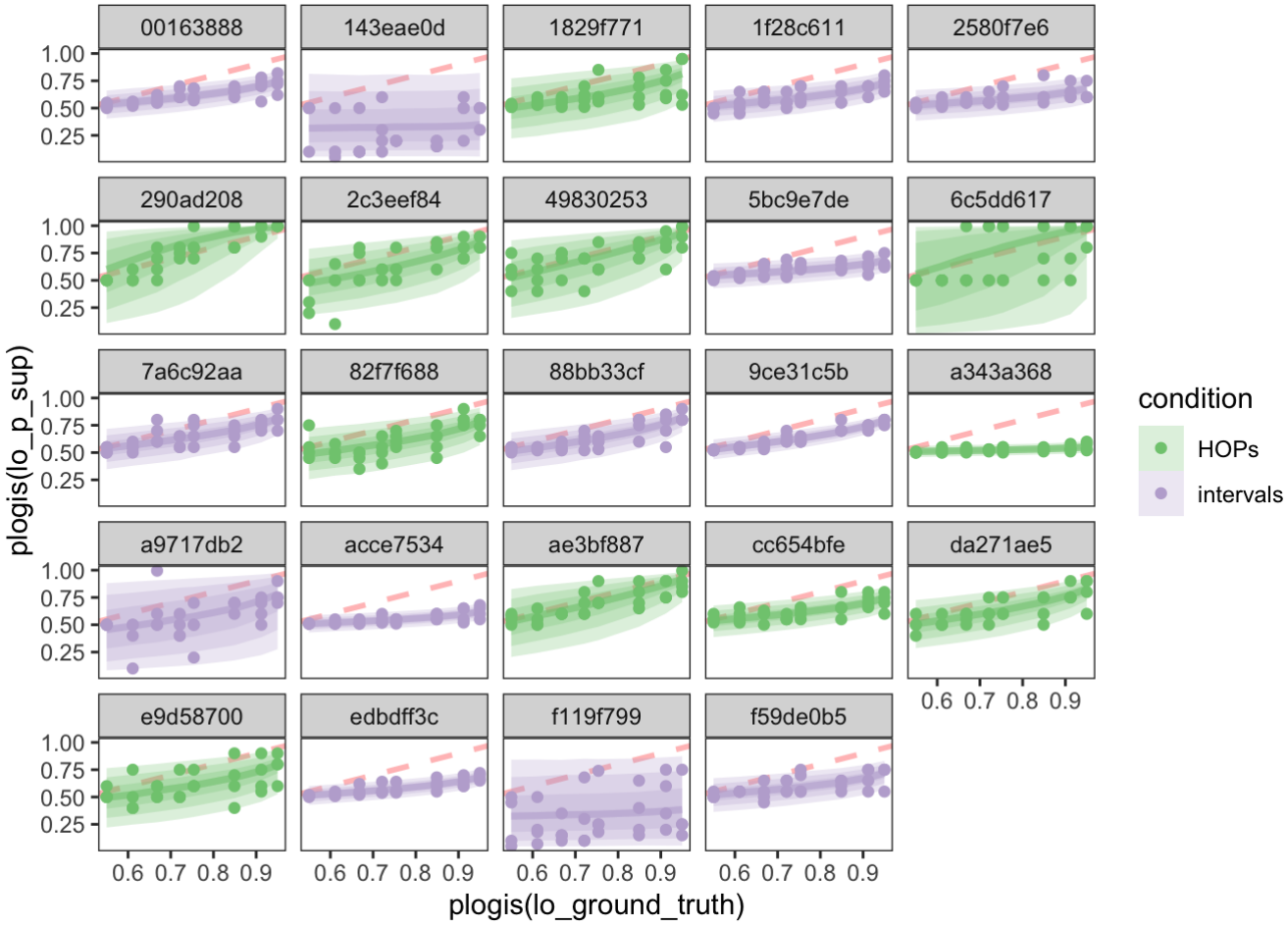
Let's look at posterior predictions per worker to get a more detailed sense of fit quality.

```
model_df %>%
  group_by(lo_ground_truth, worker_id) %>%
  add_predicted_draws(m.wrkr.llo_p_sup) %>%
  ggplot(aes(x = lo_ground_truth, y = lo_p_sup, color = condition, fill = condition)) +
  geom_abline(intercept = 0, slope = 1, size = 1, alpha = .3, color = "red", linetype =
"dashed") + # ground truth
  stat_lineribbon(aes(y = .prediction), .width = c(.95, .80, .50), alpha = .25) +
  geom_point(data = model_df) +
  scale_fill_brewer(type = "qual", palette = 1) +
  scale_color_brewer(type = "qual", palette = 1) +
  coord_cartesian(xlim = quantile(model_df$lo_ground_truth, c(0, 1)),
                  ylim = quantile(model_df$lo_p_sup, c(0, 1))) +
  theme_bw() +
  theme(panel.grid = element_blank()) +
  facet_wrap(~ worker_id)
```



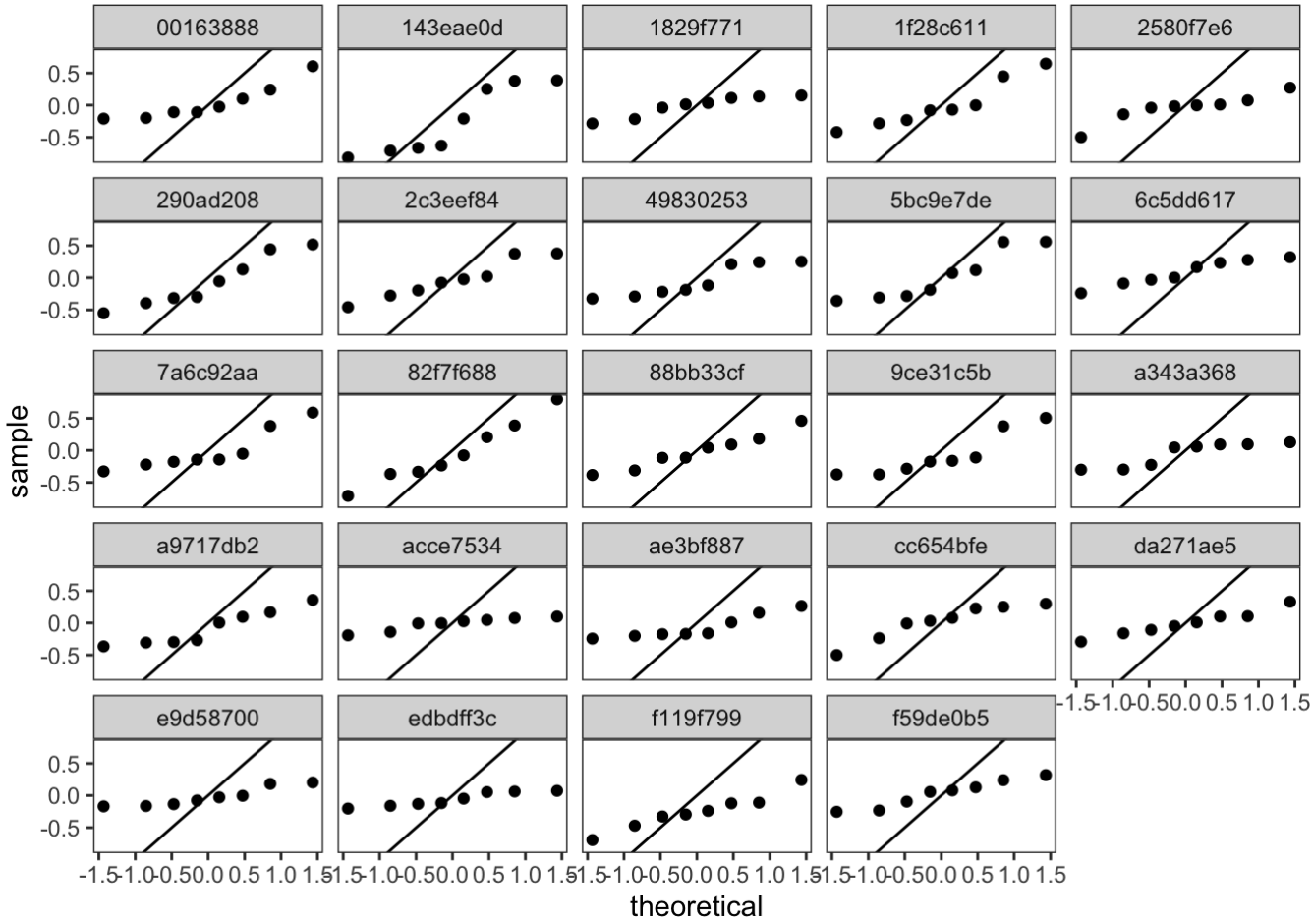
What does this look like in probability units?

```
model_df %>%
  group_by(lo_ground_truth, worker_id) %>%
  add_predicted_draws(m.wrkr.llo_p_sup) %>%
  ggplot(aes(x = plogis(lo_ground_truth), y = plogis(lo_p_sup), color = condition, fill
= condition)) +
  geom_abline(intercept = 0, slope = 1, size = 1, alpha = .3, color = "red", linetype =
"dashed") + # ground truth
  stat_lineribbon(aes(y = plogis(.prediction)), .width = c(.95, .80, .50), alpha = .25)
 +
  geom_point(data = model_df) +
  scale_fill_brewer(type = "qual", palette = 1) +
  scale_color_brewer(type = "qual", palette = 1) +
  coord_cartesian(xlim = quantile(plogis(model_df$lo_ground_truth), c(0, 1)),
                  ylim = quantile(plogis(model_df$lo_p_sup), c(0, 1))) +
  theme_bw() +
  theme(panel.grid = element_blank()) +
  facet_wrap(~ worker_id)
```



One thing we're trying to gage here is whether our model has predictive validity at the level of each worker. To examine this more closely we'll look at QQ plots for residuals at the worker level.

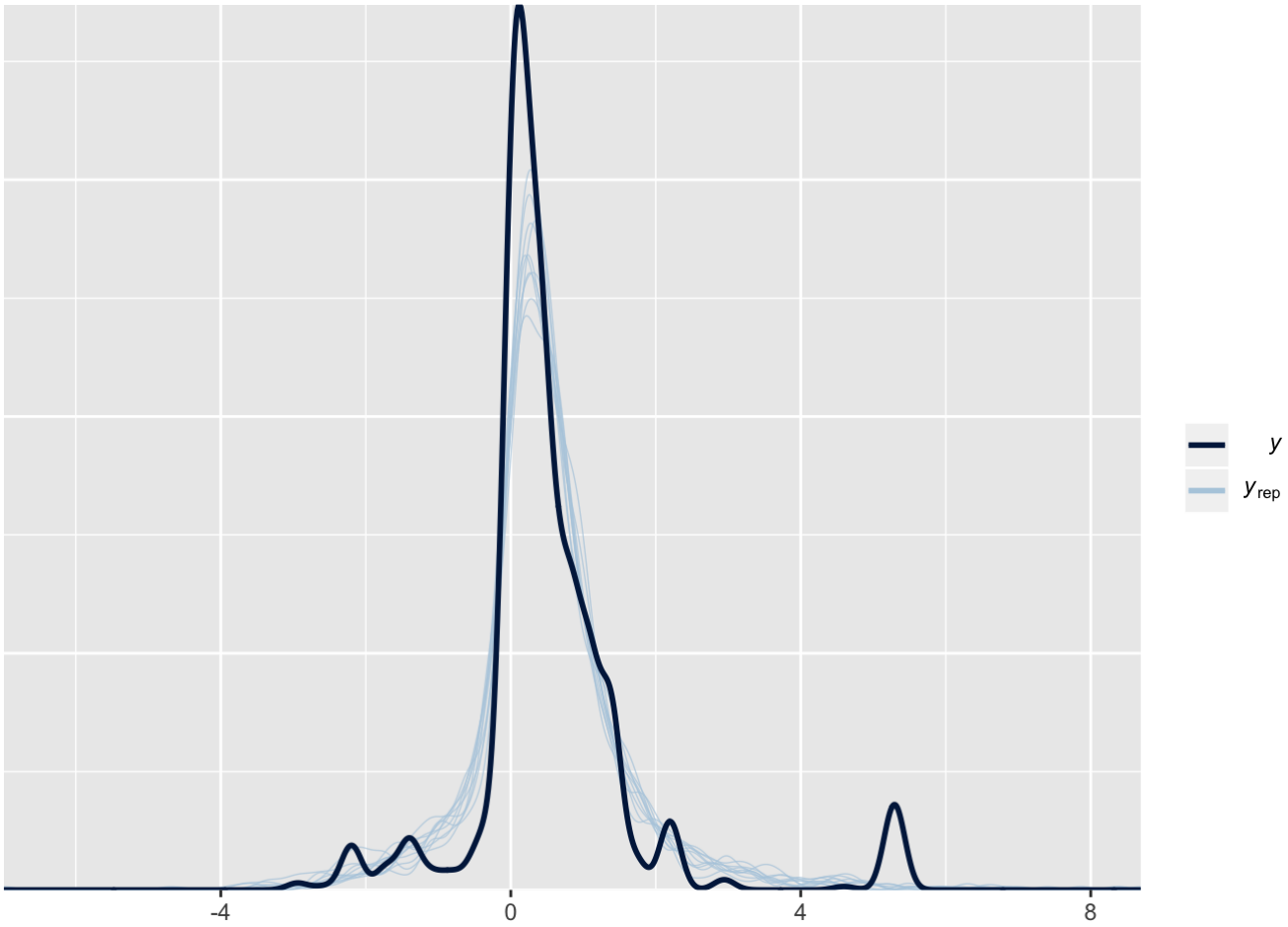
```
model_df %>%
  add_predicted_draws(m.wrkr.llo_p_sup) %>%
  group_by(lo_ground_truth, worker_id) %>%
  summarise(
    p_residual = mean(.prediction < lo_p_sup), # what proportion of predicted judgments
 are less than the observed response?
    z_residual = qnorm(p_residual)             # what are the z-scores of these cumulati
ve probabilities?
  ) %>%
  ggplot(aes(sample = z_residual)) +
  geom_qq() +
  geom_abline() +
  theme_bw() +
  theme(panel.grid = element_blank()) +
  facet_wrap(~ worker_id)
```



These don't look great. We can see that there is some clustering of responses, probably reflecting a preference for round numbers on the response scale.

```
pp_check(m.wrkr.llo_p_sup)
```

```
## Using 10 posterior samples for ppc type 'dens_overlay' by default.
```



As long as the location and scale of the predictions look reasonably in line with the empirical data (which they do), we don't really care too much if the model doesn't predict every small anomally. This plot showing predictive densities alongside the observed data is resassuring insofar as we are doing a decent job of modeling the things we care about.

Let's see if our predictive validity improves at the worker level when we add our experimental manipulations as predictors.

Add Predictors to Answer Research Questions

In order to answer our research questions, we need to account for the interaction of the ground truth with whether means are present vs absent, whether visualized uncertainty is high vs low, and what uncertainty visualization condition a user was assigned to. We'll add predictors for each of these factors to our hierarchical model in turn.

Presence/Absence of the Mean

Our primary research question is how the presence of the mean impacts the slopes of linear models in log odds space. To test this, we'll add an interaction between the presence of the mean and the ground truth.

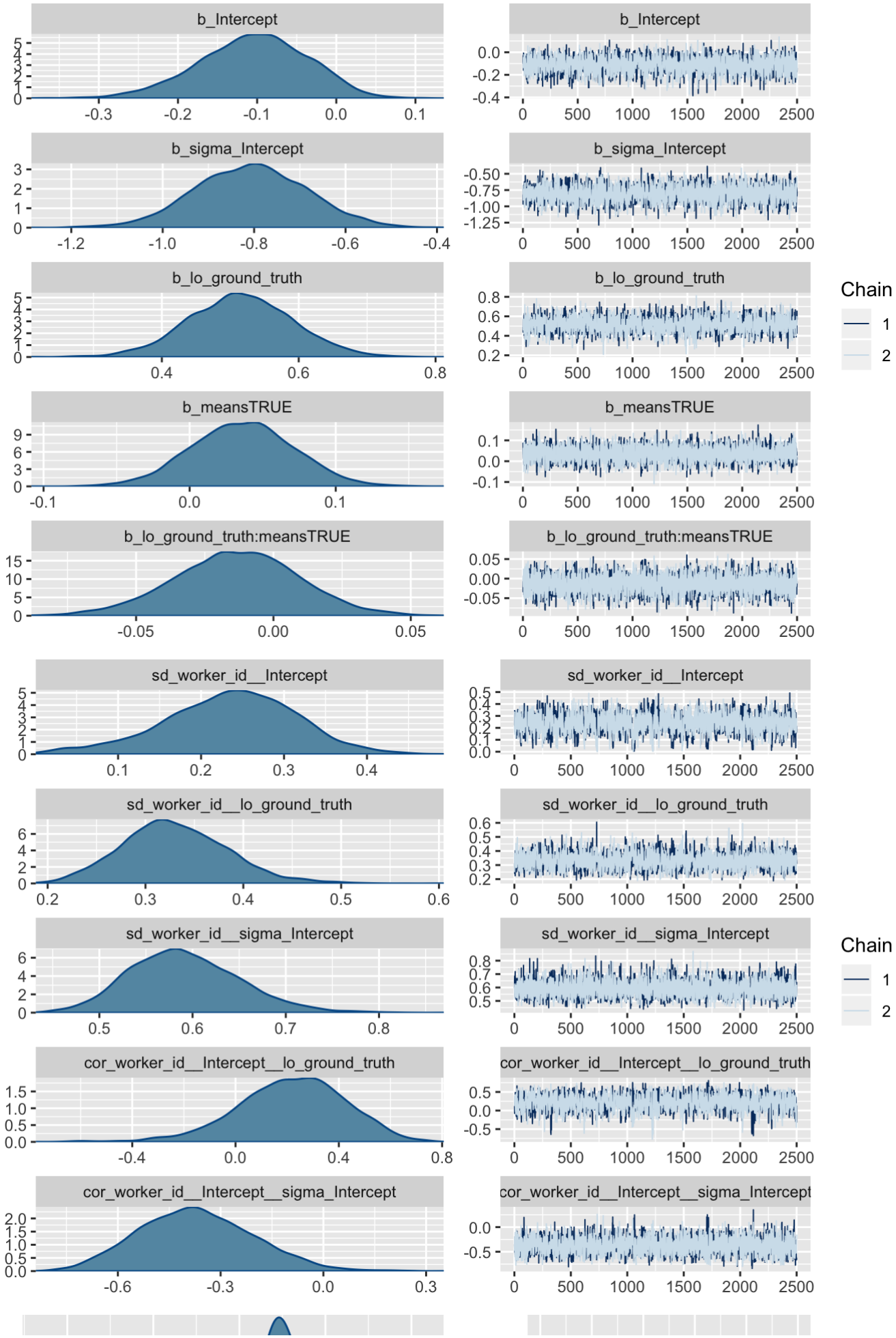
We use the same priors as we did for the previous model. Now, let's fit the model to our data.

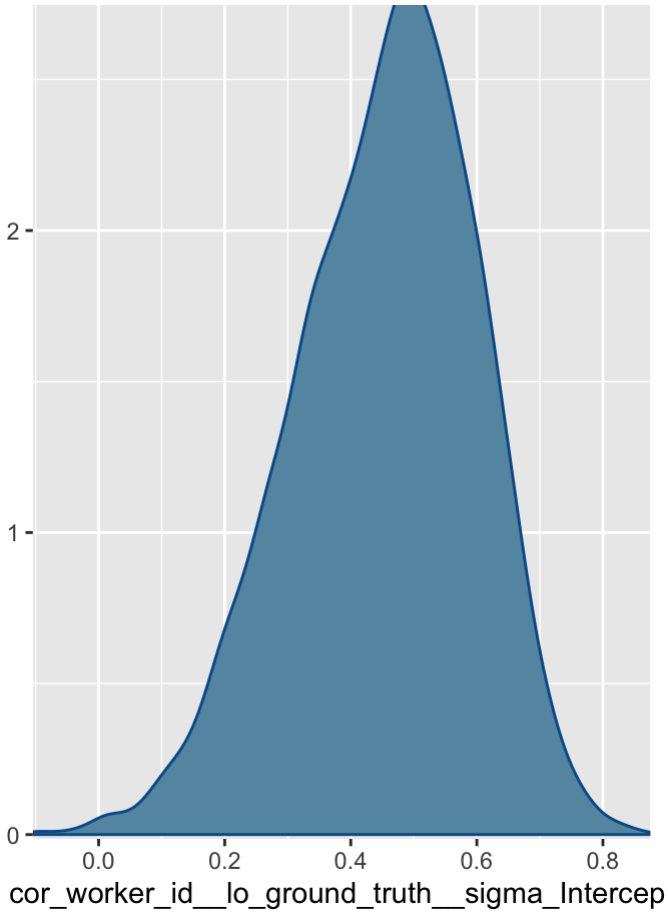
```
# hierarchical LLO model with fixed effects on slope and residual variance conditioned o
n the presence/absence of the mean
m.wrkr.means.llo_p_sup <- brm(data = model_df, family = "gaussian",
                              formula = bf(lo_p_sup ~  (1 + lo_ground_truth|sharecor|wor
ker_id) + lo_ground_truth*means,
                                           sigma ~ (1|sharecor|worker_id)),
                              prior = c(prior(normal(1, 0.5), class = b),
                                        prior(normal(1.3, 1), class = Intercept),
                                        prior(normal(0, 0.15), class = sd, group = worke
r_id),
                                        prior(normal(0, 0.15), class = sd, dpar = sigm
a),
                                        prior(lkj(4), class = cor)),
                              iter = 3000, warmup = 500, chains = 2, cores = 2,
                              control = list(adapt_delta = 0.99, max_treedepth = 12),
                              file = "model-fits/llo_mdl-wrkr_means")
```

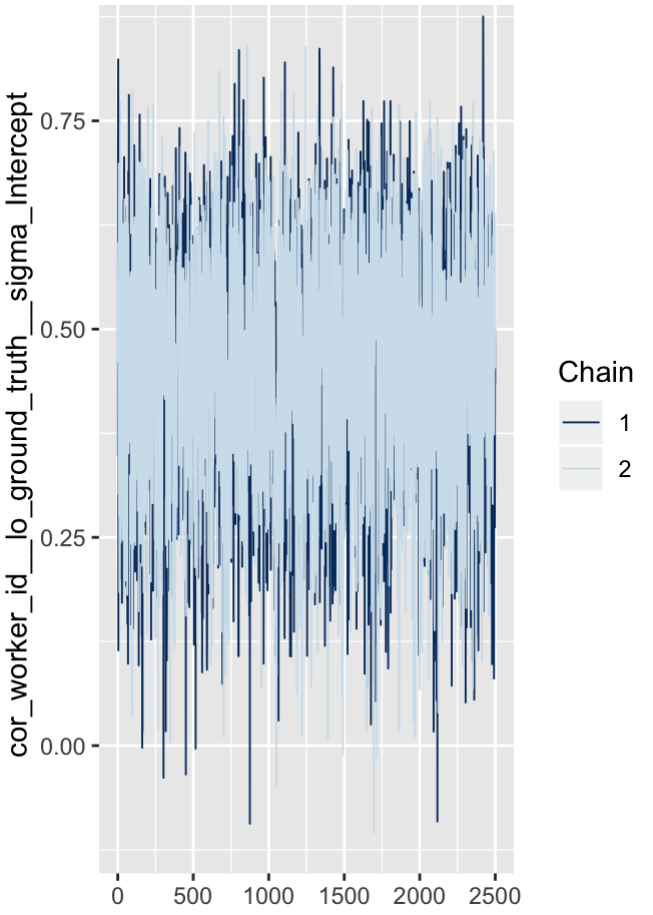
Check diagnostics:

- Trace plots

```
# trace plots
plot(m.wrkr.means.llo_p_sup)
```

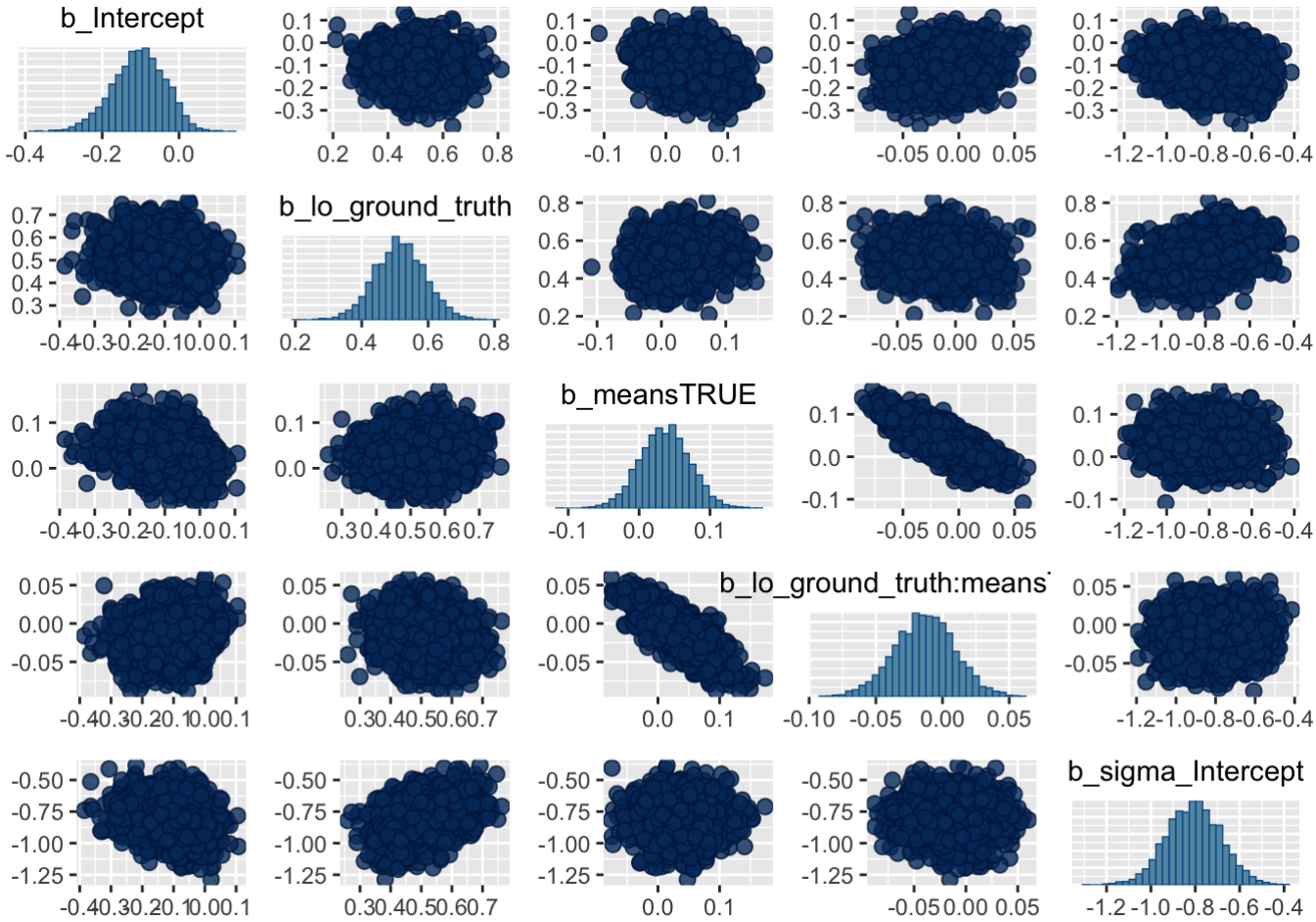

b_Intercept
b_sigma_Intercept
b_lo_ground_truth
b_meansTRUE
b_lo_ground_truth:meansTRUE
sd_worker_id__Intercept
sd_worker_id__lo_ground_truth
sd_worker_id__sigma_Intercept
cor_worker_id__Intercept__lo_ground_truth
cor_worker_id__Intercept__sigma_Intercept
Chain
1
2



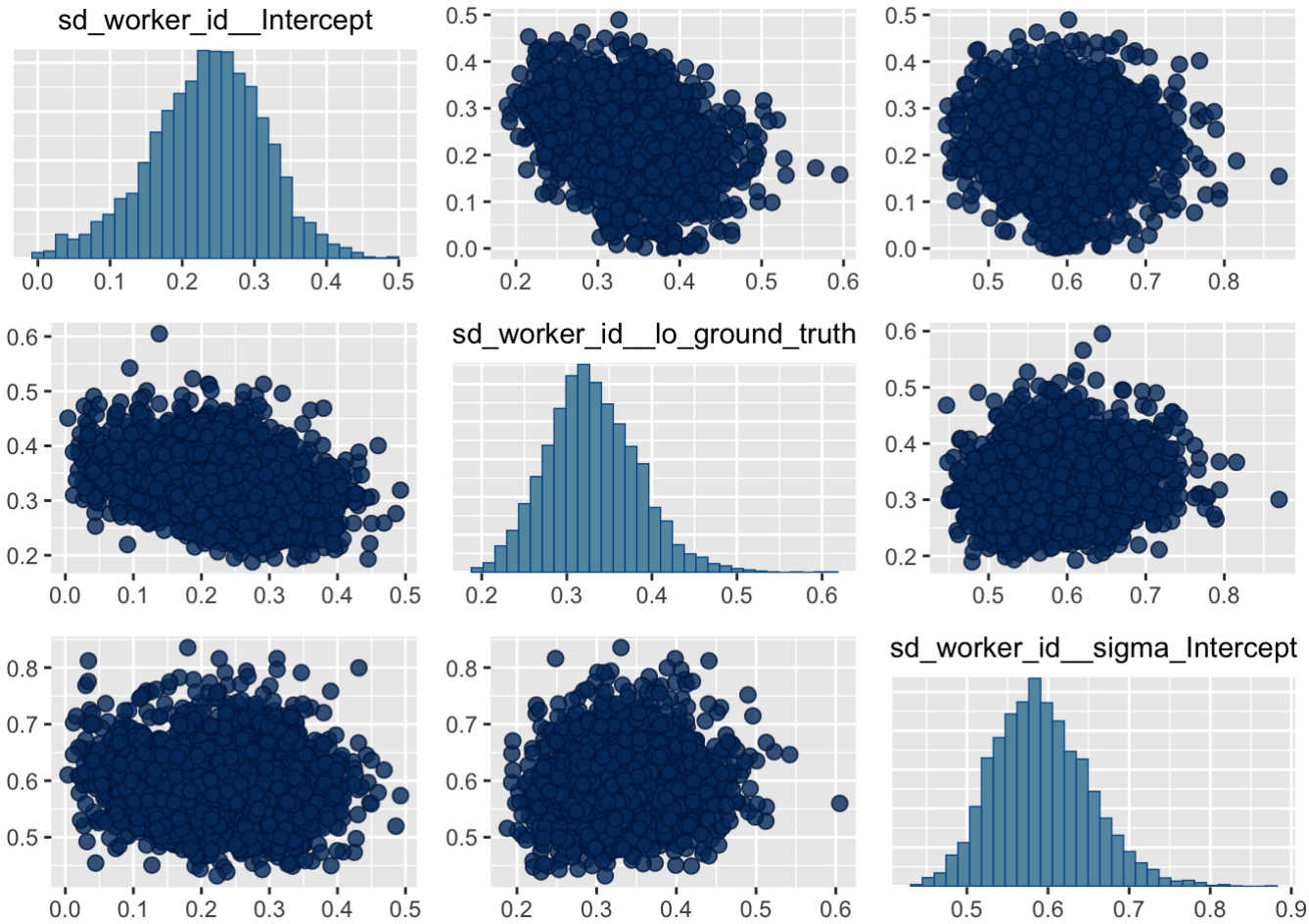


- Pairs plot

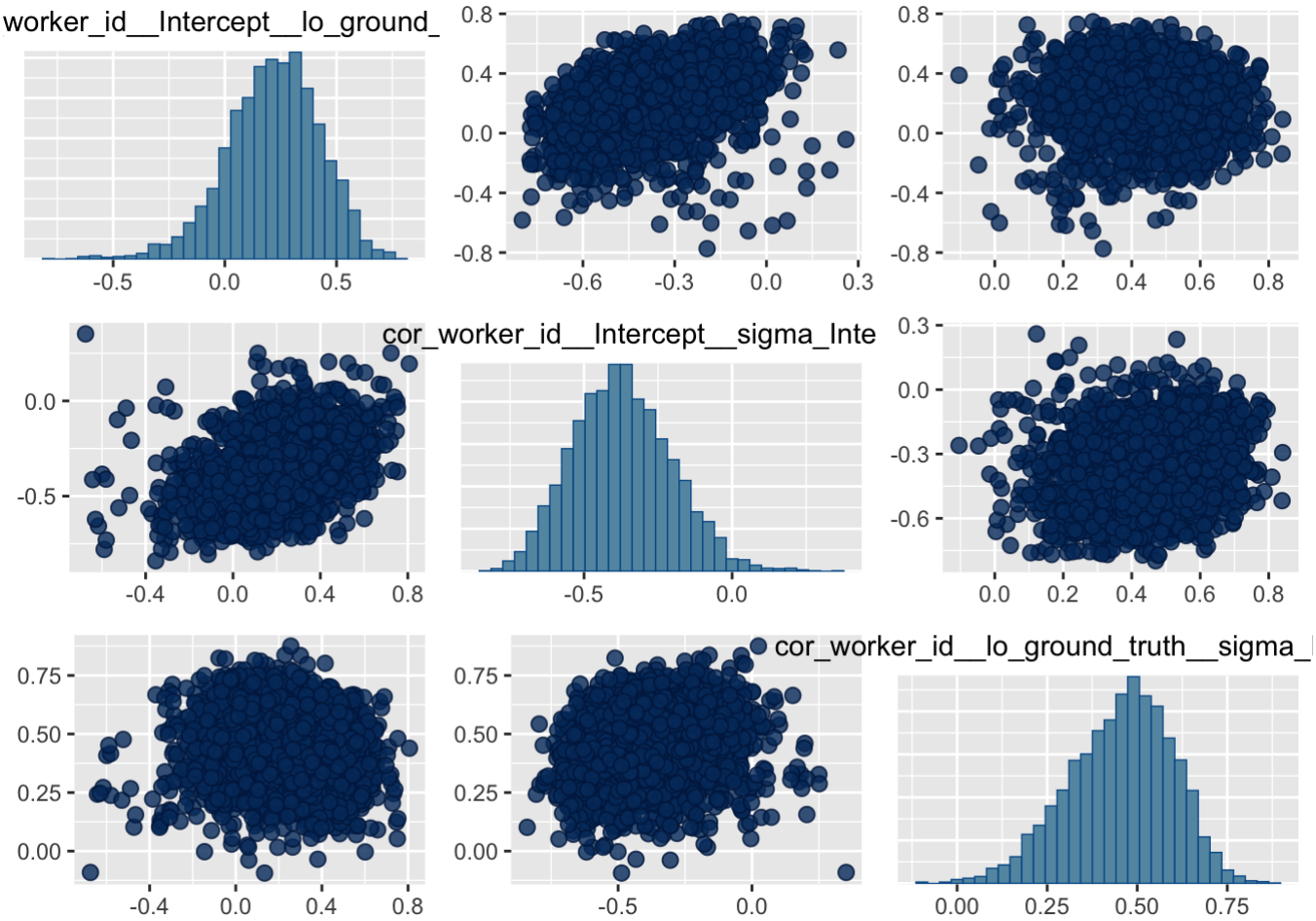
```
# pairs plot (fixed effects)
pairs(m.wrkr.means.llo_p_sup, exact_match = TRUE, pars = c("b_Intercept",
                                                           "b_lo_ground_truth",
                                                           "b_meansTRUE",
                                                           "b_lo_ground_truth:meansTRUE"
,
                                                           "b_sigma_Intercept"))
```




```
# pairs plot (random effects)
pairs(m.wrkr.means.llo_p_sup, exact_match = TRUE, pars = c("sd_worker_id__Intercept",
                                                           "sd_worker_id__lo_ground_trut
h",
                                                           "sd_worker_id__sigma_Intercep
t"))
```




```
# pairs plot (covariance matrix)
pairs(m.wrkr.means.llo_p_sup, pars = c("cor_worker_id__"))
```



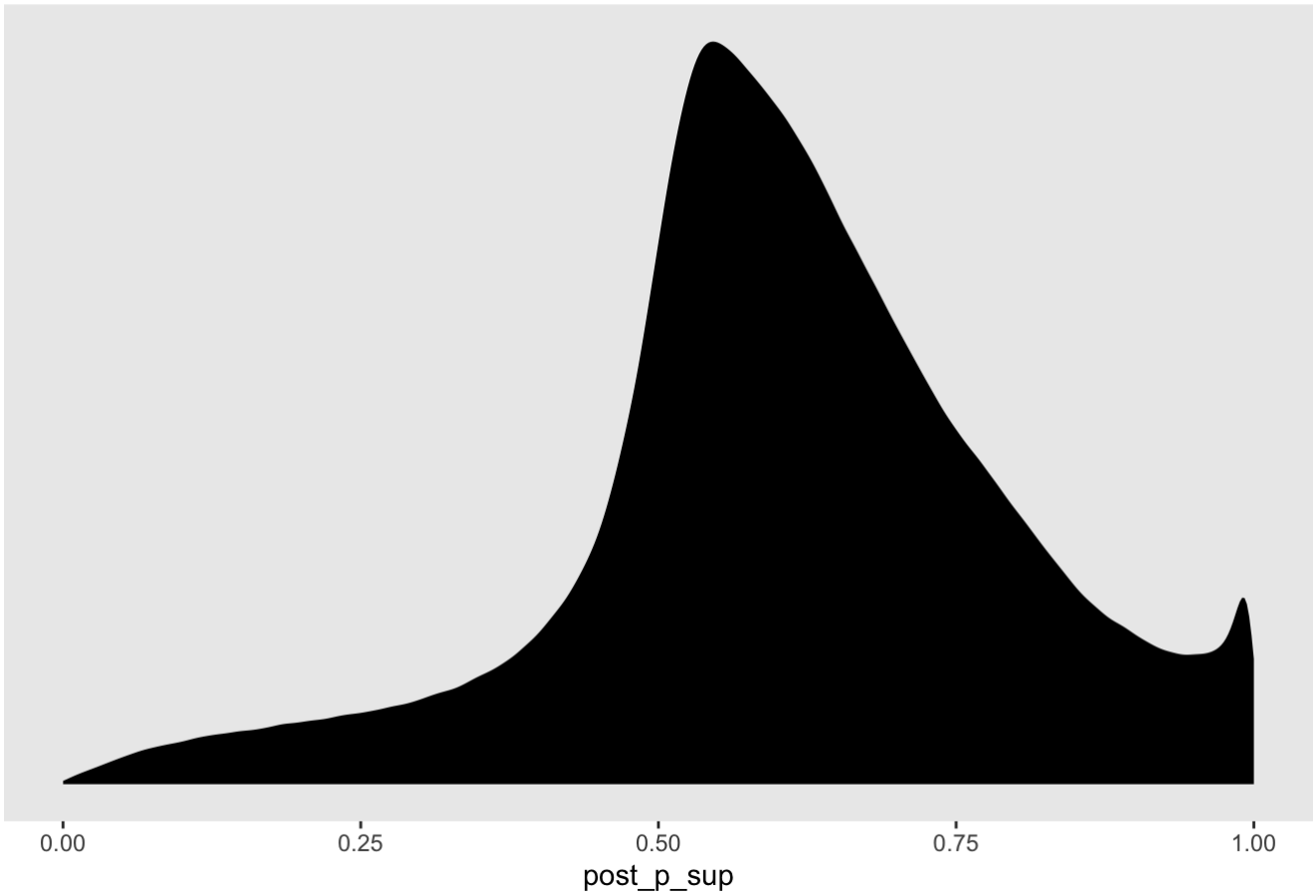
- Summary

```
# model summary
print(m.wrkr.means.llo_p_sup)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = log
## Formula: lo_p_sup ~ (1 + lo_ground_truth | sharecor | worker_id) + lo_ground_truth *
means
##          sigma ~ (1 | sharecor | worker_id)
##    Data: model_df (Number of observations: 768)
## Samples: 2 chains, each with iter = 3000; warmup = 500; thin = 1;
##          total post-warmup samples = 5000
##
## Group-Level Effects:
## ~worker_id (Number of levels: 24)
##                                     Estimate Est.Error l-95% CI u-95% CI
## sd(Intercept)                           0.23      0.08     0.06     0.38
## sd(lo_ground_truth)                     0.33      0.05     0.23     0.45
## sd(sigma_Intercept)                     0.59      0.06     0.49     0.72
## cor(Intercept,lo_ground_truth)          0.22      0.21    -0.23     0.59
## cor(Intercept,sigma_Intercept)         -0.37      0.17    -0.67    -0.04
## cor(lo_ground_truth,sigma_Intercept)    0.45      0.14     0.15     0.70
##                                     Eff.Sample Rhat
## sd(Intercept)                              848 1.00
## sd(lo_ground_truth)                       2341 1.00
## sd(sigma_Intercept)                       4364 1.00
## cor(Intercept,lo_ground_truth)            1213 1.00
## cor(Intercept,sigma_Intercept)            2340 1.00
## cor(lo_ground_truth,sigma_Intercept)       2131 1.00
##
## Population-Level Effects:
##                           Estimate Est.Error l-95% CI u-95% CI Eff.Sample
## Intercept                    -0.10      0.07    -0.25     0.02       1896
## sigma_Intercept              -0.81      0.12    -1.04    -0.57       2912
## lo_ground_truth               0.52      0.08     0.36     0.67       1884
## meansTRUE                     0.04      0.04    -0.03     0.10       8058
## lo_ground_truth:meansTRUE    -0.01      0.02    -0.06     0.03       8284
##                           Rhat
## Intercept                 1.00
## sigma_Intercept           1.00
## lo_ground_truth           1.00
## meansTRUE                 1.00
## lo_ground_truth:meansTRUE 1.00
##
## Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
## is a crude measure of effective sample size, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

Let's check our posterior predictive distribution.

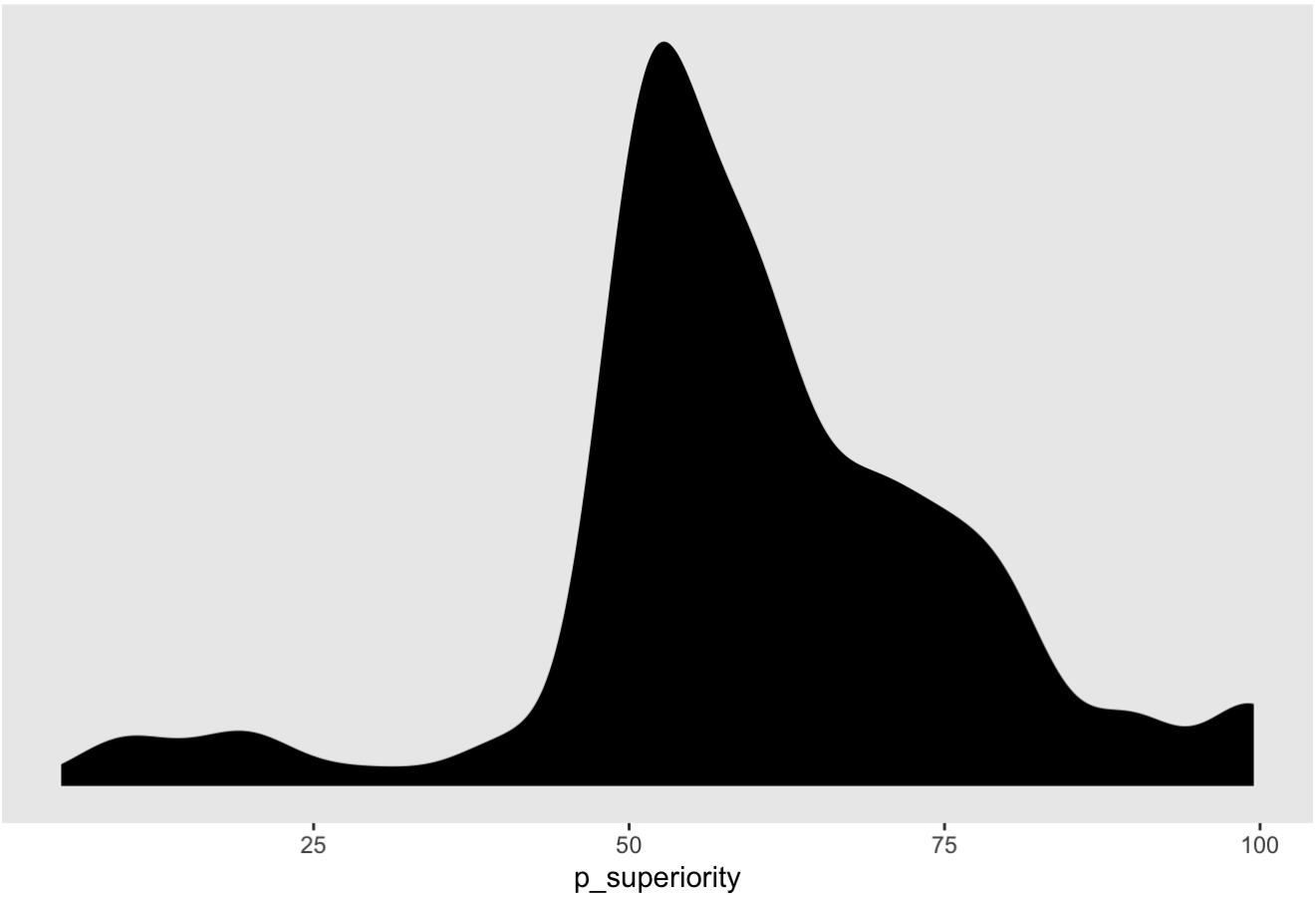
```
# posterior predictive check
model_df %>%
  select(lo_ground_truth, worker_id, means) %>%
  add_predicted_draws(m.wrkr.means.llo_p_sup, prediction = "lo_p_sup", seed = 1234) %>%
  mutate(
    # transform to probability units
    post_p_sup = plogis(lo_p_sup)
  ) %>%
  ggplot(aes(x = post_p_sup)) +
  geom_density(fill = "black", size = 0) +
  scale_y_continuous(NULL, breaks = NULL) +
  labs(subtitle = "Posterior predictive distribution for probability of superiority") +
  theme(panel.grid = element_blank())
```



How do these predictions compare to the observed data?

```
# data density
model_df %>%
  ggplot(aes(x = p_superiority)) +
  geom_density(fill = "black", size = 0) +
  scale_y_continuous(NULL, breaks = NULL) +
  labs(subtitle = "Data distribution for probability of superiority") +
  theme(panel.grid = element_blank())
```

Data distribution for probability of superiority



Running a leave one out posterior predictive check, we can see that overall this model has decent predictive validity.

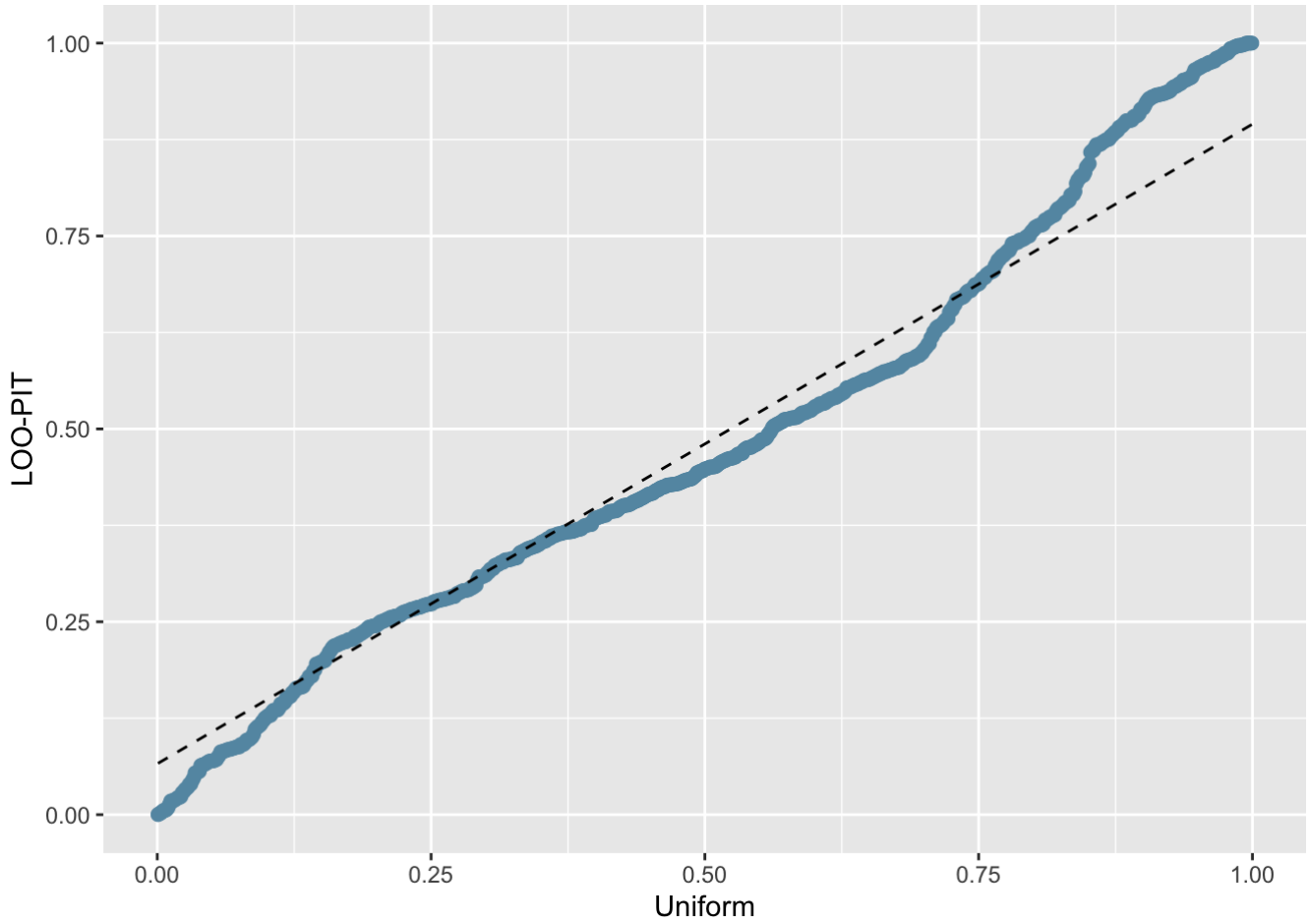
```
# set up data for LOO posterior predictive check
y <- model_df$lo_p_sup
yrep <- posterior_predict(m.wrkr.means.llo_p_sup)

# run LOO to get weights
loo <- loo(m.wrkr.means.llo_p_sup, save_psis = TRUE, cores = 2)
```

```
## Warning: Found 4 observations with a pareto_k > 0.7 in model
## 'm.wrkr.means.llo_p_sup'. It is recommended to set 'reloo = TRUE' in order
## to calculate the ELPD without the assumption that these observations are
## negligible. This will refit the model 4 times to compute the ELPDs for the
## problematic observations directly.
```

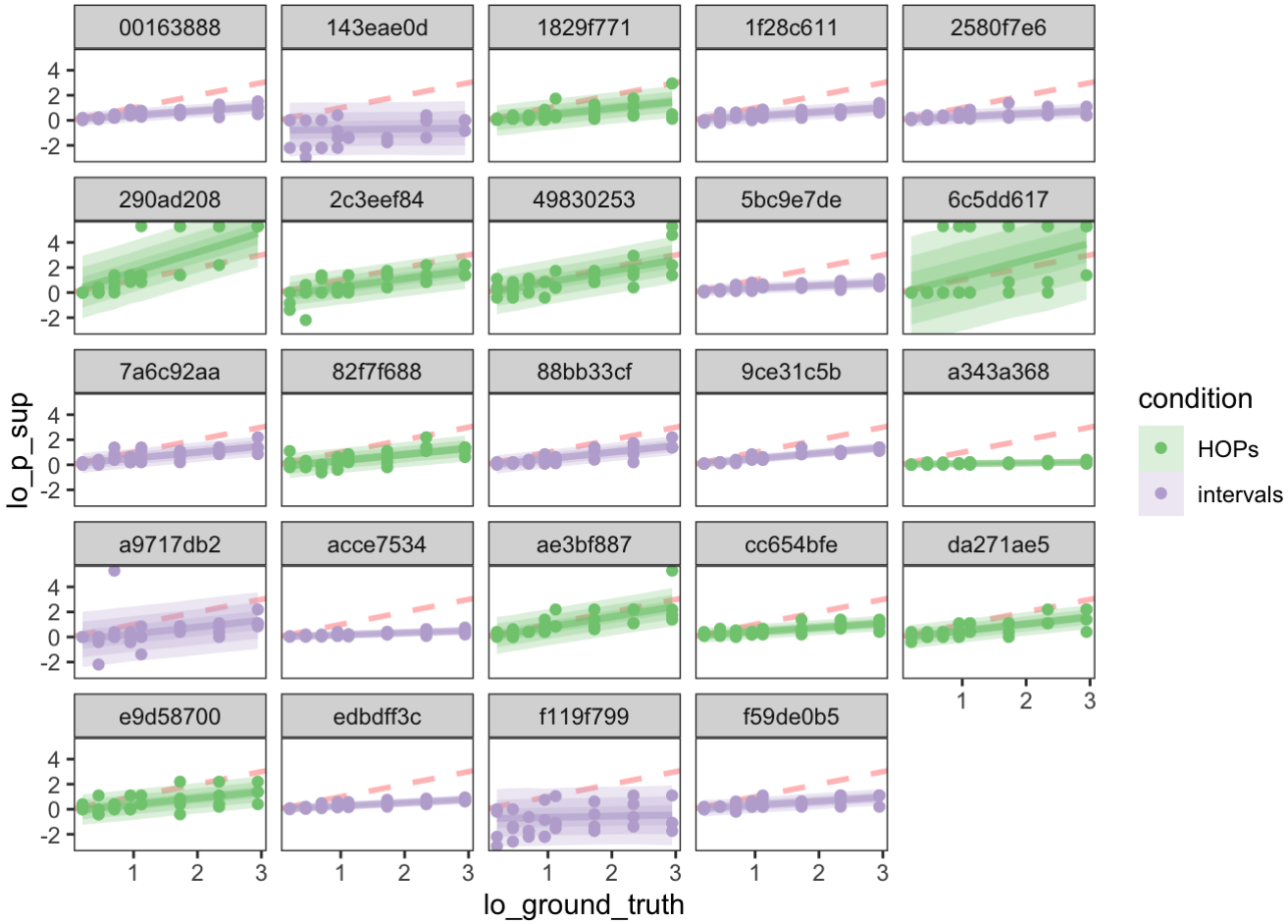
```
psis <- loo$psis_object
lw <- weights(psis)
```

```
ppc_loo_pit_qq(y, yrep, lw = lw)
```



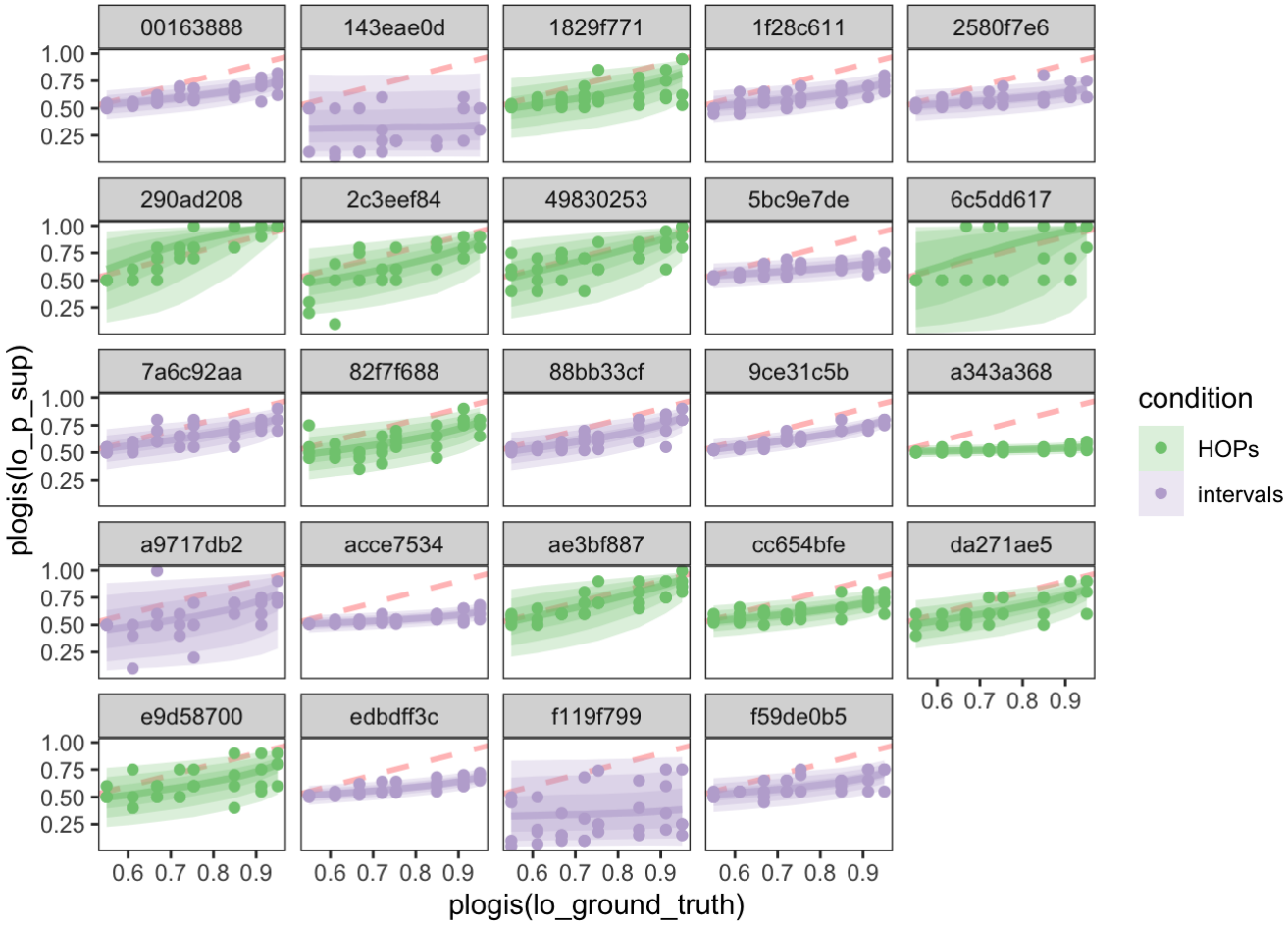
Let's take a look at predictions per worker and visualization condition to get a more granular sense of our model fit.

```
model_df %>%
  group_by(lo_ground_truth, worker_id, means) %>%
  add_predicted_draws(m.wrkr.means.llo_p_sup) %>%
  ggplot(aes(x = lo_ground_truth, y = lo_p_sup, color = condition, fill = condition)) +
  geom_abline(intercept = 0, slope = 1, size = 1, alpha = .3, color = "red", linetype =
"dashed") + # ground truth
  stat_lineribbon(aes(y = .prediction), .width = c(.95, .80, .50), alpha = .25) +
  geom_point(data = model_df) +
  scale_fill_brewer(type = "qual", palette = 1) +
  scale_color_brewer(type = "qual", palette = 1) +
  coord_cartesian(xlim = quantile(model_df$lo_ground_truth, c(0, 1)),
                  ylim = quantile(model_df$lo_p_sup, c(0, 1))) +
  theme_bw() +
  theme(panel.grid = element_blank()) +
  facet_wrap(~ worker_id)
```



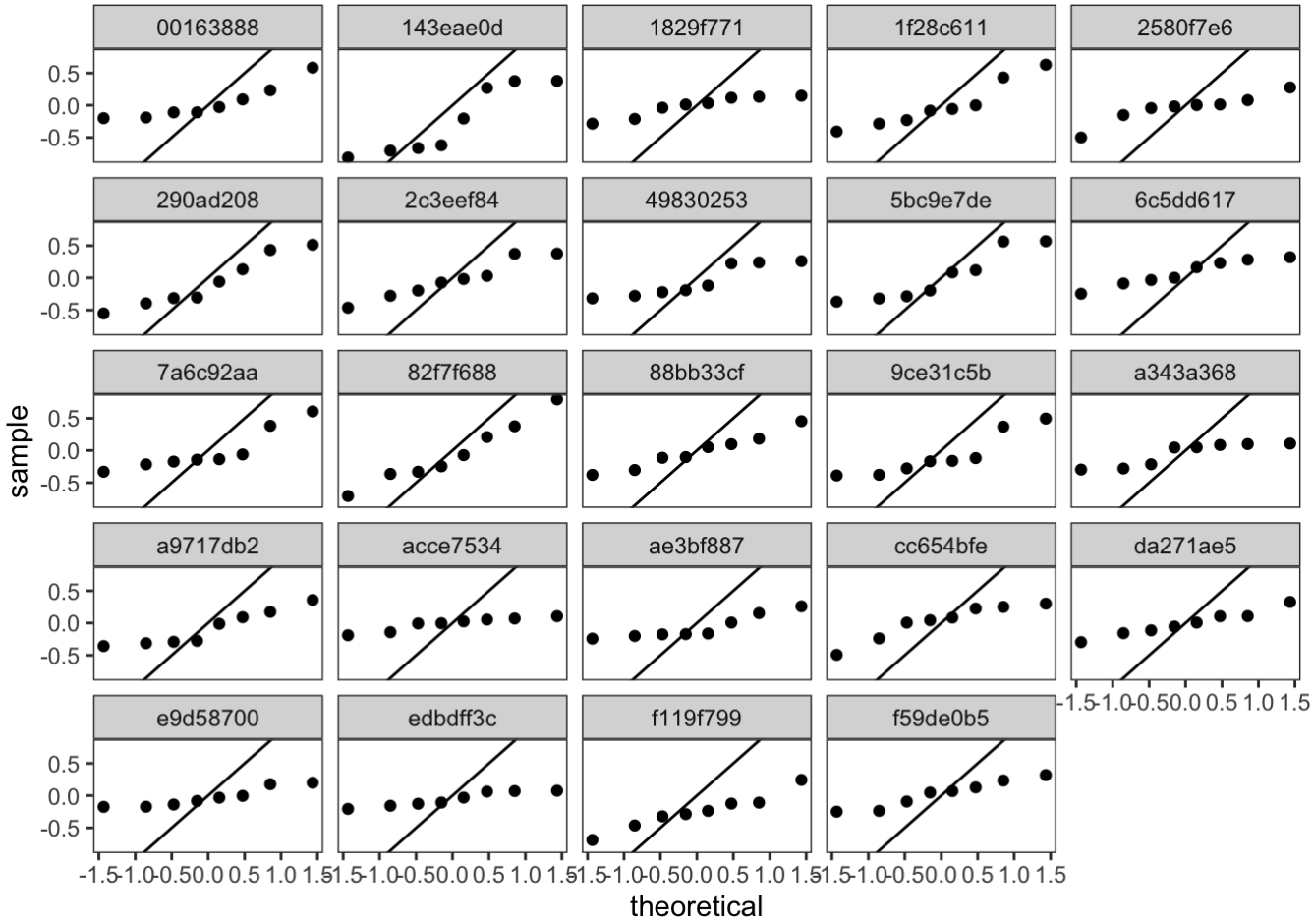
What does this look like in probability units?

```
model_df %>%
  group_by(lo_ground_truth, worker_id, means) %>%
  add_predicted_draws(m.wrkr.means.llo_p_sup) %>%
  ggplot(aes(x = plogis(lo_ground_truth), y = plogis(lo_p_sup), color = condition, fill
= condition)) +
  geom_abline(intercept = 0, slope = 1, size = 1, alpha = .3, color = "red", linetype =
"dashed") + # ground truth
  stat_lineribbon(aes(y = plogis(.prediction)), .width = c(.95, .80, .50), alpha = .25)
 +
  geom_point(data = model_df) +
  scale_fill_brewer(type = "qual", palette = 1) +
  scale_color_brewer(type = "qual", palette = 1) +
  coord_cartesian(xlim = quantile(plogis(model_df$lo_ground_truth), c(0, 1)),
                  ylim = quantile(plogis(model_df$lo_p_sup), c(0, 1))) +
  theme_bw() +
  theme(panel.grid = element_blank()) +
  facet_wrap(~ worker_id)
```



To examine more closely whether our model has predictive validity at the level of each worker, we'll look at QQ plots for residuals at the worker level.

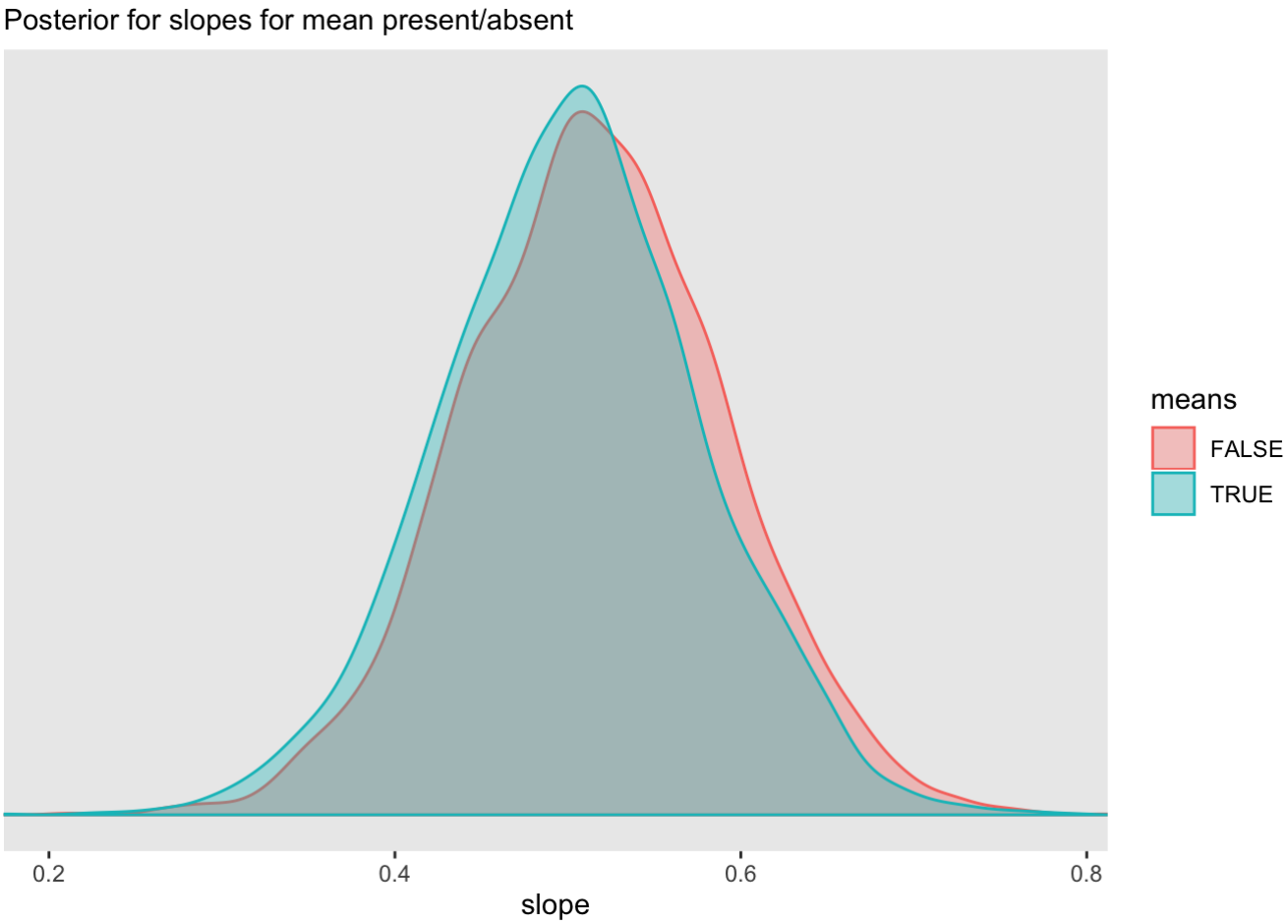
```
model_df %>%
  add_predicted_draws(m.wrkr.means.llo_p_sup) %>%
  group_by(lo_ground_truth, worker_id) %>%
  summarise(
    p_residual = mean(.prediction < lo_p_sup), # what proportion of predicted judgments
 are less than the observed response?
    z_residual = qnorm(p_residual)             # what are the z-scores of these cumulati
ve probabilities?
  ) %>%
  ggplot(aes(sample = z_residual)) +
  geom_qq() +
  geom_abline() +
  theme_bw() +
  theme(panel.grid = element_blank()) +
  facet_wrap(~ worker_id)
```



These still look pretty terrible.

With this model we can take a first stab at addressing our research question about the presence of extrinsic means. What does the posterior for the slope of the LLO model look like when means are present vs absent, ignoring other manipulations for now? Since we are building a complex model, we'll forego calculating maringal effects by manually combining parameters. Instead we'll use add_fitted_draws and compare_levels from tidybayes to get our slopes, and then we'll take their weighted average grouping by the parameters for which we want marginal effects.

```
model_df %>%
  group_by(means) %>%
  data_grid(lo_ground_truth = c(0, 1)) %>%          # get fitted draws (in log odds unit
s) only for ground truth of 0 and 1
  add_fitted_draws(m.wrkr.means.llo_p_sup, re_formula = NA) %>%
  compare_levels(.value, by = lo_ground_truth) %>%  # calculate the difference between f
its at 1 and 0 (i.e., slope)
  rename(slope = .value) %>%
  group_by(means, .draw) %>%                        # group by predictors to keep
  summarise(slope = weighted.mean(slope)) %>%       # marginalize out visualization cond
ition by taking a weighted average
  ggplot(aes(x = slope, group = means, color = means, fill = means)) +
  geom_density(alpha = 0.35) +
  scale_x_continuous(expression(slope), expand = c(0, 0)) +
  scale_y_continuous(NULL, breaks = NULL) +
  labs(subtitle = "Posterior for slopes for mean present/absent") +
  theme(panel.grid = element_blank())
```



Recall that a slope of 1 represents no bias. This chart suggests that people are biased with or without adding means. We should not be surprised to see little to no effect in this model. The mean difference is a good heuristic for probability of superiority when variance of visualized estimates is high, but it is not a good heuristic when variance is low. Thus, we should expect to see the effect we are looking for as an interaction between the presence of the mean and the level of uncertainty.

Level of Uncertainty Shown

Another factor that we manipulate is the level of uncertainty presented to chart users. We expect level of uncertainty (sd_diff) to determine the impact of extrinsic means on performance. To test this, we'll add an interaction between sd_diff, means, and the ground truth.

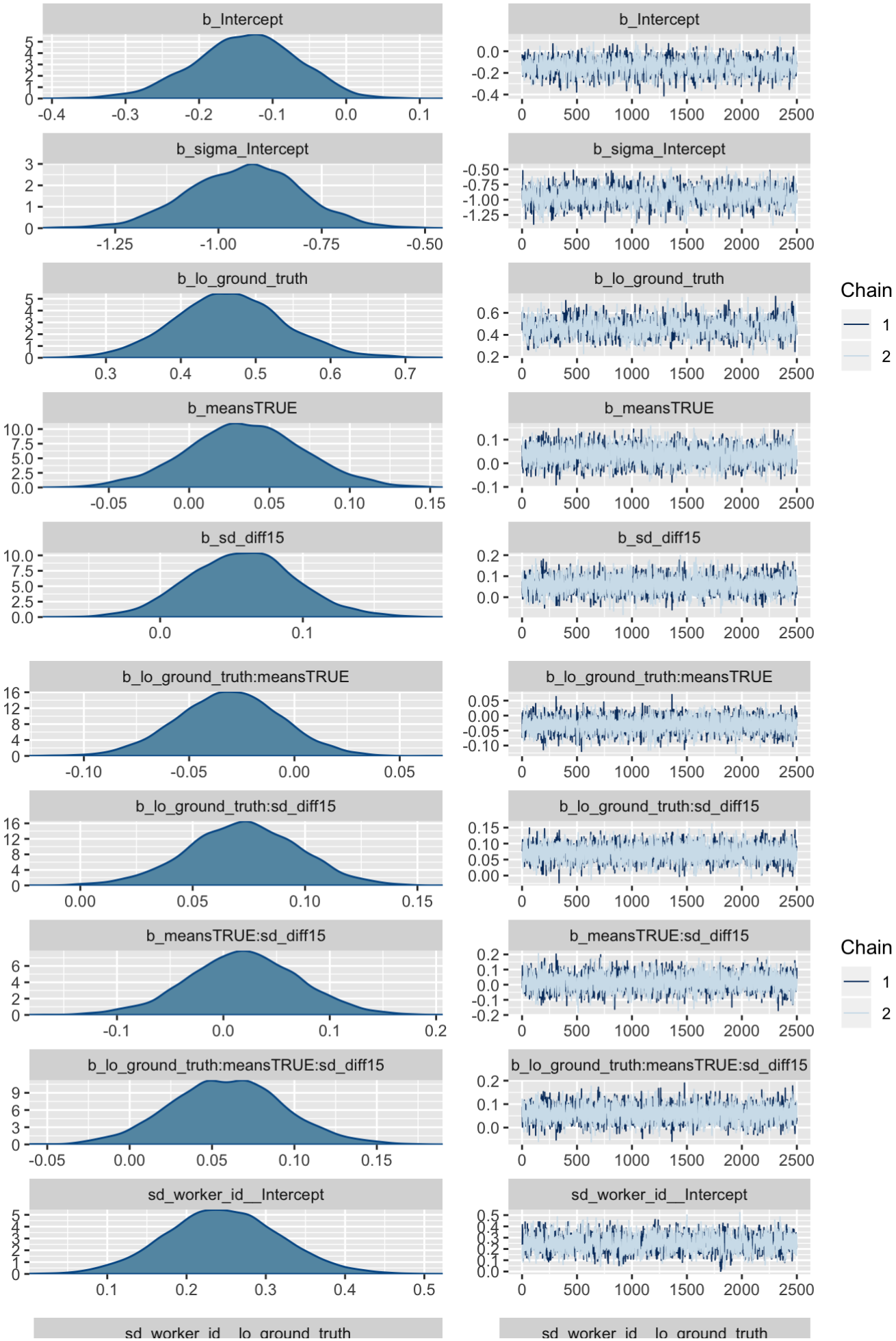
We use the same priors as we did for the previous model. Now, let's fit the model to our data.

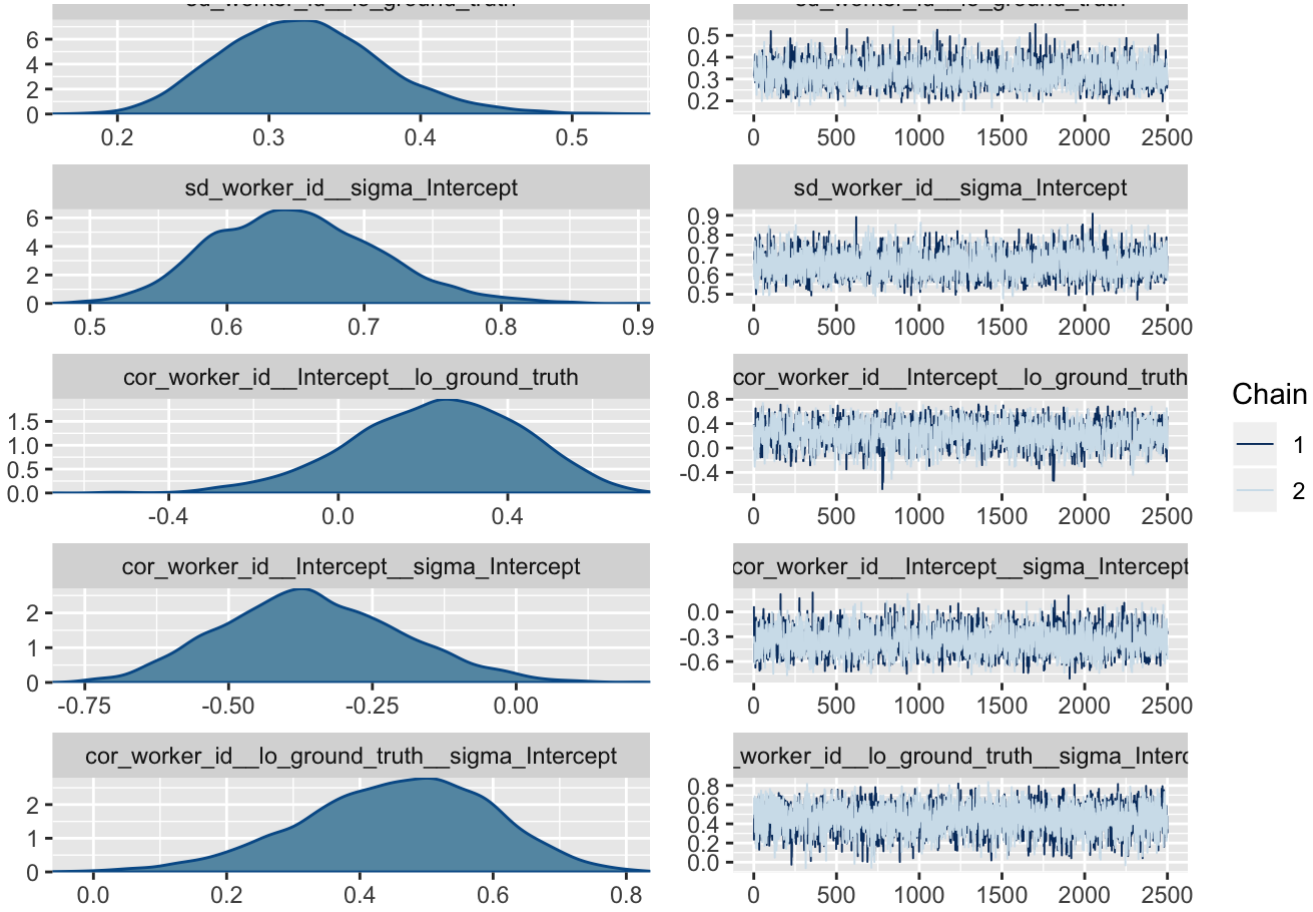
```
# hierarchical LLO model
m.wrkr.means.sd.llo_p_sup <- brm(data = model_df, family = "gaussian",
                            formula = bf(lo_p_sup ~  (1 + lo_ground_truth|sharecor|worke
r_id) + lo_ground_truth*means*sd_diff,
                                         sigma ~ (1|sharecor|worker_id)),
                            prior = c(prior(normal(1, 0.5), class = b),
                                      prior(normal(1.3, 1), class = Intercept),
                                      prior(normal(0, 0.15), class = sd, group = worker_
id),
                                      # prior(normal(0, 0.3), class = b, dpar = sigma),
                                      prior(normal(0, 0.15), class = sd, dpar = sigma),
                                      prior(lkj(4), class = cor)),
                            iter = 3000, warmup = 500, chains = 2, cores = 2,
                            control = list(adapt_delta = 0.99, max_treedepth = 12),
                            file = "model-fits/llo_mdl-wrkr_means_sd")
```

Check diagnostics:

- Trace plots

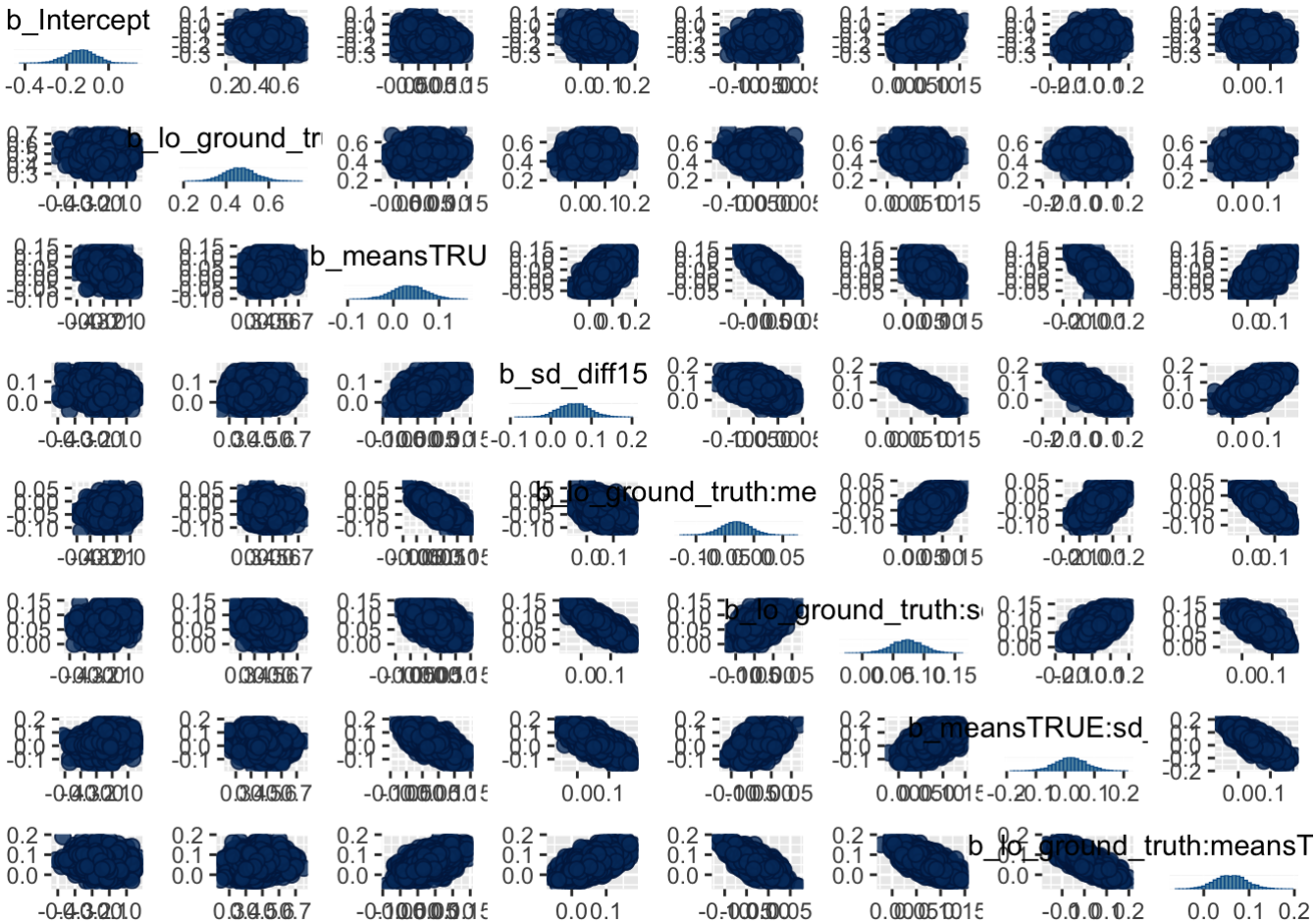
```
# trace plots
plot(m.wrkr.means.sd.llo_p_sup)
```


b_Intercept
b_sigma_Intercept
b_lo_ground_truth
b_meansTRUE
b_sd_diff15
b_lo_ground_truth:meansTRUE
b_lo_ground_truth:sd_diff15
b_meansTRUE:sd_diff15
b_lo_ground_truth:meansTRUE:sd_diff15
sd_worker_id__Intercept
Chain
1
2

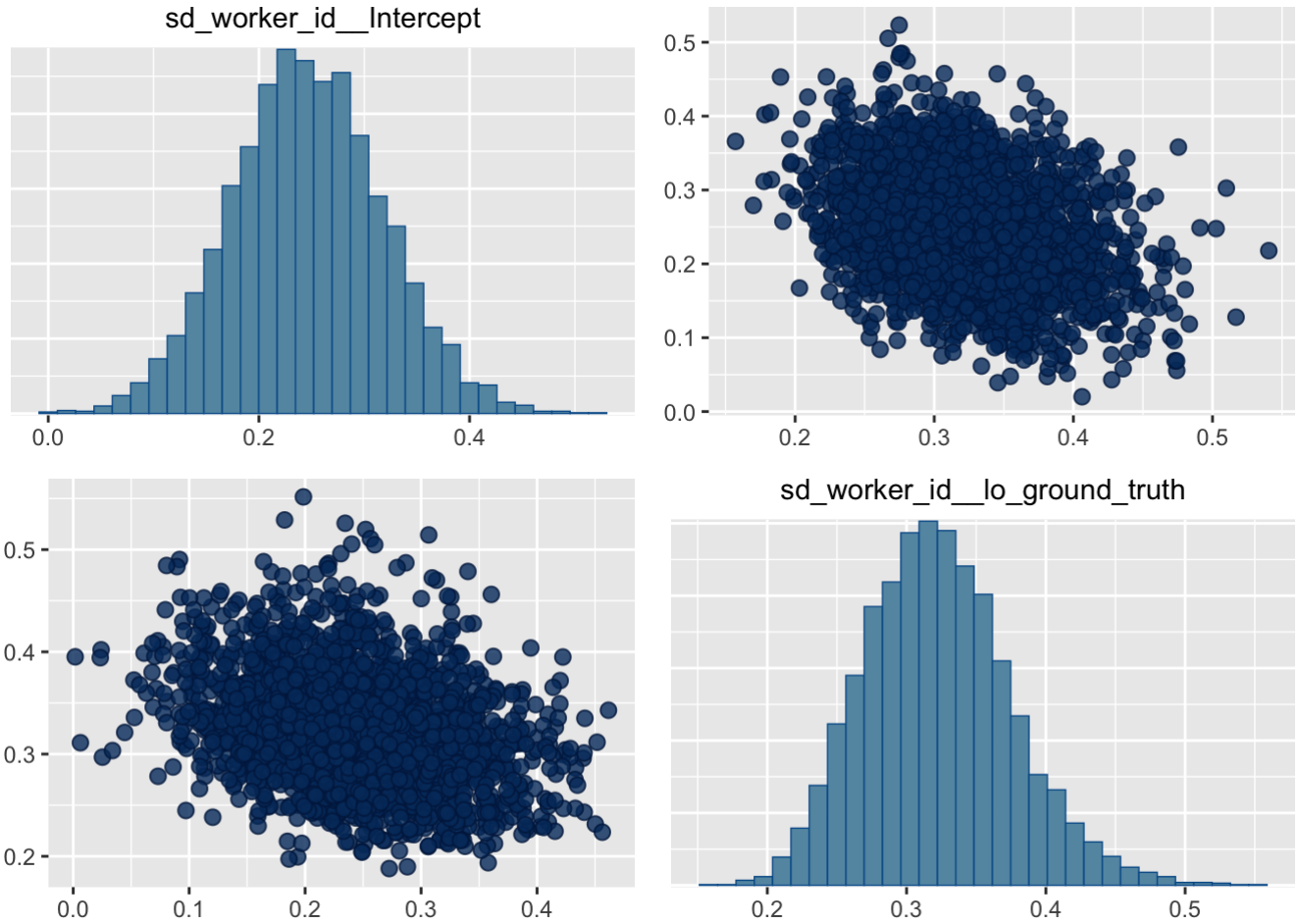


- Pairs plot

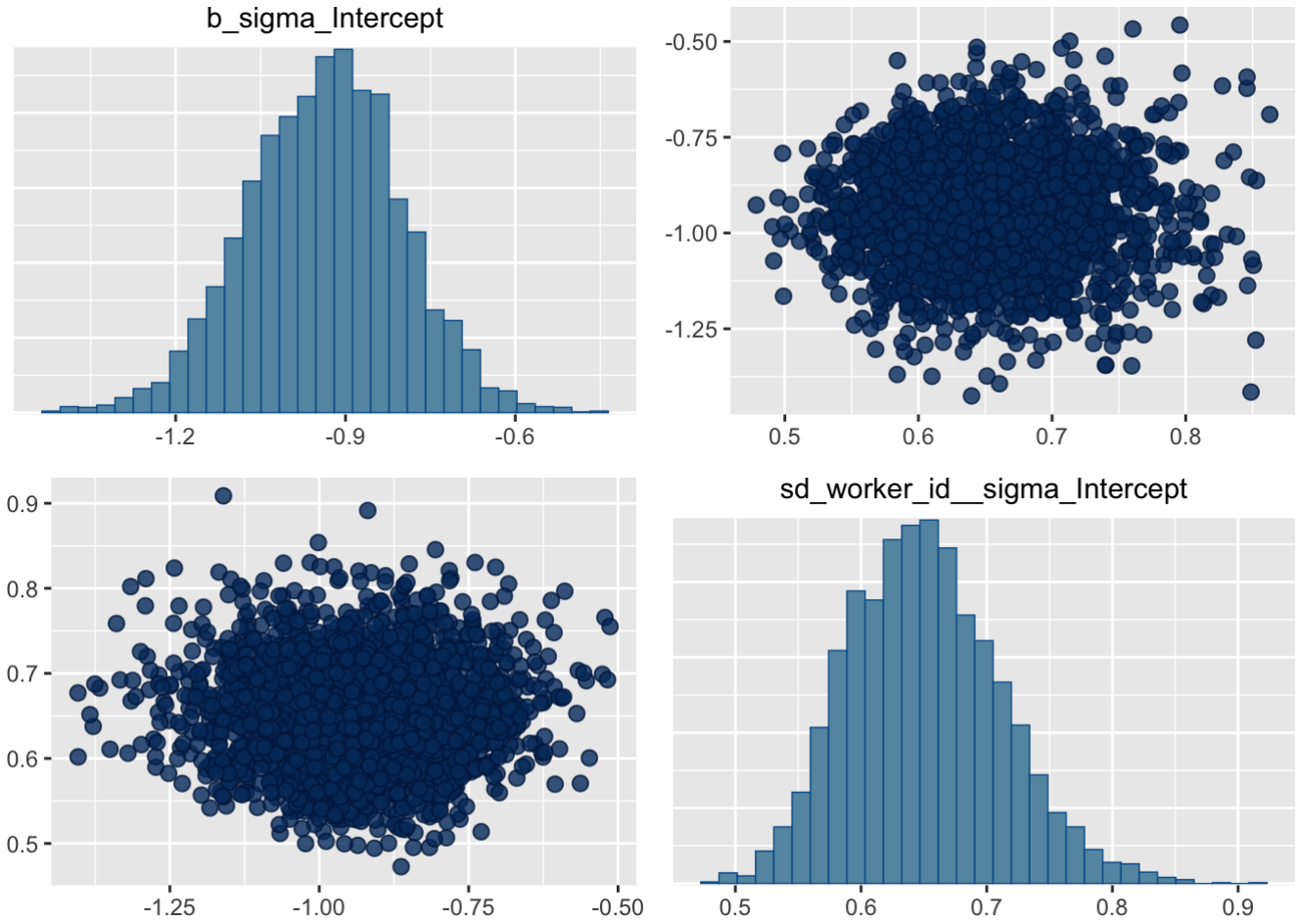
```
# pairs plot (LLO params)
pairs(m.wrkr.means.sd.llo_p_sup, exact_match = TRUE, pars = c("b_Intercept",
                                                              "b_lo_ground_truth",
                                                              "b_meansTRUE",
                                                              "b_sd_diff15",
                                                              "b_lo_ground_truth:meansTR
UE",
                                                              "b_lo_ground_truth:sd_diff
15",
                                                              "b_meansTRUE:sd_diff15",
                                                              "b_lo_ground_truth:meansTR
UE:sd_diff15"))
```



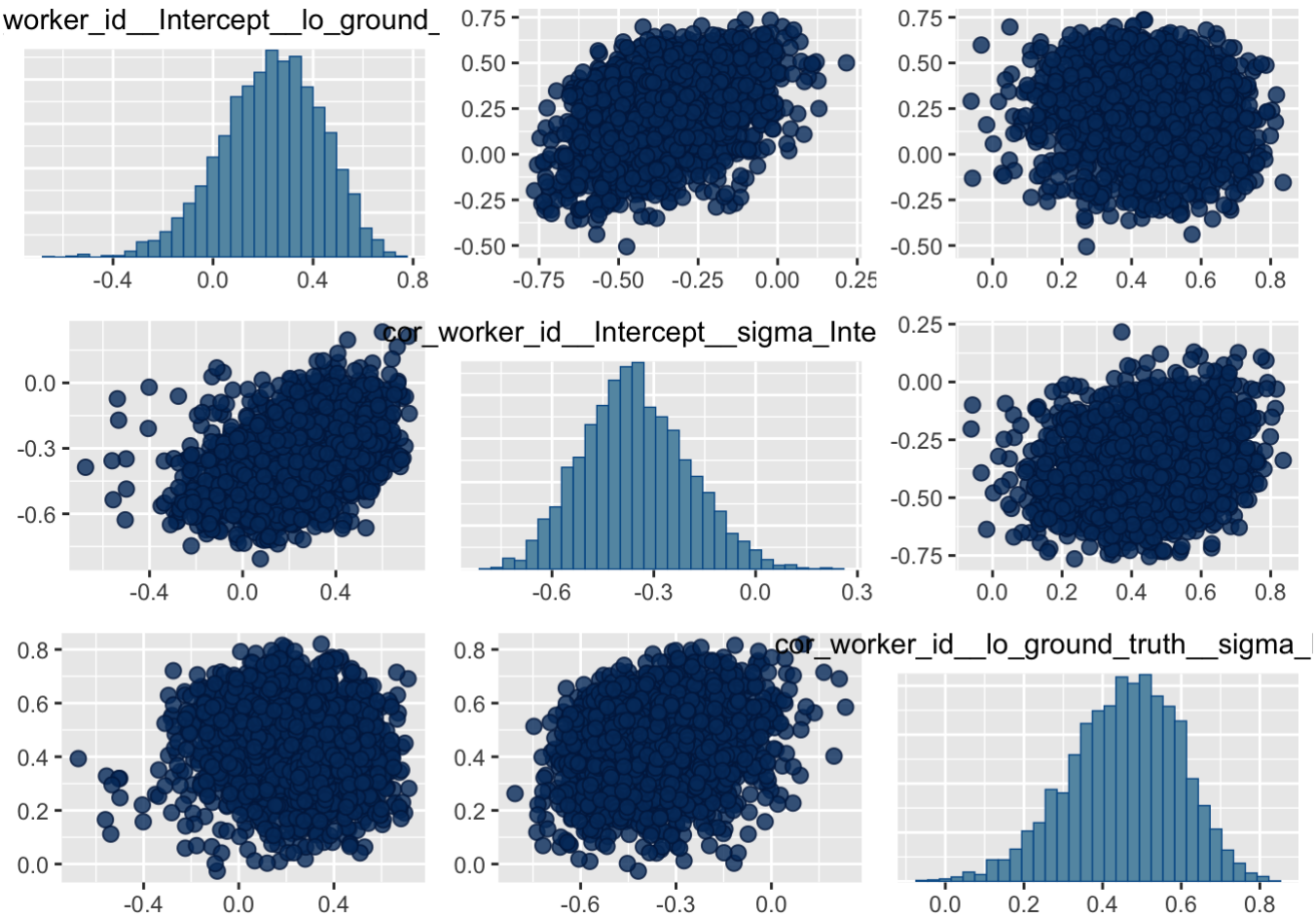

```
# pairs plot (random effects on lo_p_sup)
pairs(m.wrkr.means.sd.llo_p_sup, exact_match = TRUE, pars = c("sd_worker_id__Intercept",
                                                              "sd_worker_id__lo_ground_t
ruth"))
```




```
# pairs plot (sigma params)
pairs(m.wrkr.means.sd.llo_p_sup, exact_match = TRUE, pars = c("b_sigma_Intercept",
                                                              "sd_worker_id__sigma_Inter
cept"))
```




```
pairs(m.wrkr.means.sd.llo_p_sup, pars = c("cor_worker_id__"))
```



- Summary

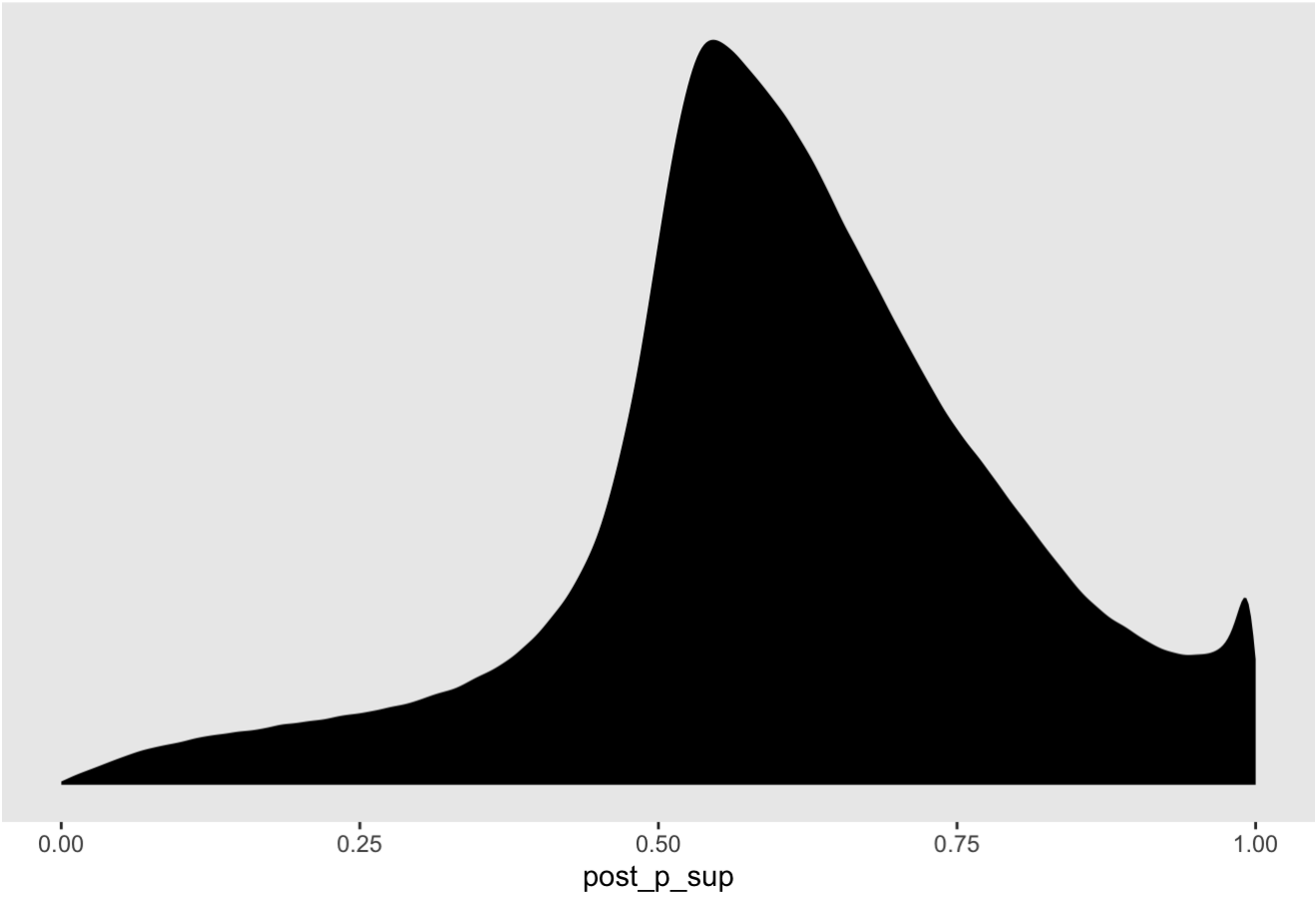
```
# model summary
print(m.wrkr.means.sd.llo_p_sup)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = log
## Formula: lo_p_sup ~ (1 + lo_ground_truth | sharecor | worker_id) + lo_ground_truth *
means * sd_diff
##          sigma ~ (1 | sharecor | worker_id)
##    Data: model_df (Number of observations: 768)
## Samples: 2 chains, each with iter = 3000; warmup = 500; thin = 1;
##          total post-warmup samples = 5000
##
## Group-Level Effects:
## ~worker_id (Number of levels: 24)
##                                    Estimate Est.Error l-95% CI u-95% CI
## sd(Intercept)                          0.25      0.07     0.11     0.39
## sd(lo_ground_truth)                    0.32      0.05     0.23     0.44
## sd(sigma_Intercept)                    0.65      0.06     0.54     0.78
## cor(Intercept,lo_ground_truth)         0.23      0.20    -0.19     0.59
## cor(Intercept,sigma_Intercept)        -0.35      0.15    -0.64    -0.03
## cor(lo_ground_truth,sigma_Intercept)   0.45      0.14     0.16     0.70
##                                    Eff.Sample Rhat
## sd(Intercept)                            1428 1.00
## sd(lo_ground_truth)                      1975 1.00
## sd(sigma_Intercept)                      2711 1.00
## cor(Intercept,lo_ground_truth)           1378 1.00
## cor(Intercept,sigma_Intercept)           2249 1.00
## cor(lo_ground_truth,sigma_Intercept)      2542 1.00
##
## Population-Level Effects:
##                                  Estimate Est.Error l-95% CI u-95% CI
## Intercept                           -0.14      0.07    -0.28    -0.01
## sigma_Intercept                     -0.94      0.14    -1.20    -0.68
## lo_ground_truth                      0.47      0.07     0.32     0.62
## meansTRUE                            0.04      0.04    -0.04     0.11
## sd_diff15                            0.06      0.04    -0.01     0.13
## lo_ground_truth:meansTRUE           -0.03      0.02    -0.08     0.02
## lo_ground_truth:sd_diff15            0.07      0.02     0.02     0.12
## meansTRUE:sd_diff15                  0.02      0.05    -0.09     0.12
## lo_ground_truth:meansTRUE:sd_diff15  0.06      0.03    -0.01     0.13
##                                  Eff.Sample Rhat
## Intercept                              1822 1.00
## sigma_Intercept                        1644 1.00
## lo_ground_truth                        1345 1.00
## meansTRUE                              3569 1.00
## sd_diff15                              3595 1.00
## lo_ground_truth:meansTRUE              3385 1.00
## lo_ground_truth:sd_diff15              3463 1.00
## meansTRUE:sd_diff15                    3355 1.00
## lo_ground_truth:meansTRUE:sd_diff15    3009 1.00
##
## Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
## is a crude measure of effective sample size, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

Let's check our posterior predictive distribution.

```
# posterior predictive check
model_df %>%
  select(lo_ground_truth, worker_id, means) %>%
  add_predicted_draws(m.wrkr.means.llo_p_sup, prediction = "lo_p_sup", seed = 1234) %>%
  mutate(
    # transform to probability units
    post_p_sup = plogis(lo_p_sup)
  ) %>%
  ggplot(aes(x = post_p_sup)) +
  geom_density(fill = "black", size = 0) +
  scale_y_continuous(NULL, breaks = NULL) +
  labs(subtitle = "Posterior predictive distribution for probability of superiority") +
  theme(panel.grid = element_blank())
```

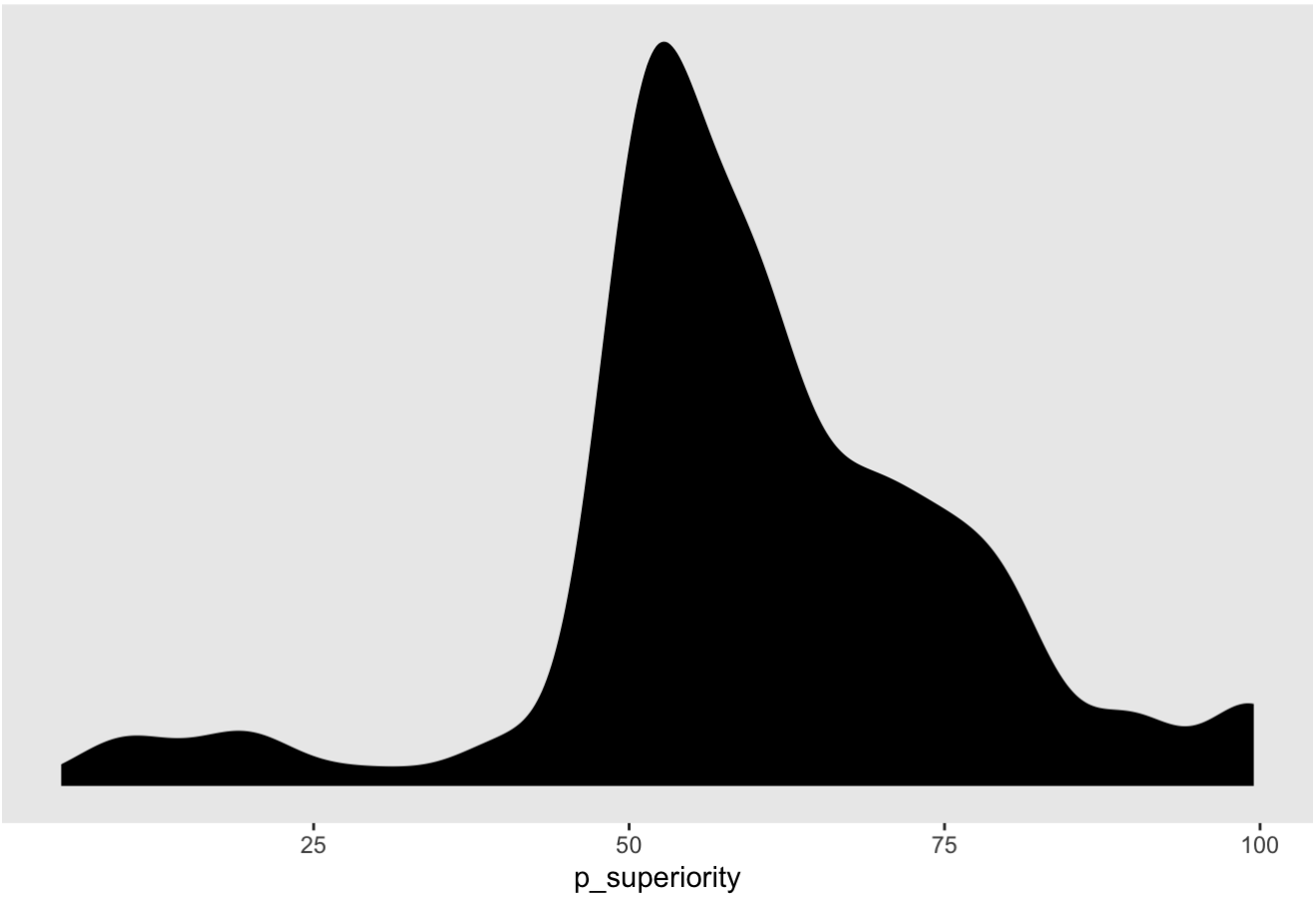
Posterior predictive distribution for probability of superiority



How do these predictions compare to the observed data?

```
# data density
model_df %>%
  ggplot(aes(x = p_superiority)) +
  geom_density(fill = "black", size = 0) +
  scale_y_continuous(NULL, breaks = NULL) +
  labs(subtitle = "Data distribution for probability of superiority") +
  theme(panel.grid = element_blank())
```

Data distribution for probability of superiority



Running a leave one out posterior predictive check, we can see that overall this model has decent predictive validity.

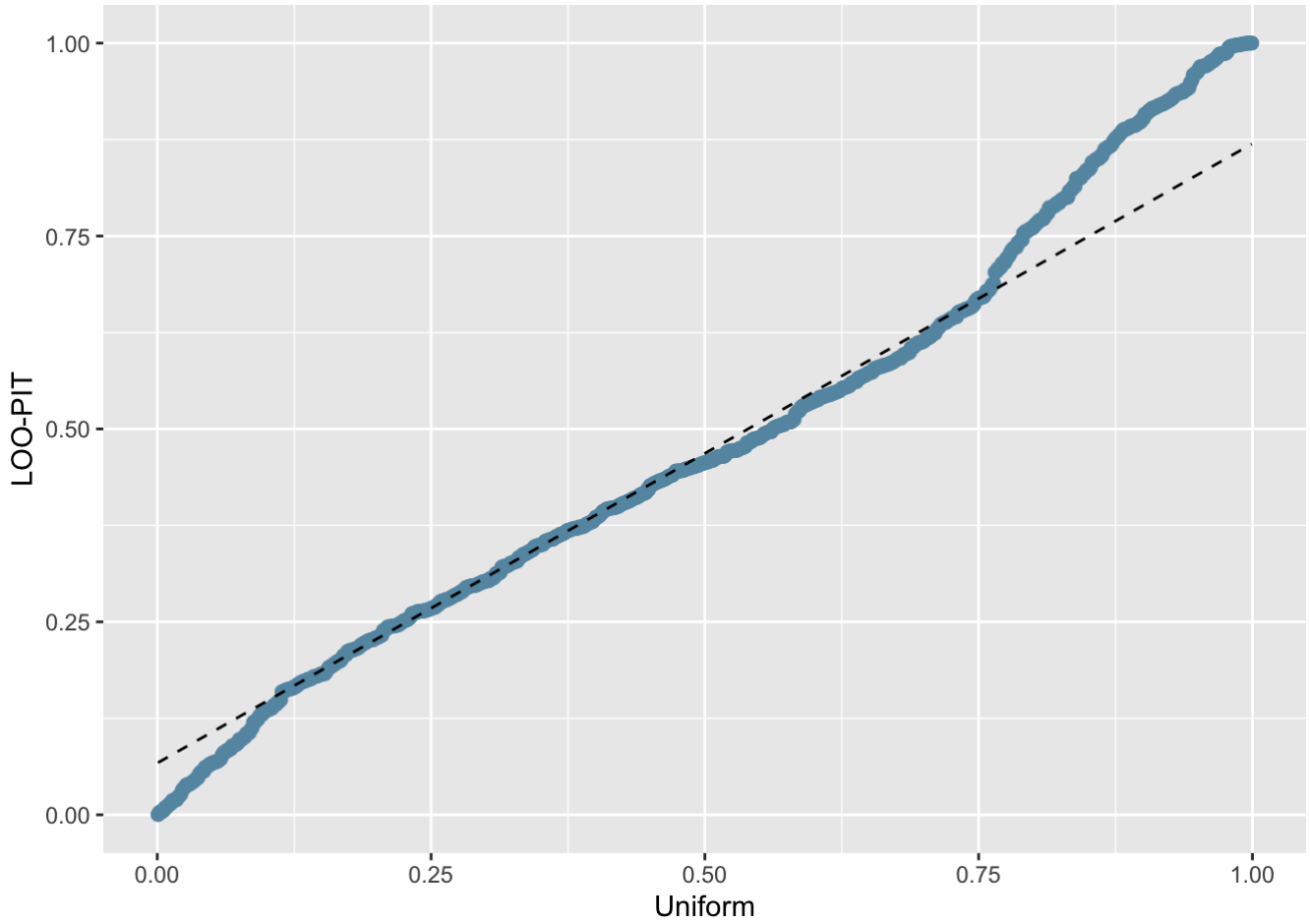
```
# set up data for LOO posterior predictive check
y <- model_df$lo_p_sup
yrep <- posterior_predict(m.wrkr.means.sd.llo_p_sup)

# run LOO to get weights
loo <- loo(m.wrkr.means.sd.llo_p_sup, save_psis = TRUE, cores = 2)
```

```
## Warning: Found 5 observations with a pareto_k > 0.7 in model
## 'm.wrkr.means.sd.llo_p_sup'. It is recommended to set 'reloo = TRUE' in
## order to calculate the ELPD without the assumption that these observations
## are negligible. This will refit the model 5 times to compute the ELPDs for
## the problematic observations directly.
```

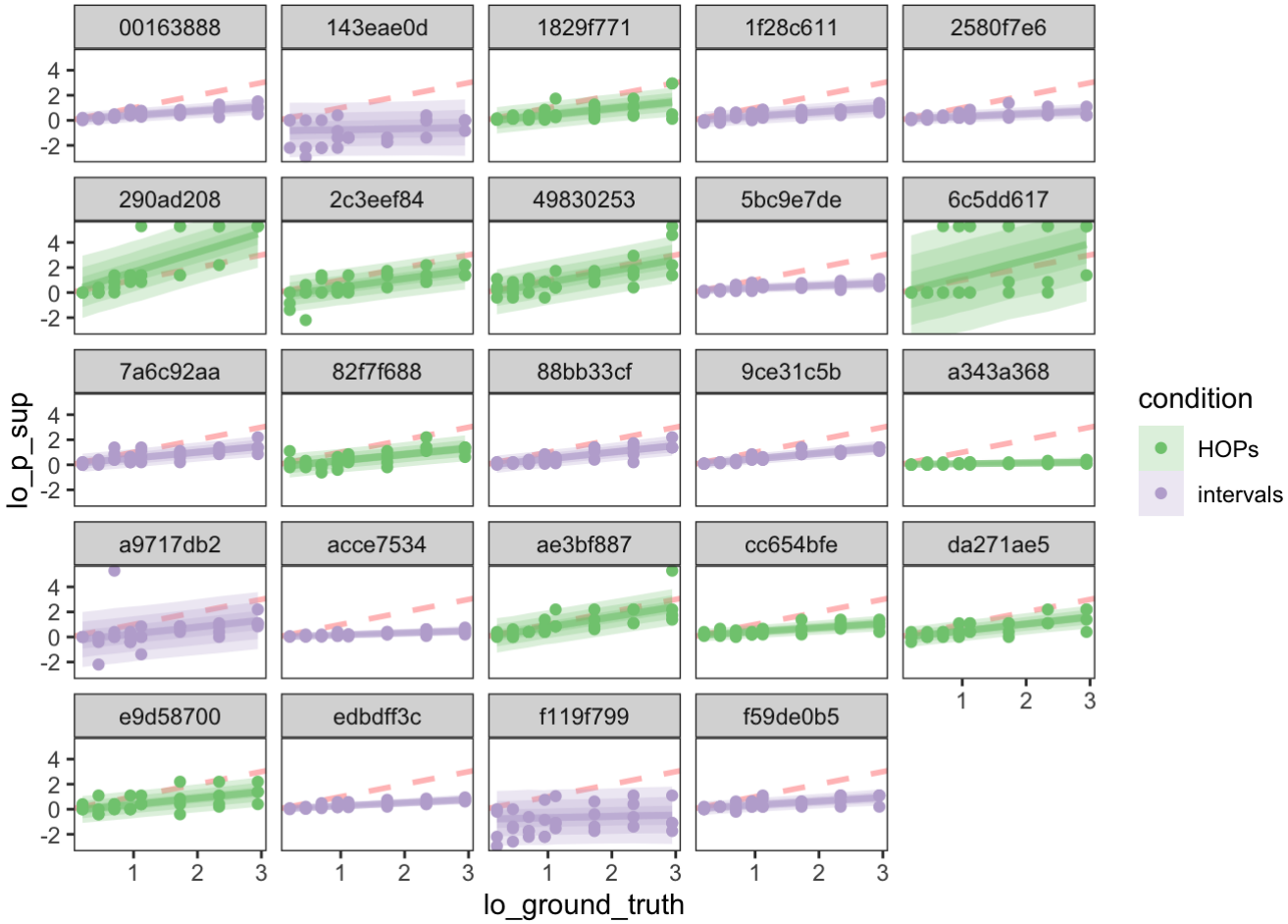
```
psis <- loo$psis_object
lw <- weights(psis)
```

```
ppc_loo_pit_qq(y, yrep, lw = lw)
```



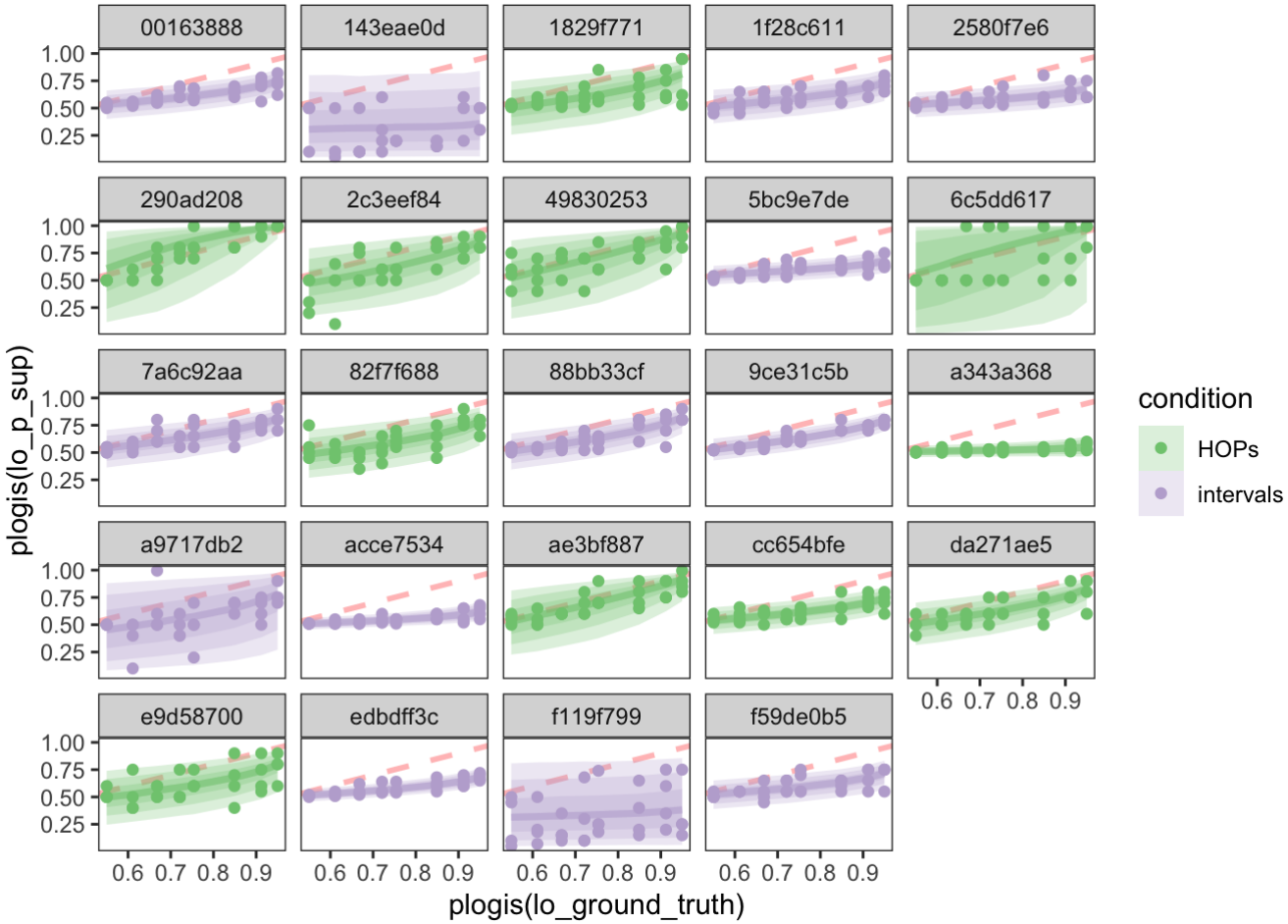
Let's take a look at predictions per worker and visualization condition to get a more granular sense of our model fit.

```
model_df %>%
  group_by(lo_ground_truth, worker_id, means, sd_diff) %>%
  add_predicted_draws(m.wrkr.means.sd.llo_p_sup) %>%
  ggplot(aes(x = lo_ground_truth, y = lo_p_sup, color = condition, fill = condition)) +
  geom_abline(intercept = 0, slope = 1, size = 1, alpha = .3, color = "red", linetype =
"dashed") + # ground truth
  stat_lineribbon(aes(y = .prediction), .width = c(.95, .80, .50), alpha = .25) +
  geom_point(data = model_df) +
  scale_fill_brewer(type = "qual", palette = 1) +
  scale_color_brewer(type = "qual", palette = 1) +
  coord_cartesian(xlim = quantile(model_df$lo_ground_truth, c(0, 1)),
                  ylim = quantile(model_df$lo_p_sup, c(0, 1))) +
  theme_bw() +
  theme(panel.grid = element_blank()) +
  facet_wrap(~ worker_id)
```



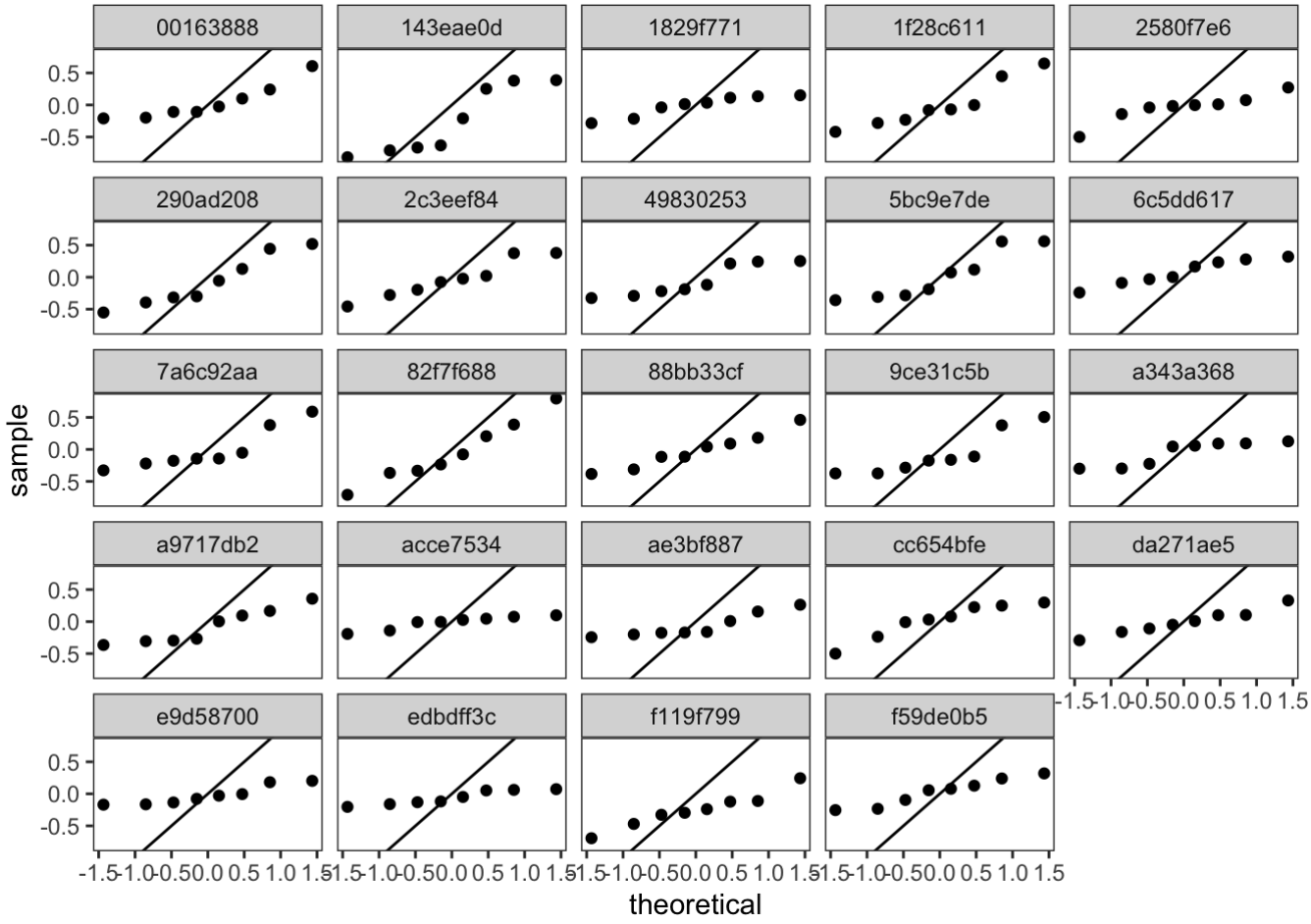
What does this look like in probability units?

```
model_df %>%
  group_by(lo_ground_truth, worker_id, means, sd_diff) %>%
  add_predicted_draws(m.wrkr.means.sd.llo_p_sup) %>%
  ggplot(aes(x = plogis(lo_ground_truth), y = plogis(lo_p_sup), color = condition, fill
= condition)) +
  geom_abline(intercept = 0, slope = 1, size = 1, alpha = .3, color = "red", linetype =
"dashed") + # ground truth
  stat_lineribbon(aes(y = plogis(.prediction)), .width = c(.95, .80, .50), alpha = .25)
 +
  geom_point(data = model_df) +
  scale_fill_brewer(type = "qual", palette = 1) +
  scale_color_brewer(type = "qual", palette = 1) +
  coord_cartesian(xlim = quantile(plogis(model_df$lo_ground_truth), c(0, 1)),
                  ylim = quantile(plogis(model_df$lo_p_sup), c(0, 1))) +
  theme_bw() +
  theme(panel.grid = element_blank()) +
  facet_wrap(~ worker_id)
```



To examine more closely whether our model has predictive validity at the level of each worker, we'll look at QQ plots for residuals at the worker level.

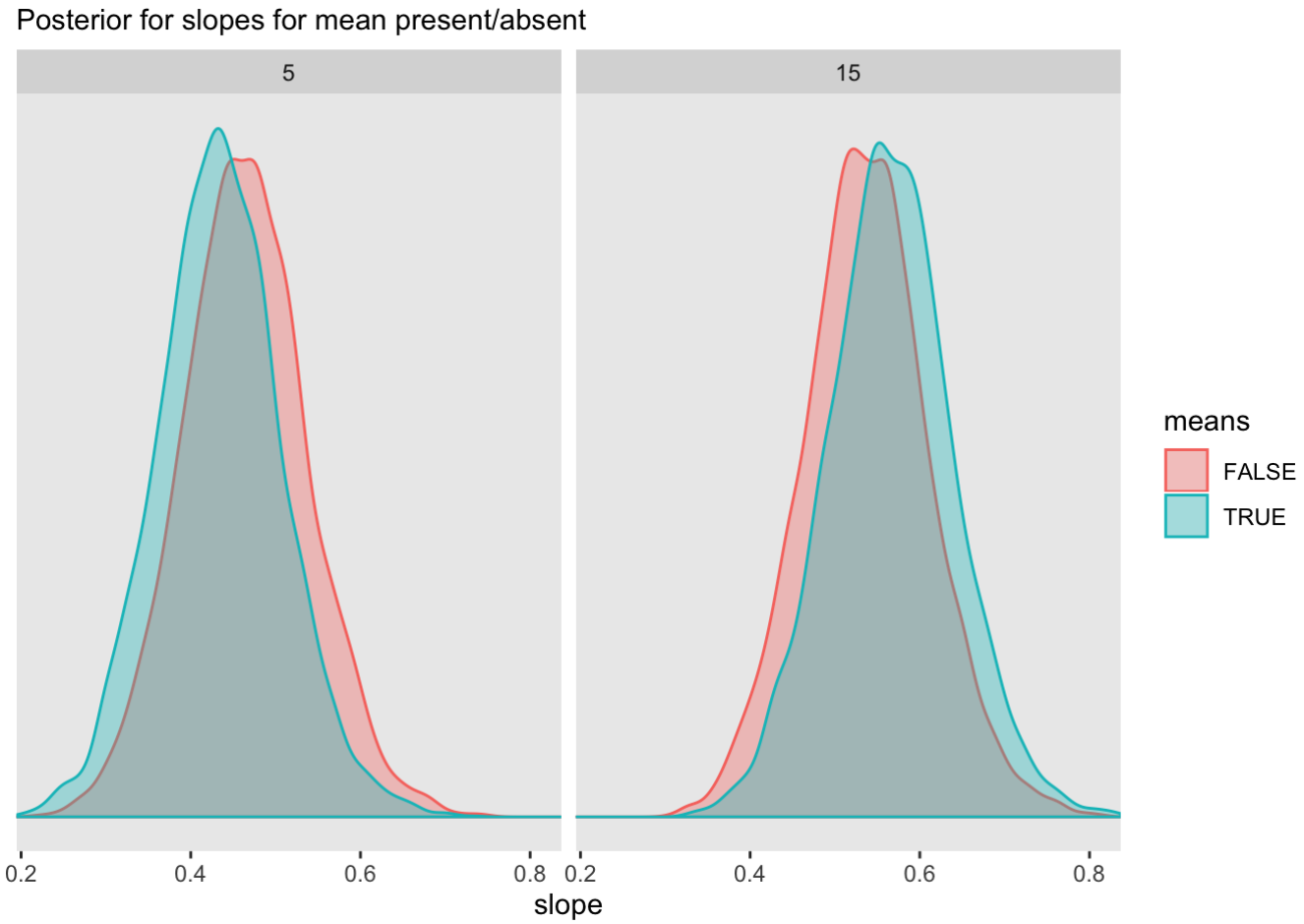
```
model_df %>%
  add_predicted_draws(m.wrkr.llo_p_sup) %>%
  group_by(lo_ground_truth, worker_id) %>%
  summarise(
    p_residual = mean(.prediction < lo_p_sup), # what proportion of predicted judgments
 are less than the observed response?
    z_residual = qnorm(p_residual)             # what are the z-scores of these cumulati
ve probabilities?
  ) %>%
  ggplot(aes(sample = z_residual)) +
  geom_qq() +
  geom_abline() +
  theme_bw() +
  theme(panel.grid = element_blank()) +
  facet_wrap(~ worker_id)
```



These still look pretty terrible.

What does the posterior for the slope of the LLO model look like when means are present vs absent at different levels of uncertainty, ignoring other manipulations?

```
model_df %>%
  group_by(means, sd_diff) %>%
  data_grid(lo_ground_truth = c(0, 1)) %>%          # get fitted draws (in log odds unit
s) only for ground truth of 0 and 1
  add_fitted_draws(m.wrkr.means.sd.llo_p_sup, re_formula = NA) %>%
  compare_levels(.value, by = lo_ground_truth) %>%  # calculate the difference between f
its at 1 and 0 (i.e., slope)
  rename(slope = .value) %>%
  group_by(means, sd_diff, .draw) %>%               # group by predictors to keep
  summarise(slope = weighted.mean(slope)) %>%       # marginalize out visualization cond
ition by taking a weighted average
  ggplot(aes(x = slope, group = means, color = means, fill = means)) +
  geom_density(alpha = 0.35) +
  scale_x_continuous(expression(slope), expand = c(0, 0)) +
  scale_y_continuous(NULL, breaks = NULL) +
  labs(subtitle = "Posterior for slopes for mean present/absent") +
  theme(panel.grid = element_blank()) +
  facet_grid(. ~ sd_diff)
```



Recall that a slope of 1 represents no bias. Here we see exactly what we expect. Adding means makes responses more biased when uncertainty is low and less biased when uncertainty is high. Interestingly, people seem less biased at baseline when uncertainty is higher.

Visualization Condition

The other thing we really want to know about is the impact of visualization condition on the slopes of linear models in log odds space. Do some visualizations lead to more extreme patterns of bias than others? To test this, we'll add an interaction between visualization condition and the ground truth. Now we have all our predictors of interest in one model.

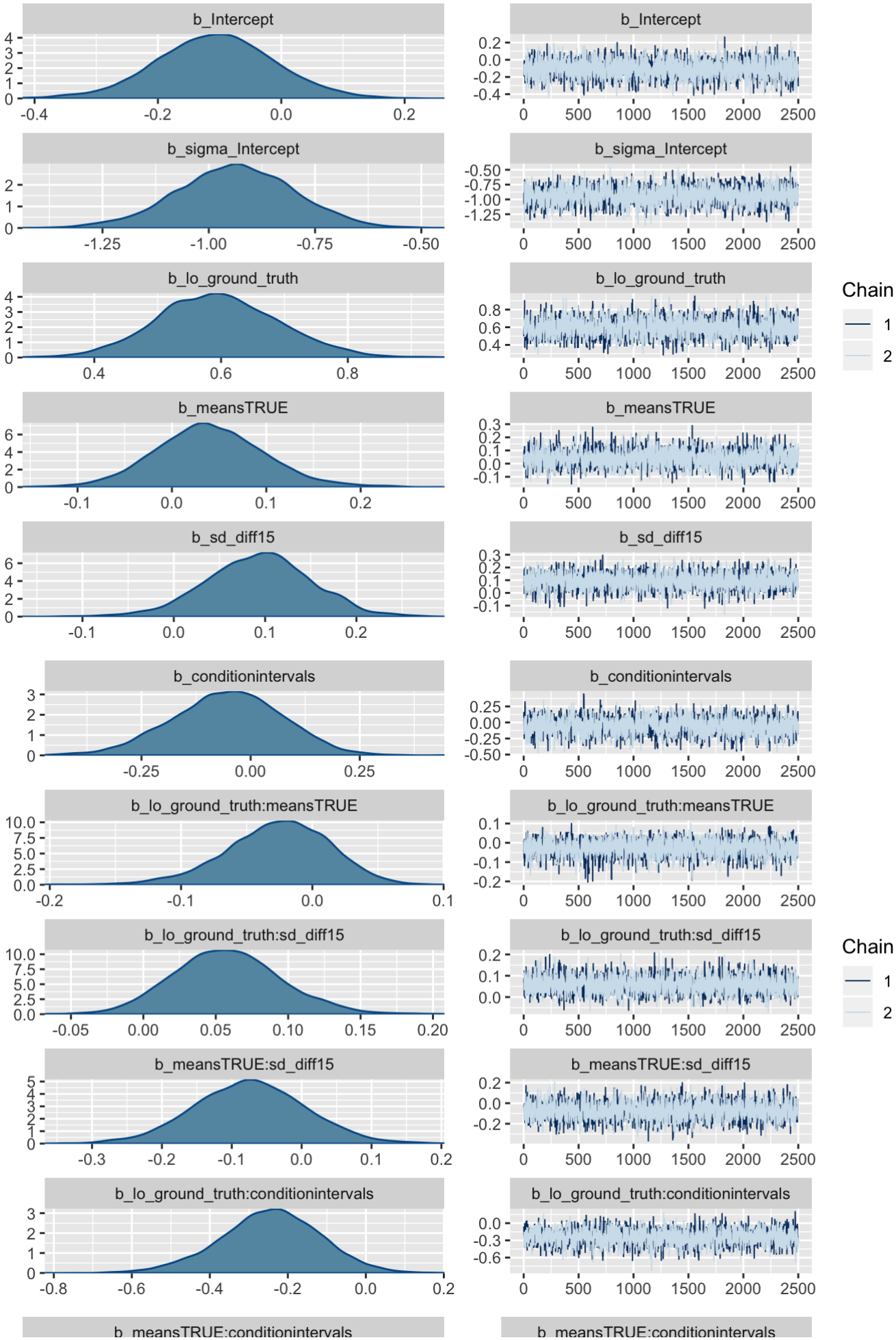
We use the same priors as we did for the previous model. Now, let's fit the model to our data.

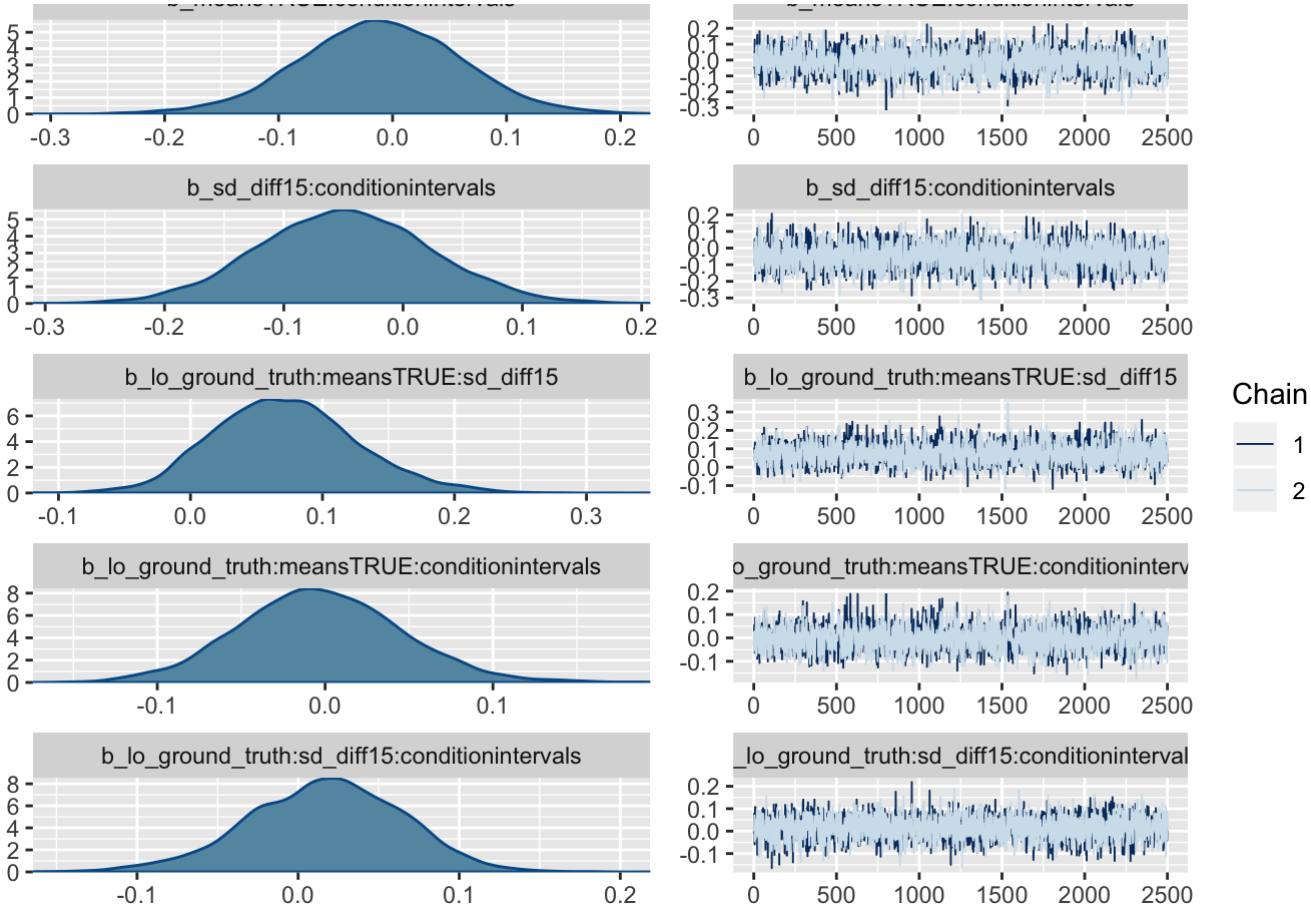
```
# hierarchical LLO model
m.wrkr.means.sd.vis.llo_p_sup <- brm(data = model_df, family = "gaussian",
                                     formula = bf(lo_p_sup ~  (1 + lo_ground_truth|share
cor|worker_id) + lo_ground_truth*means*sd_diff*condition,
                                                  sigma ~ (1|sharecor|worker_id)),
                                     prior = c(prior(normal(1, 0.5), class = b),
                                               prior(normal(1.3, 1), class = Intercept),
                                               prior(normal(0, 0.15), class = sd, group
 = worker_id),
                                               # prior(normal(0, 0.3), class = b, dpar =
sigma),
                                               prior(normal(0, 0.15), class = sd, dpar =
sigma),
                                               prior(lkj(4), class = cor)),
                                     iter = 3000, warmup = 500, chains = 2, cores = 2,
                                     control = list(adapt_delta = 0.99, max_treedepth =
12),
                                     file = "model-fits/llo_mdl-wrkr_means_sd_vis")
```

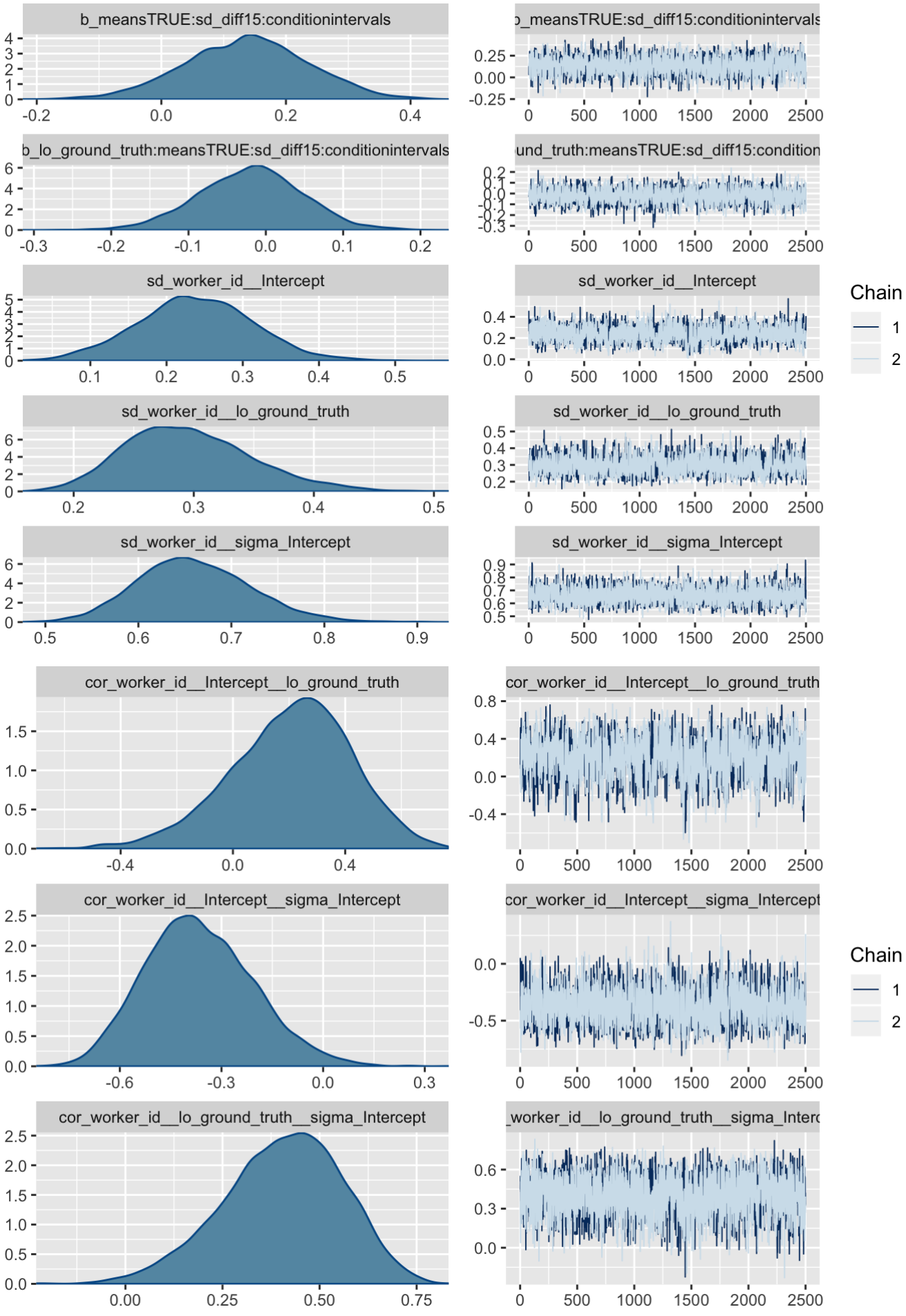
Check diagnostics:

- Trace plots

```
# trace plots
plot(m.wrkr.means.sd.vis.llo_p_sup)
```

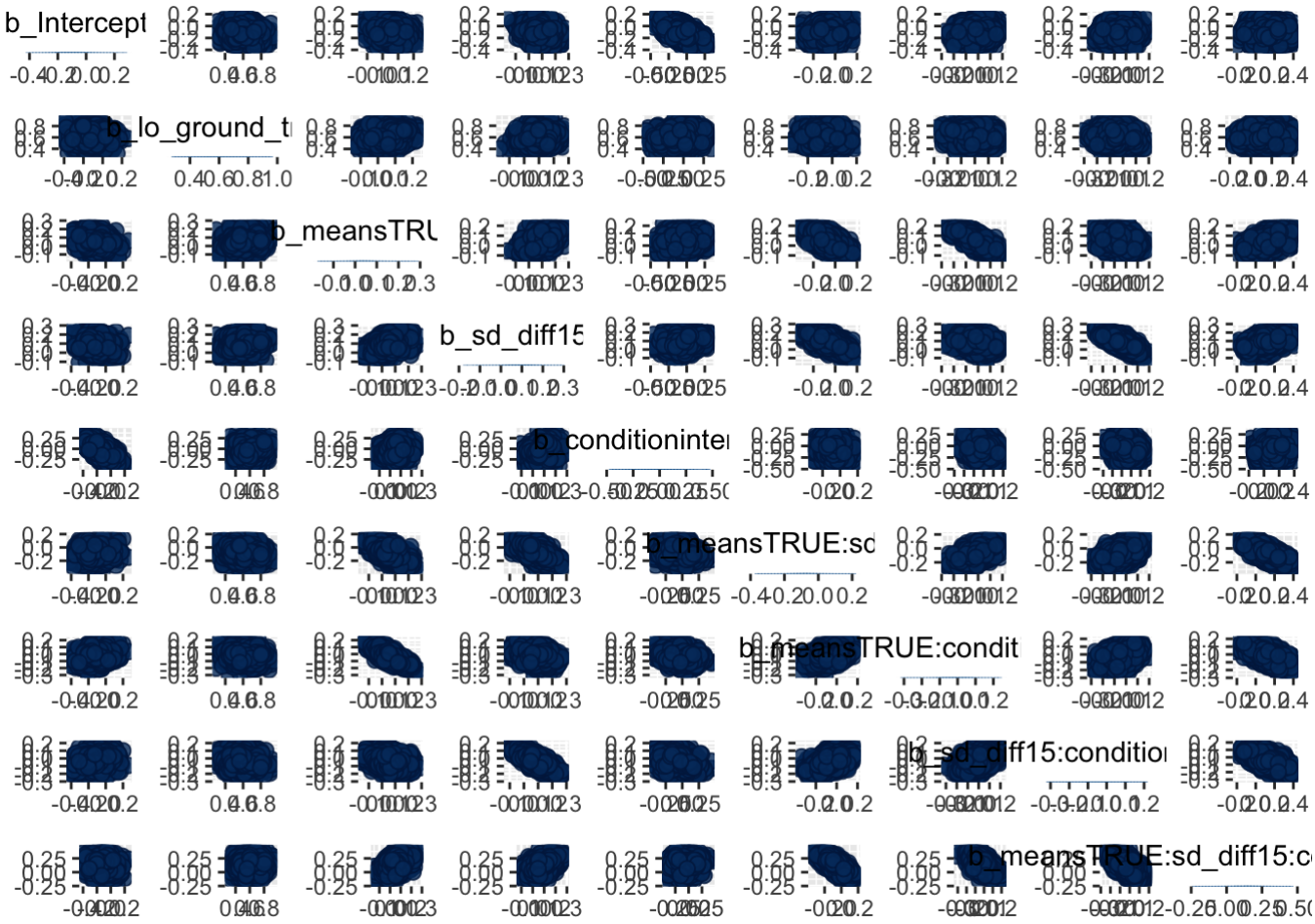

b_Intercept
b_sigma_Intercept
b_lo_ground_truth
b_meansTRUE
b_sd_diff15
b_conditionintervals
b_lo_ground_truth:meansTRUE
b_lo_ground_truth:sd_diff15
b_meansTRUE:sd_diff15
b_lo_ground_truth:conditionintervals
b_meansTRUE:conditionintervals
Chain
1
2


b_sd_diff15:conditionintervals
b_lo_ground_truth:meansTRUE:sd_diff15
b_lo_ground_truth:meansTRUE:conditionintervals
b_lo_ground_truth:sd_diff15:conditionintervals
Chain
1
2

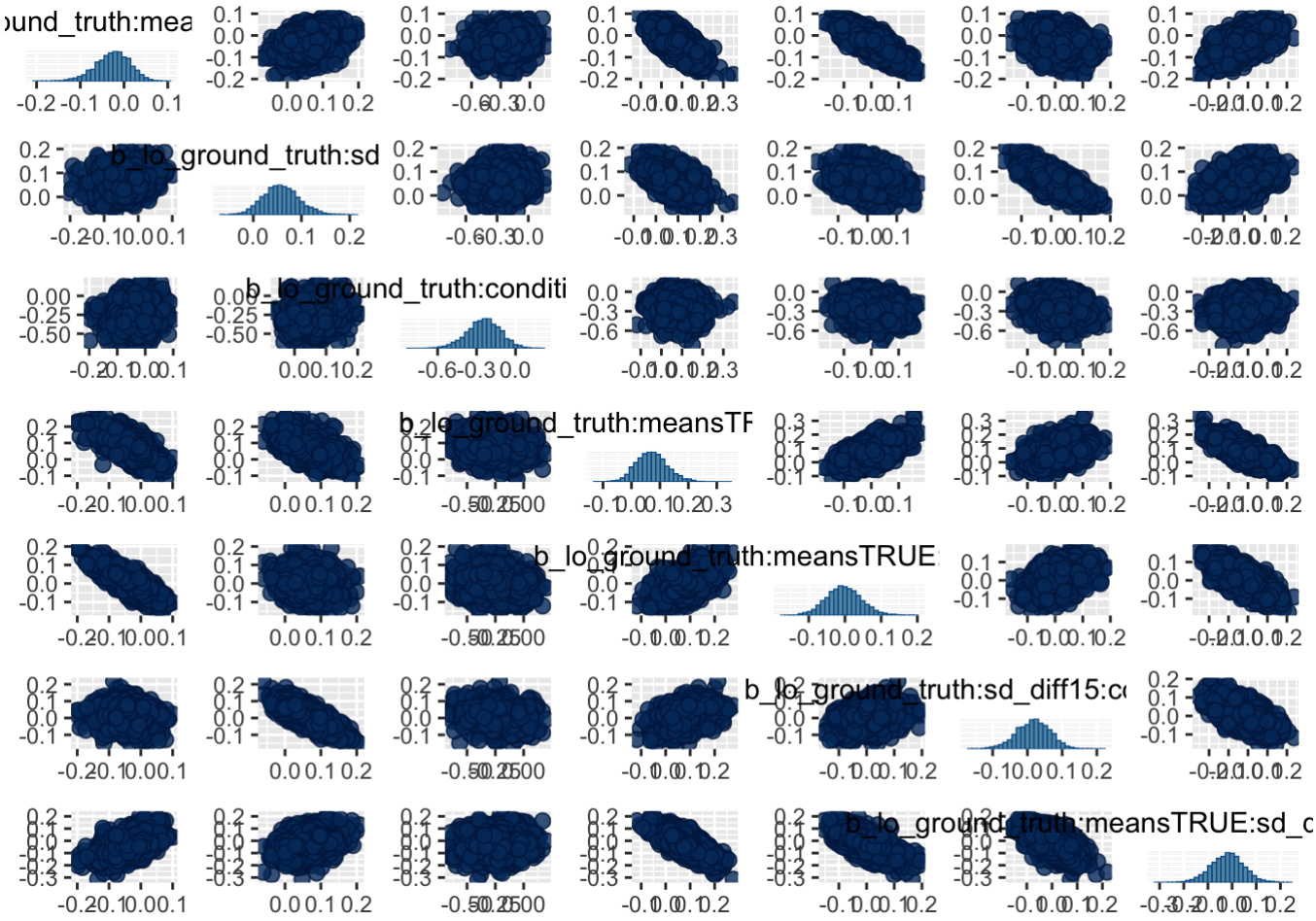

b_meansTRUE:sd_diff15:conditionintervals
b_lo_ground_truth:meansTRUE:sd_diff15:conditionintervals
sd_worker_id__Intercept
sd_worker_id__lo_ground_truth
sd_worker_id__sigma_Intercept
cor_worker_id__Intercept__lo_ground_truth
cor_worker_id__Intercept__sigma_Intercept
cor_worker_id__lo_ground_truth__sigma_Intercept
Chain
1
2

- Pairs plot

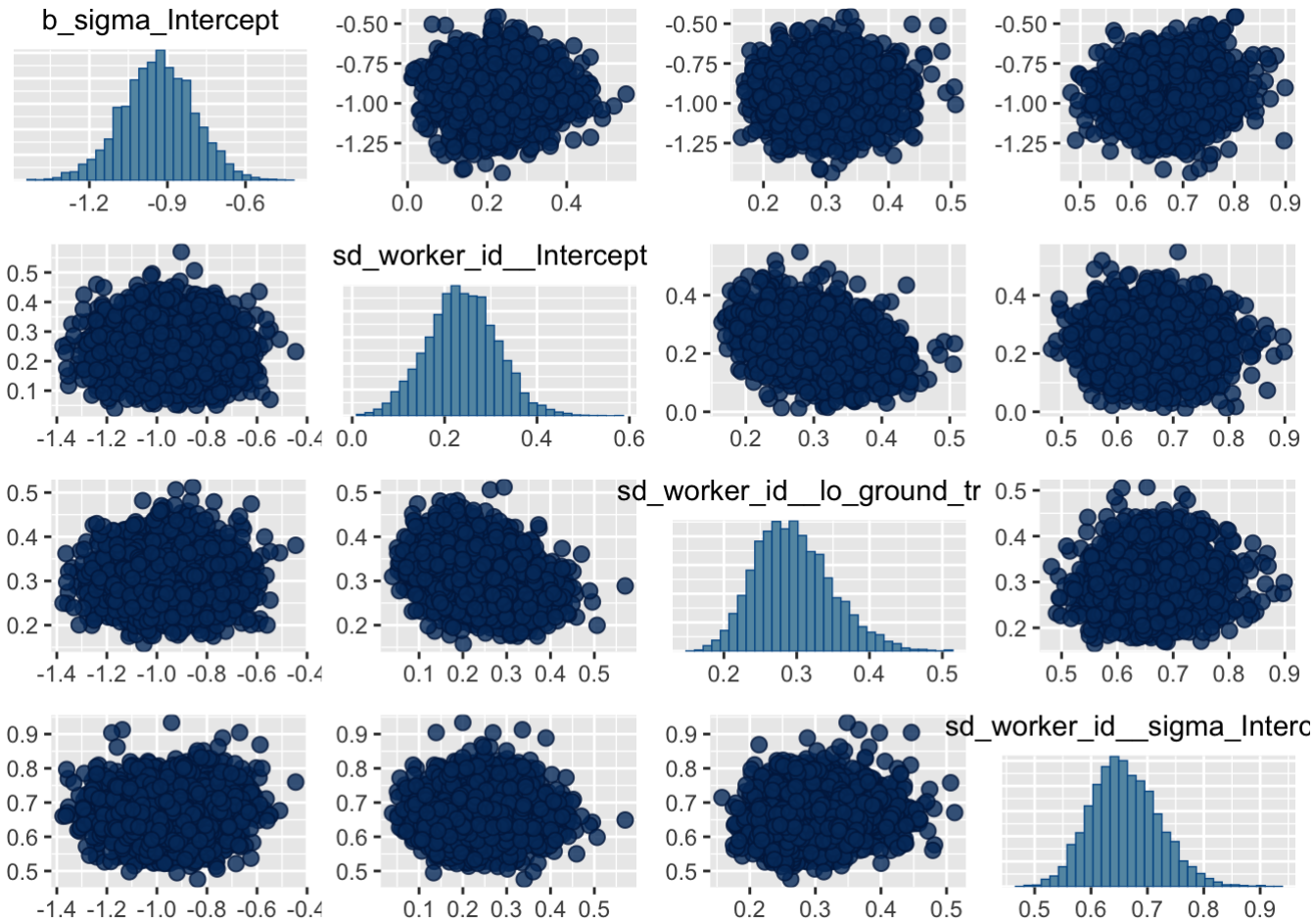
```
# pairs plot (intercepts)
pairs(m.wrkr.means.sd.vis.llo_p_sup, exact_match = TRUE, pars = c("b_Intercept",
                                                                  "b_lo_ground_truth",
                                                                  "b_meansTRUE",
                                                                  "b_sd_diff15",
                                                                  "b_conditionintervals"
,
                                                                  "b_meansTRUE:sd_diff1
5",
                                                                  "b_meansTRUE:condition
intervals",
                                                                  "b_sd_diff15:condition
intervals",
                                                                  "b_meansTRUE:sd_diff1
5:conditionintervals"))
```



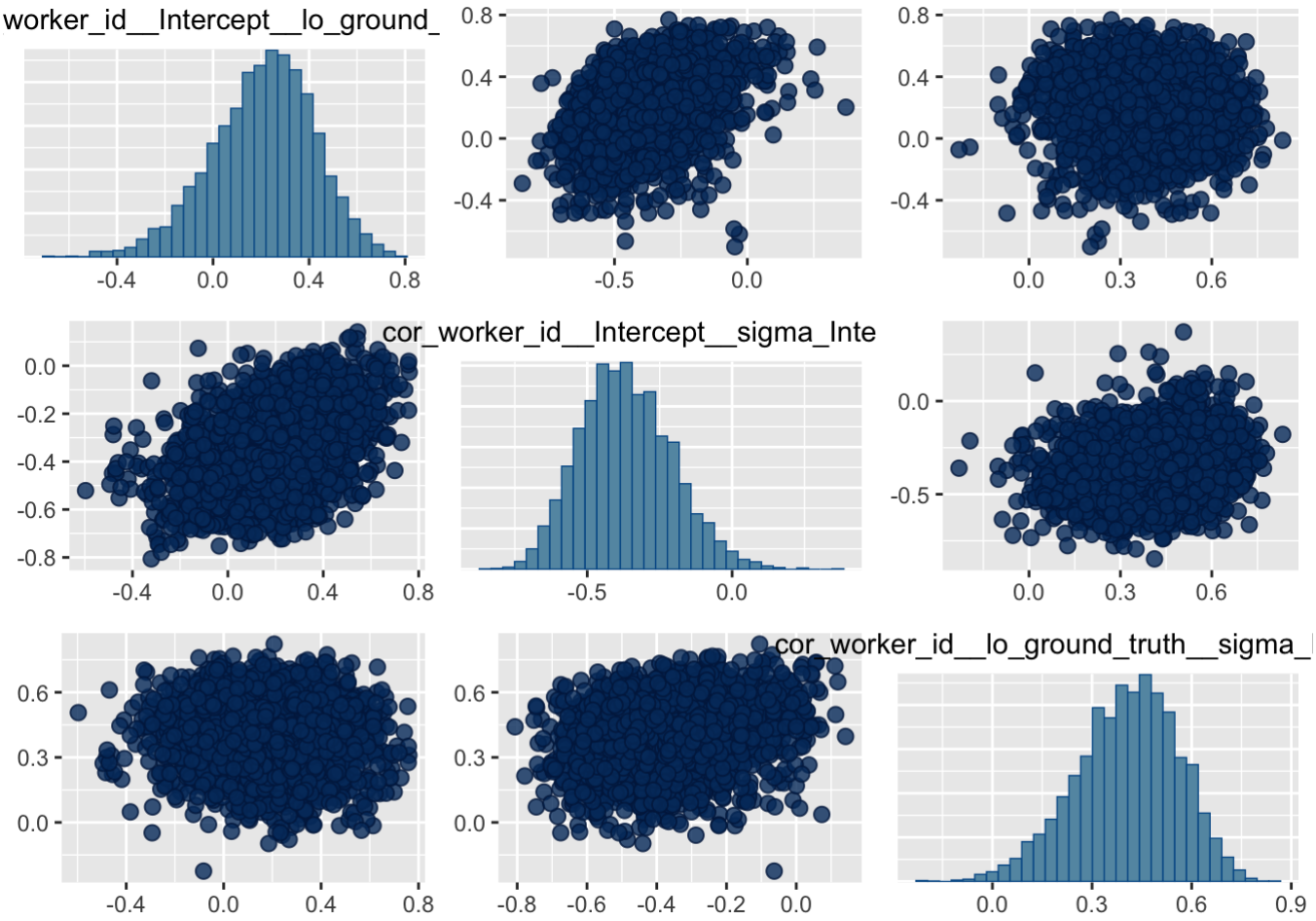
```
# pairs plot (LLO slopes)
pairs(m.wrkr.means.sd.vis.llo_p_sup, exact_match = TRUE, pars = c("b_lo_ground_truth:mea
nsTRUE",
                                                                  "b_lo_ground_truth:sd_
diff15",
                                                                  "b_lo_ground_truth:con
ditionintervals",
                                                                  "b_lo_ground_truth:mea
nsTRUE:sd_diff15",
                                                                  "b_lo_ground_truth:mea
nsTRUE:conditionintervals",
                                                                  "b_lo_ground_truth:sd_
diff15:conditionintervals",
                                                                  "b_lo_ground_truth:mea
nsTRUE:sd_diff15:conditionintervals"))
```




```
# pairs plot (random effects)
pairs(m.wrkr.means.sd.vis.llo_p_sup, exact_match = TRUE, pars = c("b_sigma_Intercept",
                                                                  "sd_worker_id__Interce
pt",
                                                                  "sd_worker_id__lo_grou
nd_truth",
                                                                  "sd_worker_id__sigma_I
ntercept"))
```




```
pairs(m.wrkr.means.sd.vis.llo_p_sup, pars = c("cor_worker_id__"))
```



- Summary

```
# model summary
print(m.wrkr.means.sd.vis.llo_p_sup)
```

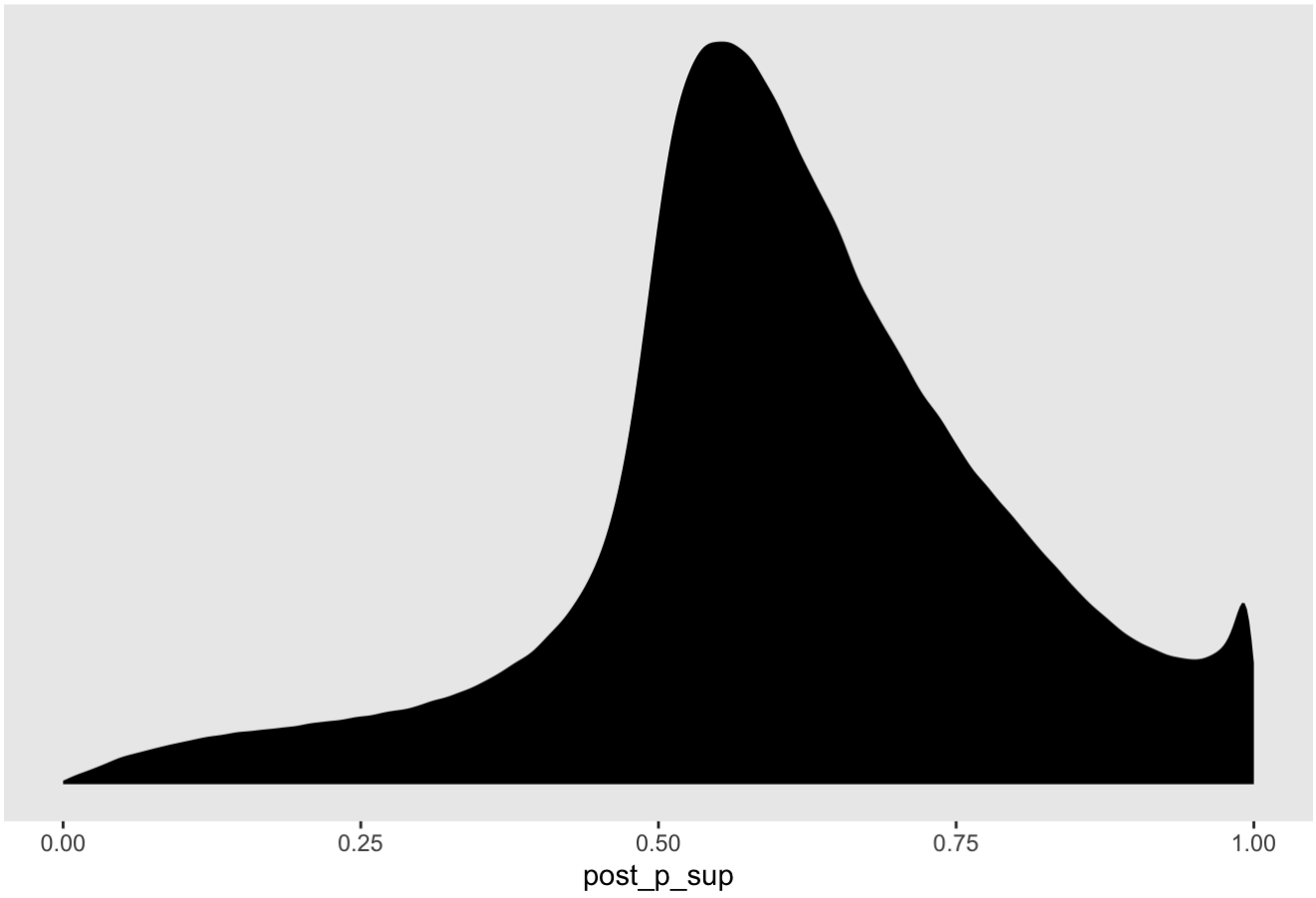
```
##  Family: gaussian
##   Links: mu = identity; sigma = log
## Formula: lo_p_sup ~ (1 + lo_ground_truth | sharecor | worker_id) + lo_ground_truth *
means * sd_diff * condition
##          sigma ~ (1 | sharecor | worker_id)
##    Data: model_df (Number of observations: 768)
## Samples: 2 chains, each with iter = 3000; warmup = 500; thin = 1;
##          total post-warmup samples = 5000
##
## Group-Level Effects:
## ~worker_id (Number of levels: 24)
##                                     Estimate Est.Error l-95% CI u-95% CI
## sd(Intercept)                           0.24      0.08     0.09     0.39
## sd(lo_ground_truth)                     0.30      0.05     0.21     0.41
## sd(sigma_Intercept)                     0.66      0.06     0.55     0.79
## cor(Intercept,lo_ground_truth)          0.21      0.22    -0.26     0.59
## cor(Intercept,sigma_Intercept)         -0.37      0.16    -0.65    -0.03
## cor(lo_ground_truth,sigma_Intercept)    0.41      0.15     0.08     0.68
##                                     Eff.Sample Rhat
## sd(Intercept)                              958 1.00
## sd(lo_ground_truth)                       1693 1.00
## sd(sigma_Intercept)                       3710 1.00
## cor(Intercept,lo_ground_truth)             948 1.00
## cor(Intercept,sigma_Intercept)            2243 1.00
## cor(lo_ground_truth,sigma_Intercept)      2366 1.00
##
## Population-Level Effects:
##                                                    Estimate Est.Error
## Intercept                                             -0.11      0.10
## sigma_Intercept                                       -0.94      0.14
## lo_ground_truth                                        0.60      0.10
## meansTRUE                                              0.04      0.06
## sd_diff15                                              0.09      0.06
## conditionintervals                                    -0.05      0.13
## lo_ground_truth:meansTRUE                             -0.03      0.04
## lo_ground_truth:sd_diff15                              0.06      0.04
## meansTRUE:sd_diff15                                   -0.07      0.08
## lo_ground_truth:conditionintervals                    -0.25      0.13
## meansTRUE:conditionintervals                          -0.01      0.07
## sd_diff15:conditionintervals                          -0.05      0.07
## lo_ground_truth:meansTRUE:sd_diff15                    0.07      0.05
## lo_ground_truth:meansTRUE:conditionintervals           -0.00      0.05
## lo_ground_truth:sd_diff15:conditionintervals            0.02      0.05
## meansTRUE:sd_diff15:conditionintervals                 0.14      0.10
## lo_ground_truth:meansTRUE:sd_diff15:conditionintervals -0.02      0.07
##                                                    l-95% CI u-95% CI
## Intercept                                             -0.30     0.08
## sigma_Intercept                                       -1.22    -0.67
## lo_ground_truth                                        0.41     0.79
## meansTRUE                                             -0.07     0.16
## sd_diff15                                             -0.02     0.20
## conditionintervals                                    -0.31     0.18
## lo_ground_truth:meansTRUE                             -0.11     0.04
```

```
## lo_ground_truth:sd_diff15                                    -0.01     0.13
## meansTRUE:sd_diff15                                          -0.23     0.08
## lo_ground_truth:conditionintervals                           -0.51    -0.00
## meansTRUE:conditionintervals                                 -0.15     0.13
## sd_diff15:conditionintervals                                 -0.19     0.09
## lo_ground_truth:meansTRUE:sd_diff15                          -0.03     0.19
## lo_ground_truth:meansTRUE:conditionintervals                 -0.10     0.10
## lo_ground_truth:sd_diff15:conditionintervals                 -0.08     0.10
## meansTRUE:sd_diff15:conditionintervals                       -0.06     0.33
## lo_ground_truth:meansTRUE:sd_diff15:conditionintervals       -0.15     0.11
##                                                          Eff.Sample Rhat
## Intercept                                                      3170 1.00
## sigma_Intercept                                                2509 1.00
## lo_ground_truth                                                2206 1.00
## meansTRUE                                                      2689 1.00
## sd_diff15                                                      3026 1.00
## conditionintervals                                             2339 1.00
## lo_ground_truth:meansTRUE                                      2513 1.00
## lo_ground_truth:sd_diff15                                      3145 1.00
## meansTRUE:sd_diff15                                            2629 1.00
## lo_ground_truth:conditionintervals                             1689 1.00
## meansTRUE:conditionintervals                                   2539 1.00
## sd_diff15:conditionintervals                                   2647 1.00
## lo_ground_truth:meansTRUE:sd_diff15                            2419 1.00
## lo_ground_truth:meansTRUE:conditionintervals                   2493 1.00
## lo_ground_truth:sd_diff15:conditionintervals                   2911 1.00
## meansTRUE:sd_diff15:conditionintervals                         2459 1.00
## lo_ground_truth:meansTRUE:sd_diff15:conditionintervals         2404 1.00
##
## Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
## is a crude measure of effective sample size, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

Let's check our posterior predictive distribution.

```
# posterior predictive check
model_df %>%
  select(lo_ground_truth, worker_id, means, sd_diff, condition) %>%
  add_predicted_draws(m.wrkr.means.sd.vis.llo_p_sup, prediction = "lo_p_sup", seed = 123
4) %>%
  mutate(
    # transform to probability units
    post_p_sup = plogis(lo_p_sup)
  ) %>%
  ggplot(aes(x = post_p_sup)) +
  geom_density(fill = "black", size = 0) +
  scale_y_continuous(NULL, breaks = NULL) +
  labs(subtitle = "Posterior predictive distribution for probability of superiority") +
  theme(panel.grid = element_blank())
```

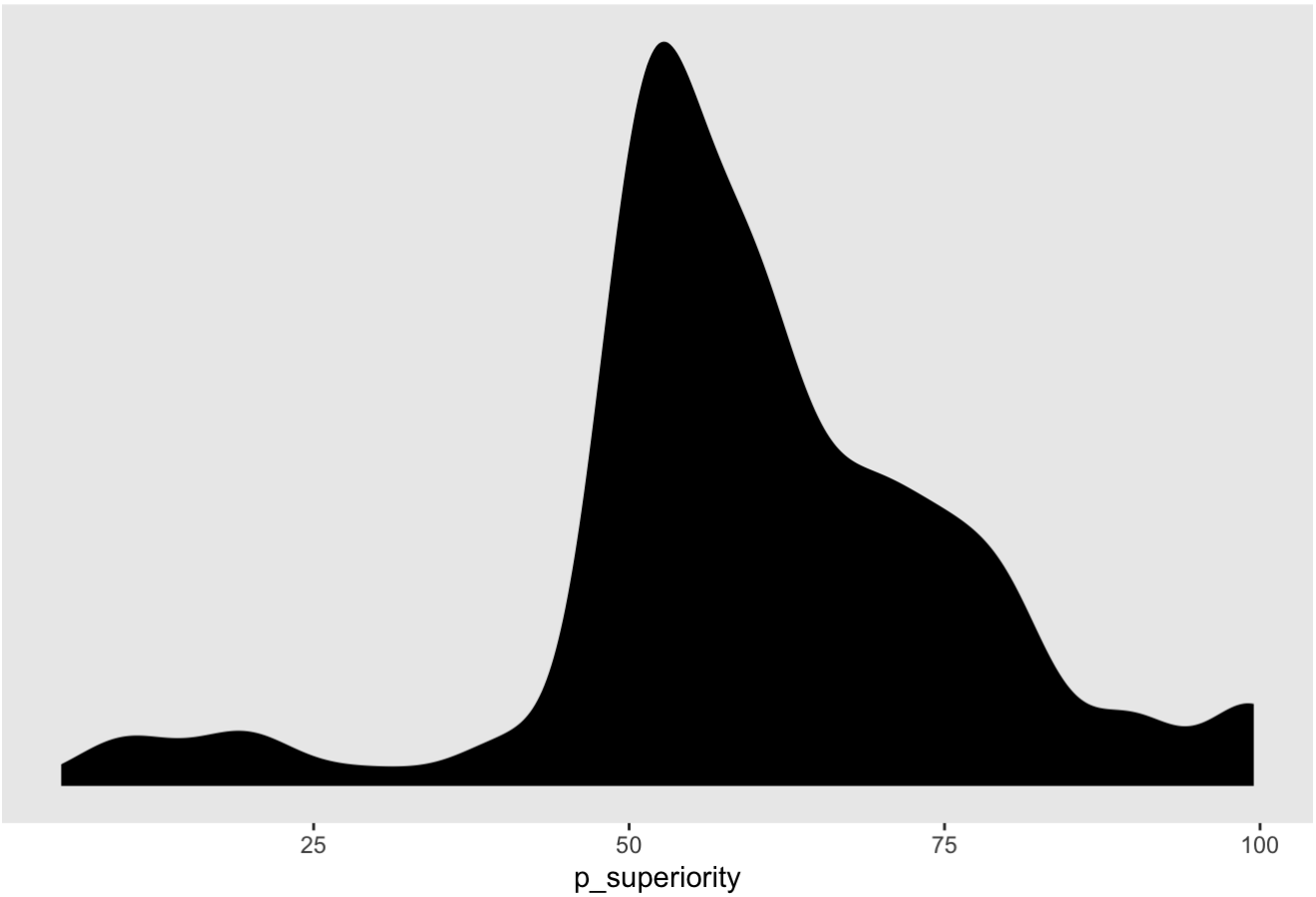
Posterior predictive distribution for probability of superiority



How do these predictions compare to the observed data?

```
# data density
model_df %>%
  ggplot(aes(x = p_superiority)) +
  geom_density(fill = "black", size = 0) +
  scale_y_continuous(NULL, breaks = NULL) +
  labs(subtitle = "Data distribution for probability of superiority") +
  theme(panel.grid = element_blank())
```

Data distribution for probability of superiority



Running a leave one out posterior predictive check, we can see that overall this model has decent predictive validity.

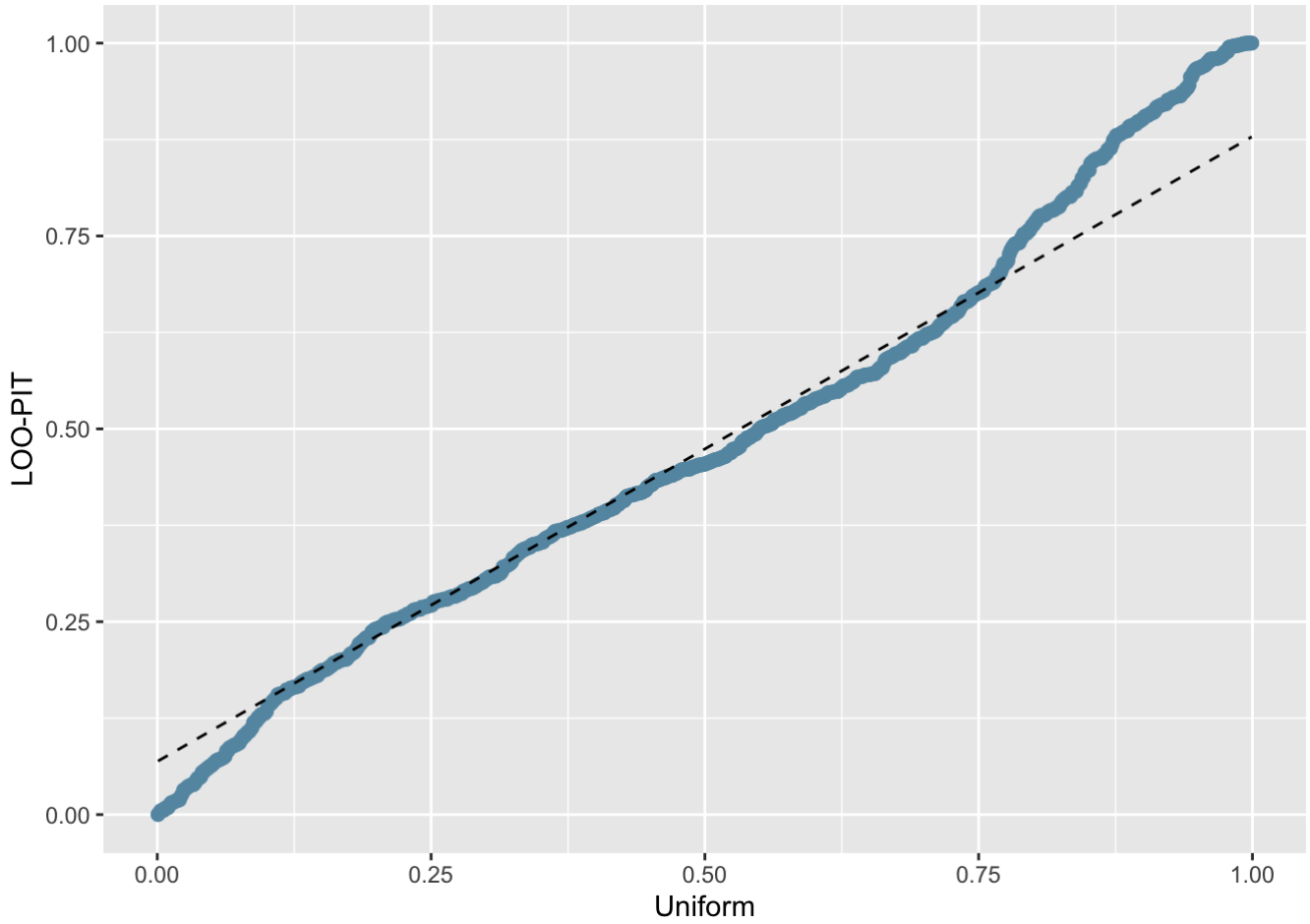
```
# set up data for LOO posterior predictive check
y <- model_df$lo_p_sup
yrep <- posterior_predict(m.wrkr.means.sd.vis.llo_p_sup)

# run LOO to get weights
loo <- loo(m.wrkr.means.sd.vis.llo_p_sup, save_psis = TRUE, cores = 2)
```

```
## Warning: Found 9 observations with a pareto_k > 0.7 in model
## 'm.wrkr.means.sd.vis.llo_p_sup'. It is recommended to set 'reloo = TRUE' in
## order to calculate the ELPD without the assumption that these observations
## are negligible. This will refit the model 9 times to compute the ELPDs for
## the problematic observations directly.
```

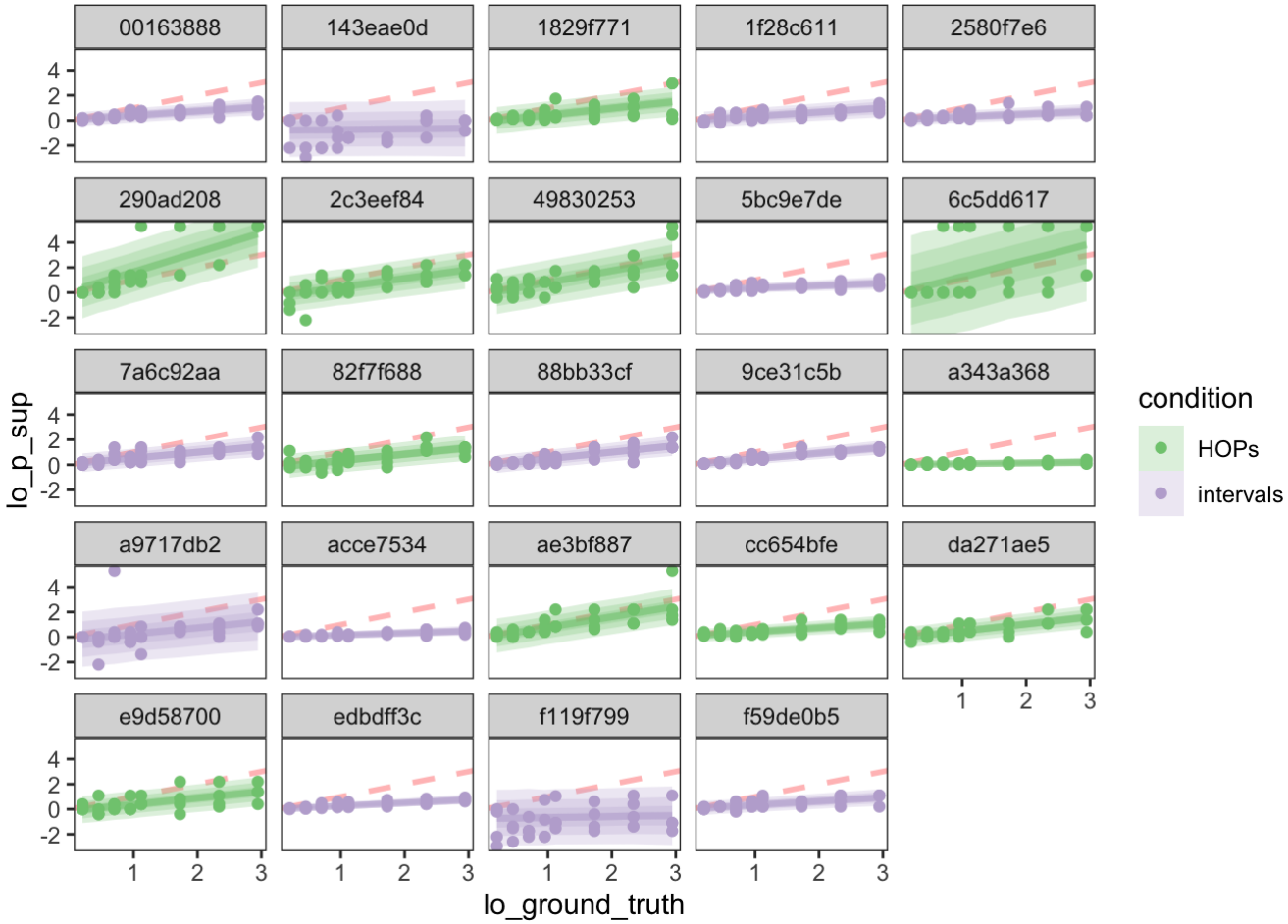
```
psis <- loo$psis_object
lw <- weights(psis)
```

```
ppc_loo_pit_qq(y, yrep, lw = lw)
```



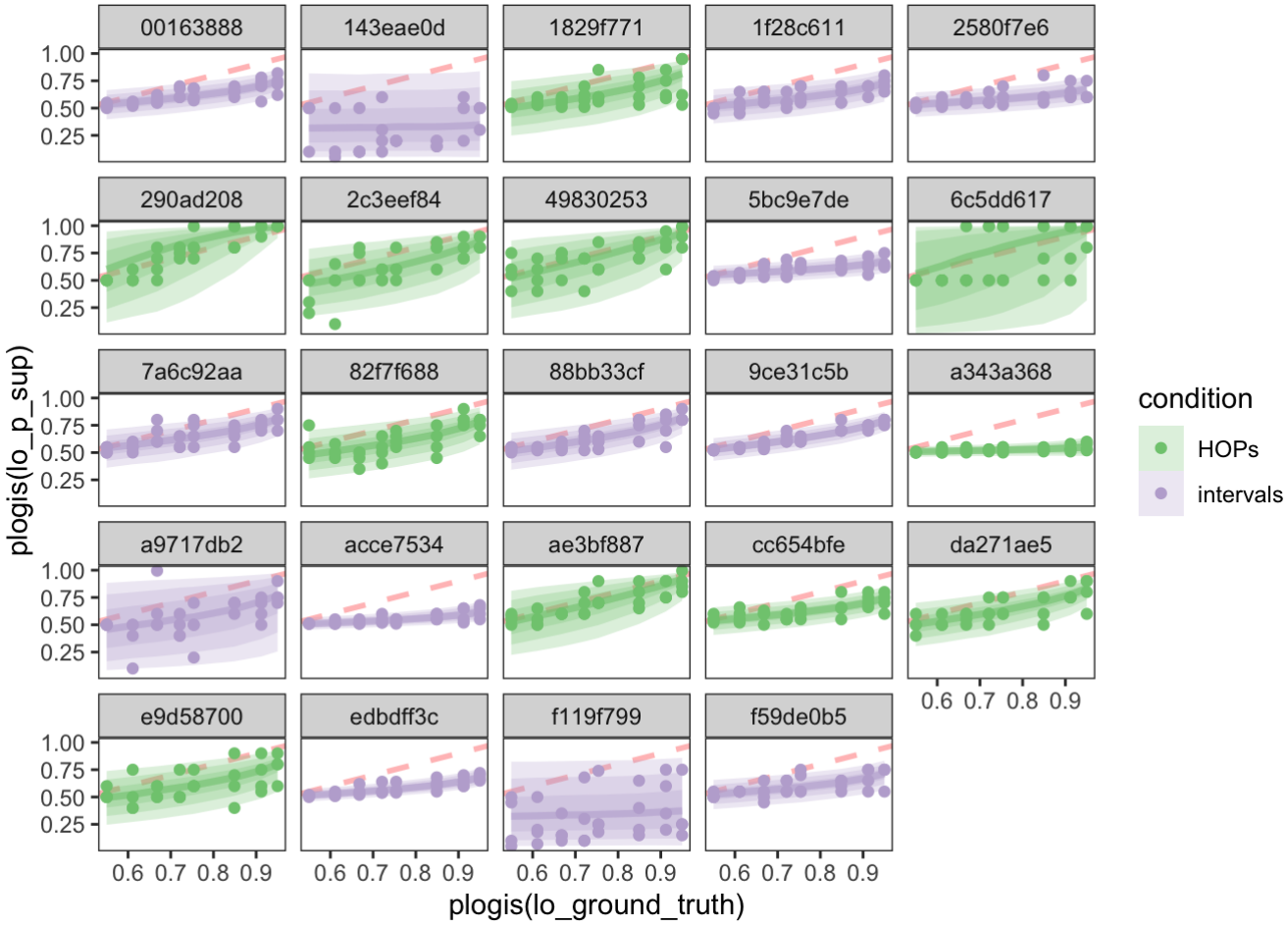
Let's take a look at predictions per worker and visualization condition to get a more granular sense of our model fit.

```
model_df %>%
  group_by(lo_ground_truth, worker_id, means, sd_diff, condition) %>%
  add_predicted_draws(m.wrkr.means.sd.vis.llo_p_sup) %>%
  ggplot(aes(x = lo_ground_truth, y = lo_p_sup, color = condition, fill = condition)) +
  geom_abline(intercept = 0, slope = 1, size = 1, alpha = .3, color = "red", linetype =
"dashed") + # ground truth
  stat_lineribbon(aes(y = .prediction), .width = c(.95, .80, .50), alpha = .25) +
  geom_point(data = model_df) +
  scale_fill_brewer(type = "qual", palette = 1) +
  scale_color_brewer(type = "qual", palette = 1) +
  coord_cartesian(xlim = quantile(model_df$lo_ground_truth, c(0, 1)),
                  ylim = quantile(model_df$lo_p_sup, c(0, 1))) +
  theme_bw() +
  theme(panel.grid = element_blank()) +
  facet_wrap(~ worker_id)
```



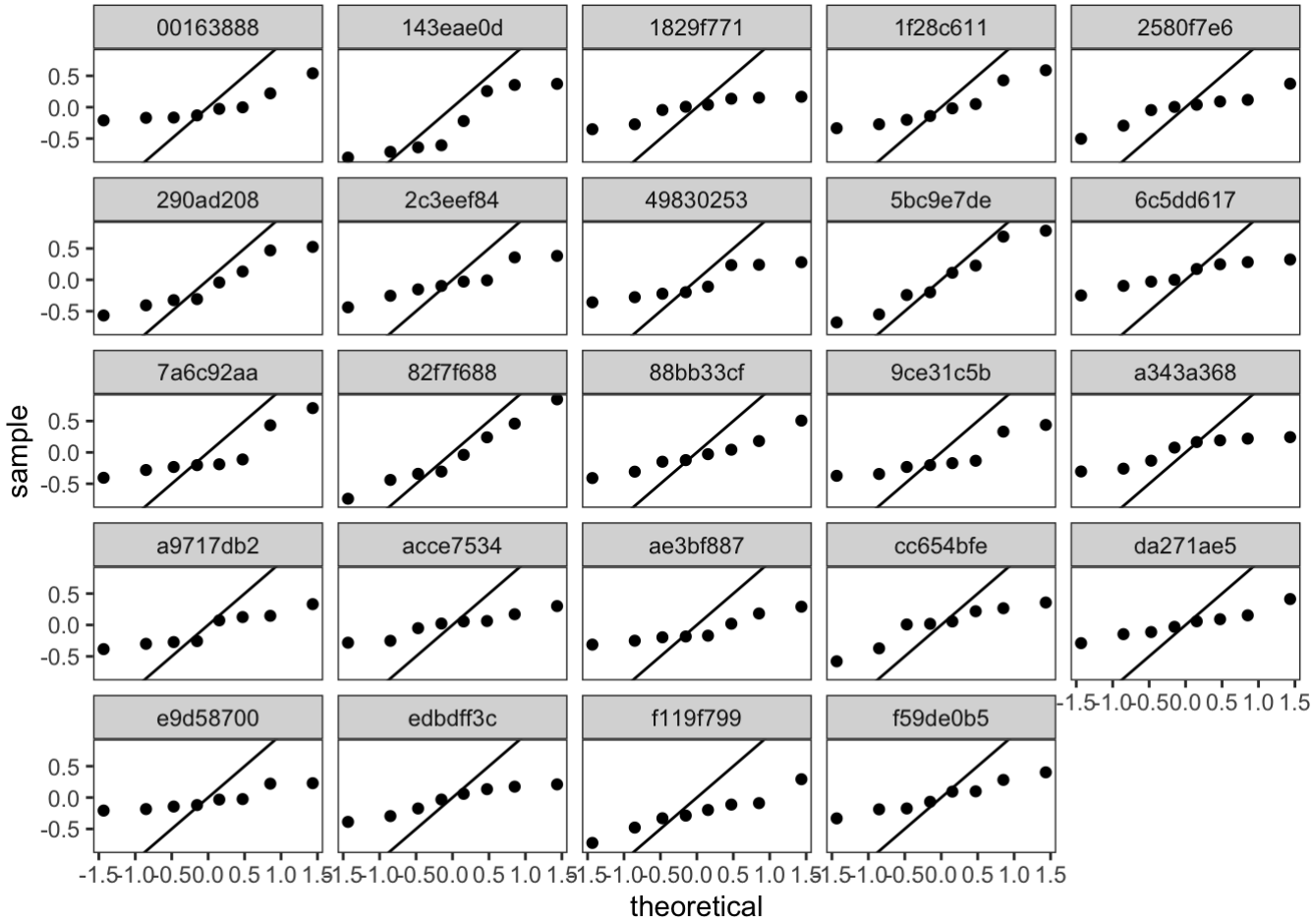
What does this look like in probability units?

```
model_df %>%
  group_by(lo_ground_truth, worker_id, means, sd_diff, condition) %>%
  add_predicted_draws(m.wrkr.means.sd.vis.llo_p_sup) %>%
  ggplot(aes(x = plogis(lo_ground_truth), y = plogis(lo_p_sup), color = condition, fill
= condition)) +
  geom_abline(intercept = 0, slope = 1, size = 1, alpha = .3, color = "red", linetype =
"dashed") + # ground truth
  stat_lineribbon(aes(y = plogis(.prediction)), .width = c(.95, .80, .50), alpha = .25)
 +
  geom_point(data = model_df) +
  scale_fill_brewer(type = "qual", palette = 1) +
  scale_color_brewer(type = "qual", palette = 1) +
  coord_cartesian(xlim = quantile(plogis(model_df$lo_ground_truth), c(0, 1)),
                  ylim = quantile(plogis(model_df$lo_p_sup), c(0, 1))) +
  theme_bw() +
  theme(panel.grid = element_blank()) +
  facet_wrap(~ worker_id)
```



To examine more closely whether our model has predictive validity at the level of each worker, we'll look at QQ plots for residuals at the worker level.

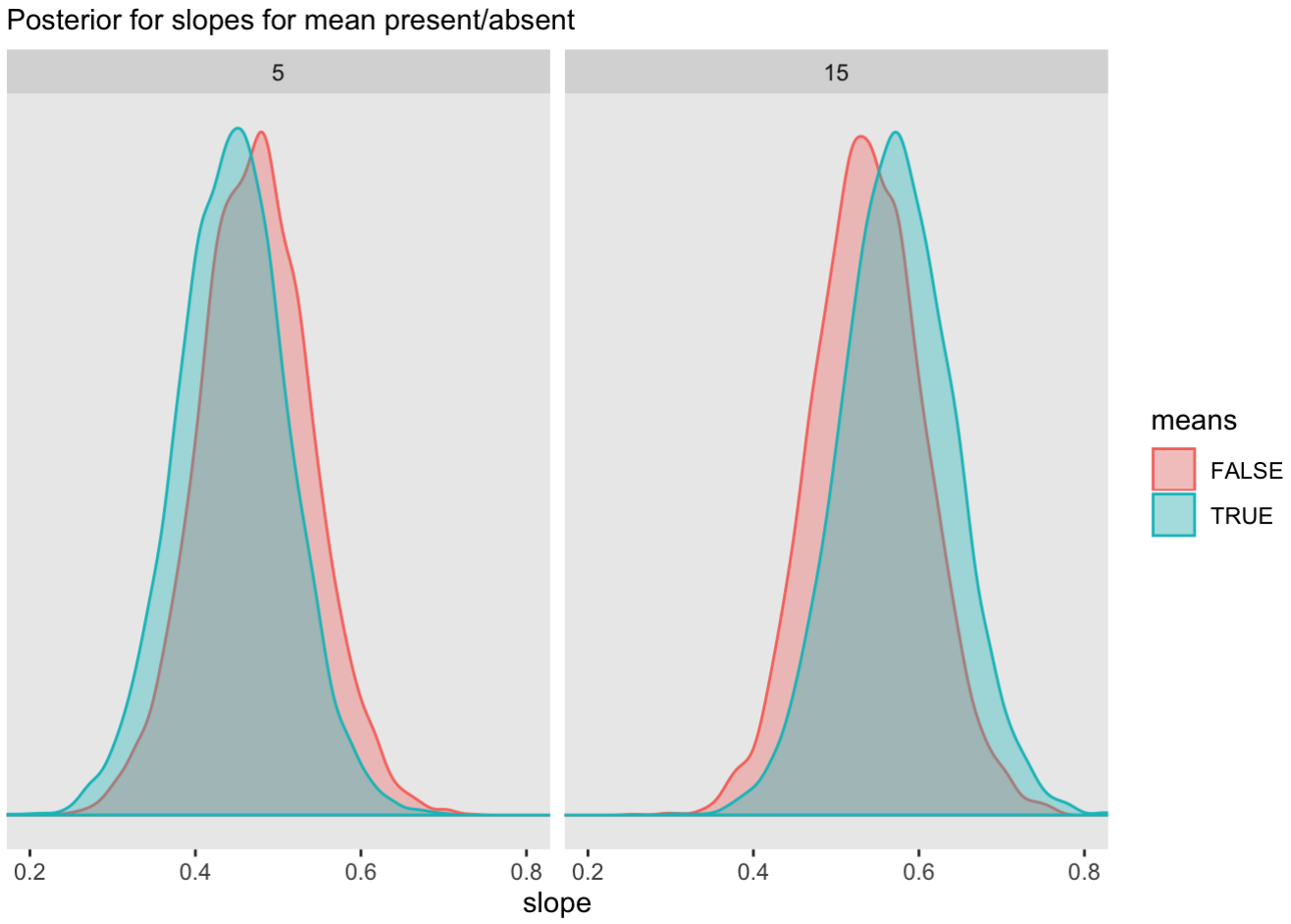
```
model_df %>%
  add_predicted_draws(m.wrkr.means.sd.vis.llo_p_sup) %>%
  group_by(lo_ground_truth, worker_id) %>%
  summarise(
    p_residual = mean(.prediction < lo_p_sup), # what proportion of predicted judgments
 are less than the observed response?
    z_residual = qnorm(p_residual)             # what are the z-scores of these cumulati
ve probabilities?
  ) %>%
  ggplot(aes(sample = z_residual)) +
  geom_qq() +
  geom_abline() +
  theme_bw() +
  theme(panel.grid = element_blank()) +
  facet_wrap(~ worker_id)
```



These still look pretty terrible.

What does the posterior for the slope of the LLO model look like when means are present vs absent at different levels of uncertainty, ignoring other manipulations?

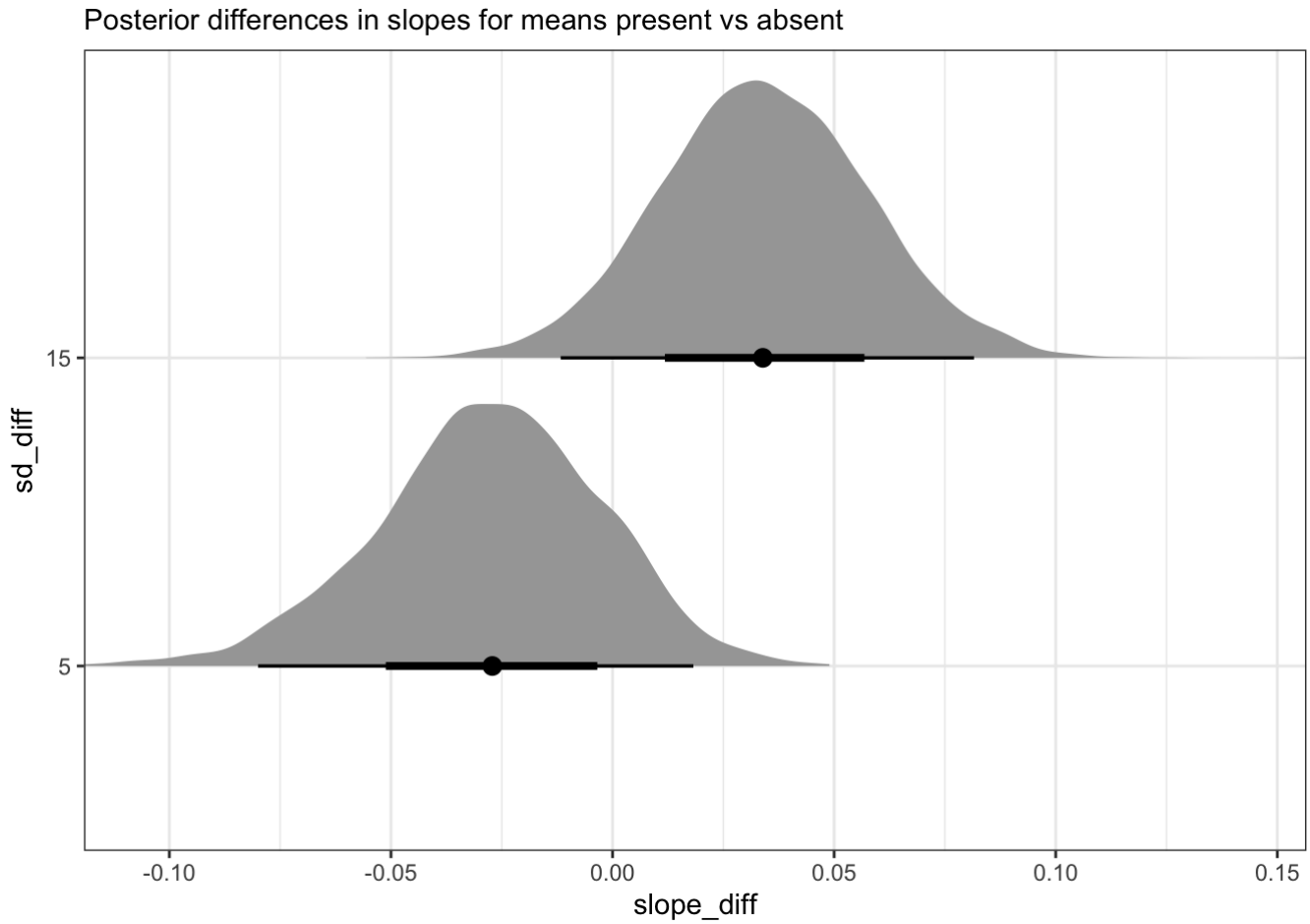
```
model_df %>%
  group_by(means, sd_diff, condition) %>%
  data_grid(lo_ground_truth = c(0, 1)) %>%          # get fitted draws (in log odds unit
s) only for ground truth of 0 and 1
  add_fitted_draws(m.wrkr.means.sd.vis.llo_p_sup, re_formula = NA) %>%
  compare_levels(.value, by = lo_ground_truth) %>%  # calculate the difference between f
its at 1 and 0 (i.e., slope)
  rename(slope = .value) %>%
  group_by(means, sd_diff, .draw) %>%               # group by predictors to keep
  summarise(slope = weighted.mean(slope)) %>%       # marginalize out visualization cond
ition by taking a weighted average
  ggplot(aes(x = slope, group = means, color = means, fill = means)) +
  geom_density(alpha = 0.35) +
  scale_x_continuous(expression(slope), expand = c(0, 0)) +
  scale_y_continuous(NULL, breaks = NULL) +
  labs(subtitle = "Posterior for slopes for mean present/absent") +
  theme(panel.grid = element_blank()) +
  facet_grid(. ~ sd_diff)
```



This effect suggests that adding means has a debiasing effect on average when visualized uncertainty is high and a biasing effect when uncertainty is low (marginalizing across visualization conditions). Again, is exactlty what we expected to see.

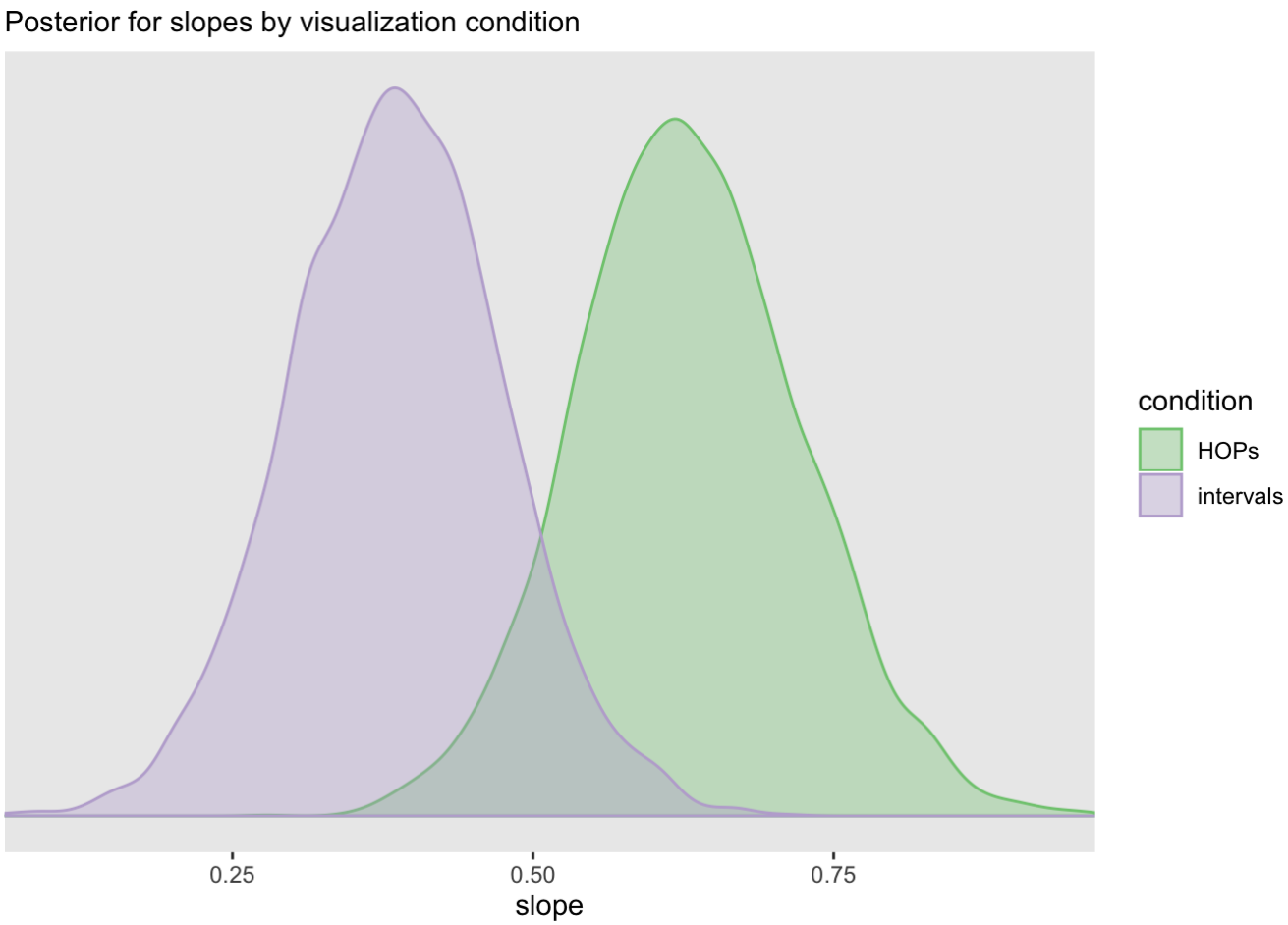
Let's look at this difference in a forest plot style display.

```
model_df %>%
  group_by(means, sd_diff, condition) %>%
  data_grid(lo_ground_truth = c(0, 1)) %>%                    # get fitted draws (in log
odds units) only for ground truth of 0 and 1
  add_fitted_draws(m.wrkr.means.sd.vis.llo_p_sup, re_formula = NA) %>%
  compare_levels(.value, by = lo_ground_truth) %>%            # calculate the difference
between fits at 1 and 0 (i.e., slope)
  compare_levels(.value, by = means) %>%                      # look at differences in s
lopes between means present vs absent
  rename(slope_diff = .value) %>%
  group_by(sd_diff, .draw) %>%                                # group by predictors to k
eep
  summarise(slope_diff = weighted.mean(slope_diff)) %>%       # marginalize out means pr
esent/absent by taking a weighted average
  ggplot(aes(x = slope_diff, y = sd_diff)) +
  stat_halfeyeh() +
  scale_x_continuous(expression(slope_diff), expand = c(0, 0)) +
  labs(subtitle = "Posterior differences in slopes for means present vs absent") +
  theme_bw()
```



What does the posterior for the slope in each visualization condition look like, marginalizing across other factors?

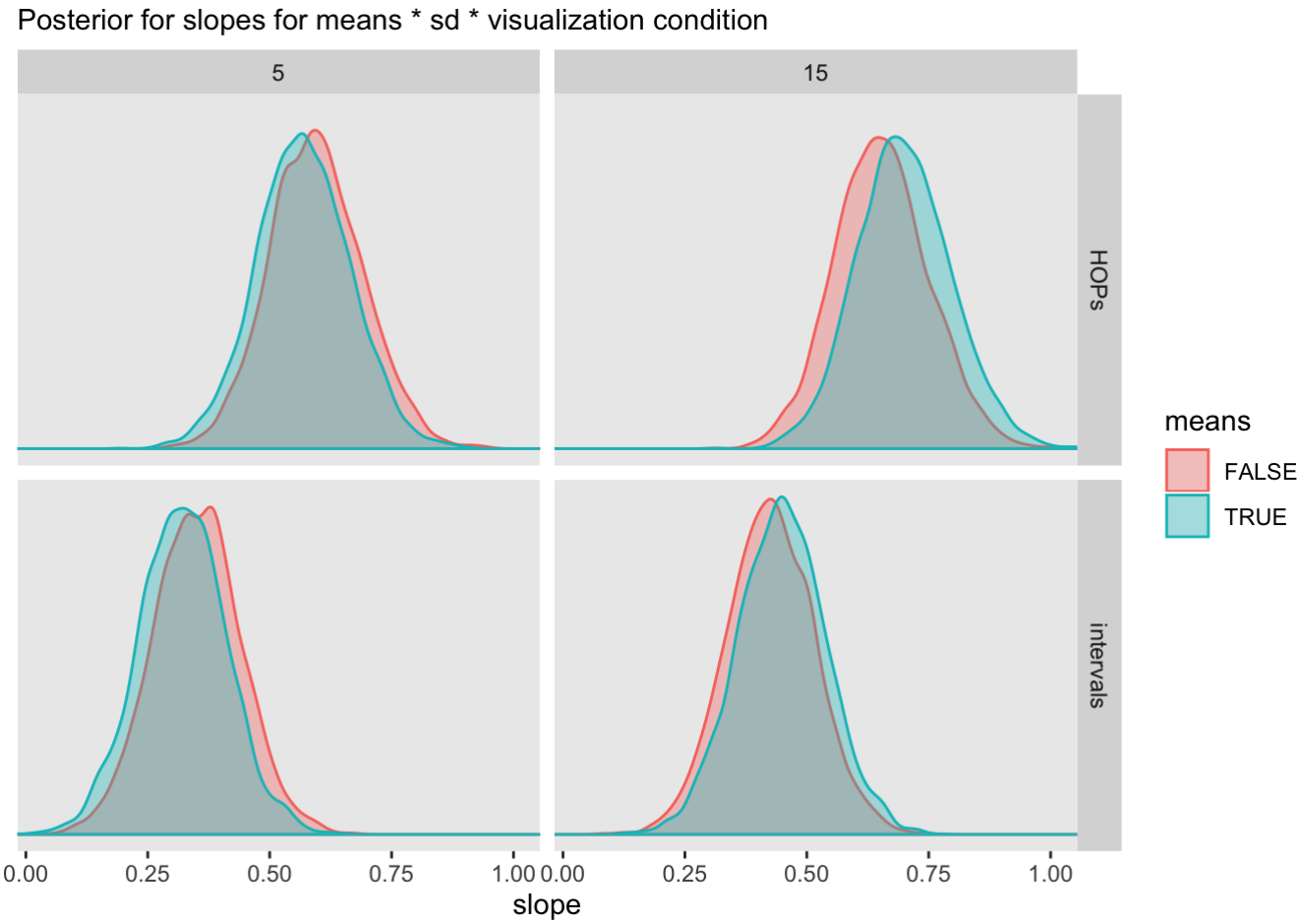
```
model_df %>%
  group_by(means, sd_diff, condition) %>%
  data_grid(lo_ground_truth = c(0, 1)) %>%          # get fitted draws (in log odds unit
s) only for ground truth of 0 and 1
  add_fitted_draws(m.wrkr.means.sd.vis.llo_p_sup, re_formula = NA) %>%
  compare_levels(.value, by = lo_ground_truth) %>%  # calculate the difference between f
its at 1 and 0 (i.e., slope)
  rename(slope = .value) %>%
  group_by(condition, .draw) %>%                    # group by predictors to keep
  summarise(slope = weighted.mean(slope)) %>%       # marginalize out means present/abse
nt by taking a weighted average
  ggplot(aes(x = slope, group = condition, color = condition, fill = condition)) +
  geom_density(alpha = 0.35) +
  scale_fill_brewer(type = "qual", palette = 1) +
  scale_color_brewer(type = "qual", palette = 1) +
  scale_x_continuous(expression(slope), expand = c(0, 0)) +
  scale_y_continuous(NULL, breaks = NULL) +
  labs(subtitle = "Posterior for slopes by visualization condition") +
  theme(panel.grid = element_blank())
```



Recall that a slope of 1 on the logit scale reflects no bias. This suggests that users are biased toward responses of 50% on the probability scale in both conditions, but especially with intervals. HOPs seem to have a substantial debiasing effect on effect size judgments when we marginalize across other manipulations.

What if we break these marginal effects down into simple effects for the interaction of the presence/absence of the mean, level of visualized uncertainty, and visualization condition?

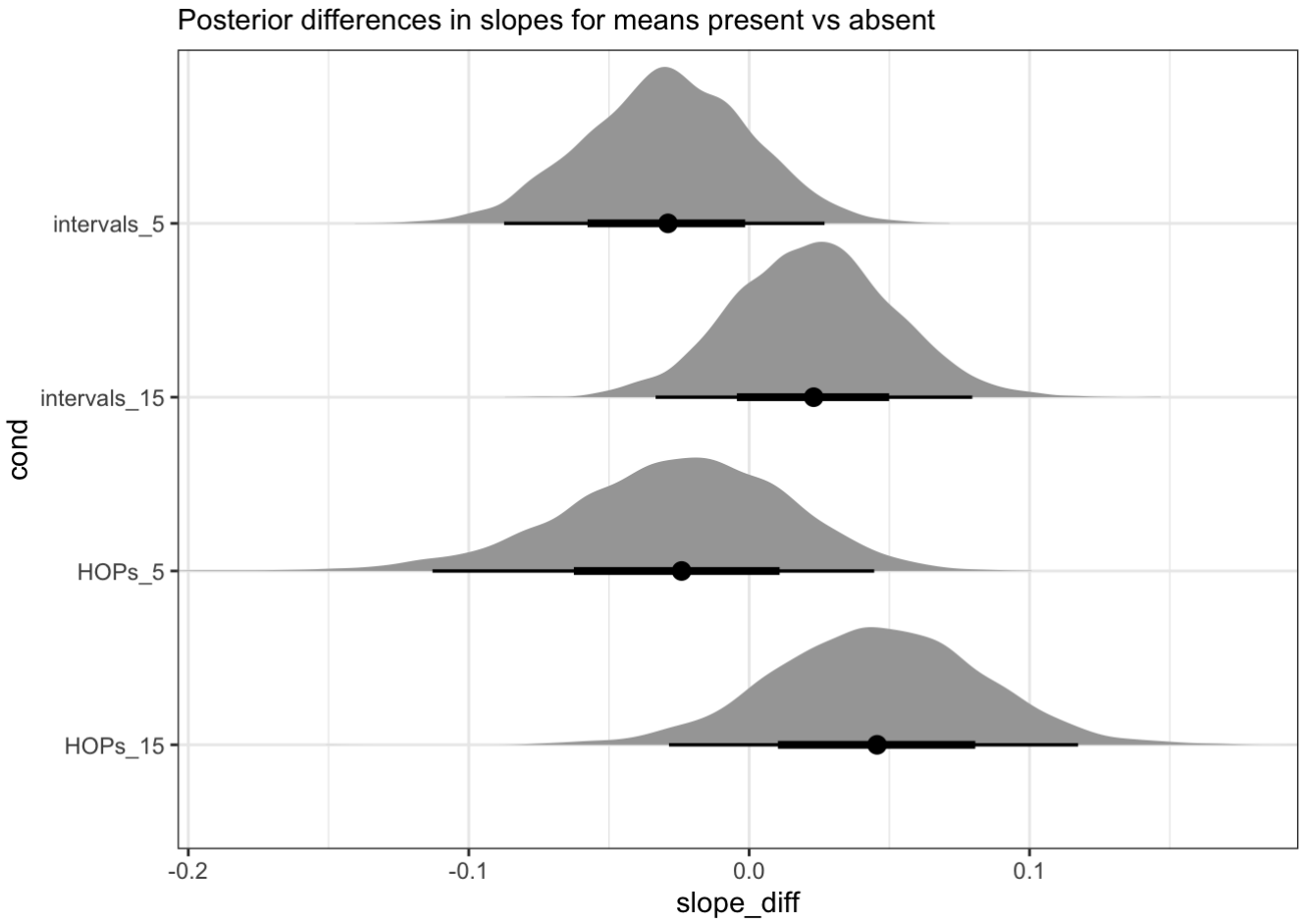
```
model_df %>%
  group_by(means, sd_diff, condition) %>%
  data_grid(lo_ground_truth = c(0, 1)) %>%          # get fitted draws (in log odds units) only for ground truth of 0 and 1
  add_fitted_draws(m.wrkr.means.sd.vis.llo_p_sup, re_formula = NA) %>%
  compare_levels(.value, by = lo_ground_truth) %>%  # calculate the difference between fits at 1 and 0 (i.e., slope)
  rename(slope = .value) %>%
  ggplot(aes(x = slope, group = means, color = means, fill = means)) +
  geom_density(alpha = 0.35) +
  scale_x_continuous(expression(slope), expand = c(0, 0)) +
  scale_y_continuous(NULL, breaks = NULL) +
  labs(subtitle = "Posterior for slopes for means * sd * visualization condition") +
  theme(panel.grid = element_blank()) +
  facet_grid(condition ~ sd_diff)
```



Again, this is what we expected to see. However, it is not clear form this chart if the effect is reliable.

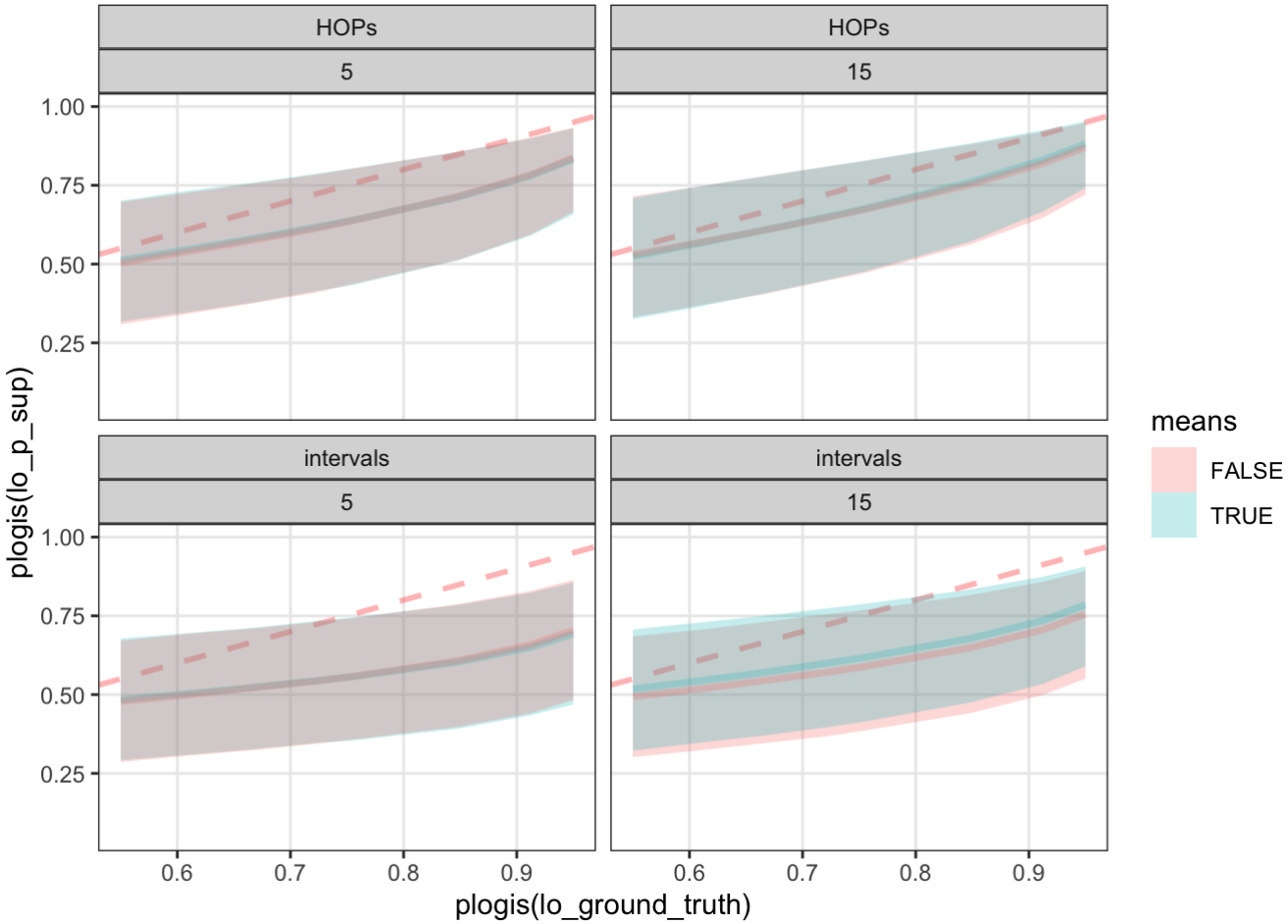
Let's look at the differences in a forest plot style display.

```
model_df %>%
  group_by(means, sd_diff, condition) %>%
  data_grid(lo_ground_truth = c(0, 1)) %>%          # get fitted draws (in log odds unit
s) only for ground truth of 0 and 1
  add_fitted_draws(m.wrkr.means.sd.vis.llo_p_sup, re_formula = NA) %>%
  compare_levels(.value, by = lo_ground_truth) %>%  # calculate the difference between f
its at 1 and 0 (i.e., slope)
  compare_levels(.value, by = means) %>%                 # look at differences in slopes
between means present vs absent
  rename(slope_diff = .value) %>%
  unite(cond, condition, sd_diff, sep = "_", remove = FALSE) %>%
  ggplot(aes(x = slope_diff, y = cond)) +
  stat_halfeyeh() +
  scale_x_continuous(expression(slope_diff), expand = c(0, 0)) +
  labs(subtitle = "Posterior differences in slopes for means present vs absent") +
  theme_bw()
```



What is the predicted pattern for responses for the average worker in each cell of this interaction?

```
model_df %>%
  group_by(lo_ground_truth, means, sd_diff, condition) %>%
  add_predicted_draws(m.wrkr.means.sd.vis.llo_p_sup, re_formula = NA) %>%
  ggplot(aes(x = plogis(lo_ground_truth), y = plogis(lo_p_sup), color = means, fill = me
ans)) +
  geom_abline(intercept = 0, slope = 1, size = 1, alpha = .3, color = "red", linetype =
"dashed") + # ground truth
  stat_lineribbon(aes(y = plogis(.prediction)), .width = c(.95), alpha = .25) +
  coord_cartesian(xlim = quantile(plogis(model_df$lo_ground_truth), c(0, 1)),
                  ylim = quantile(plogis(model_df$lo_p_sup), c(0, 1))) +
  theme_bw() +
  theme(panel.grid.minor = element_blank()) +
  facet_wrap(condition ~ sd_diff)
```



Next, we'll try to get more precise estimates by expanding our random effects and

Building Up Random Effects for Within-Subjects Manipulations

In the minimal model to answer our research questions above, estimates for the effect of means are noisier than we would like, and predictive validity within subjects is not great. We'll try to better account for heterogeneity across subjects by adding more random effects to our model for each within subjects manipulation.

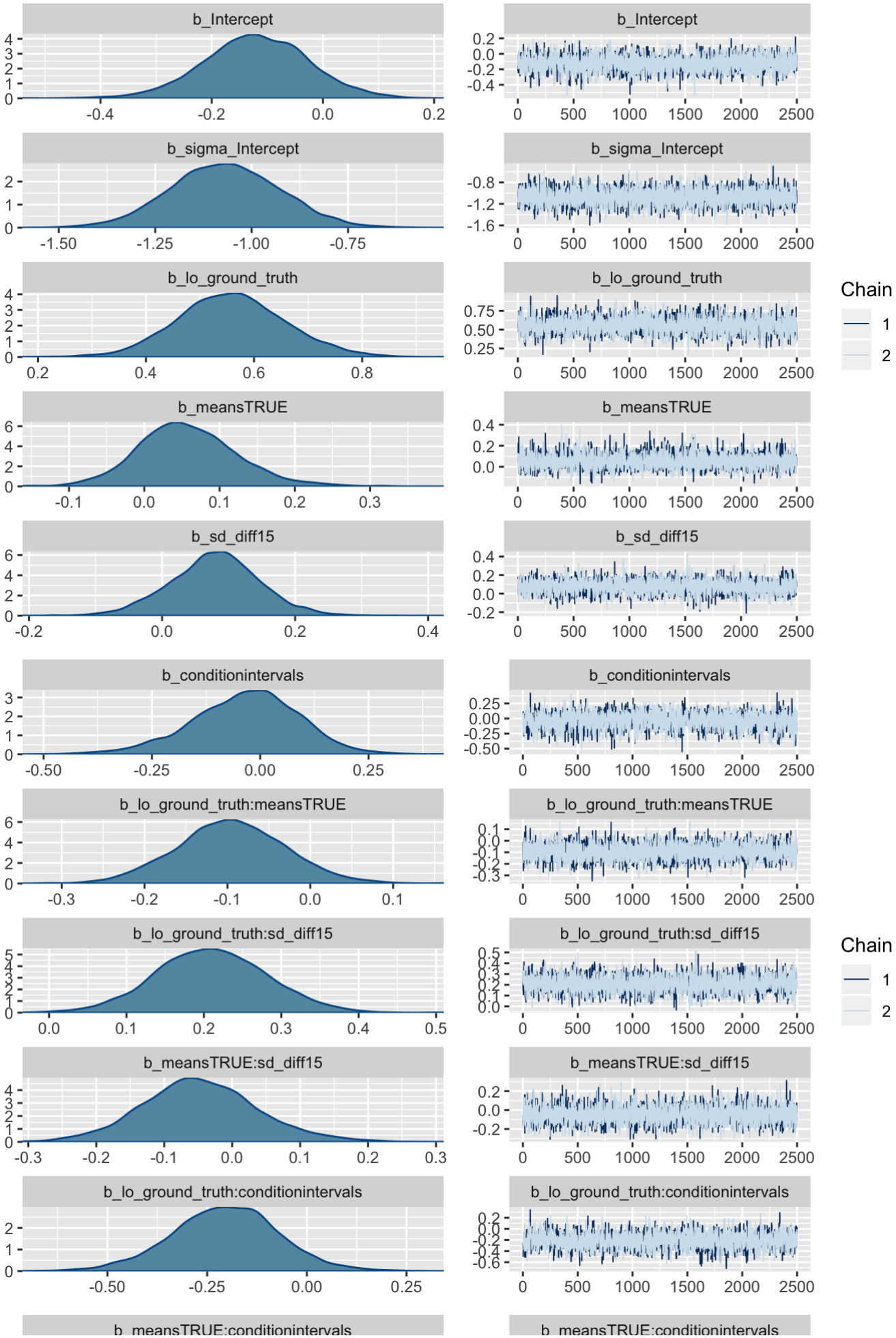
Following a principle of model expansion, we will make this changes cumulatively. We include a series of model specifications that capture plausible structure in the data and fit without any sampling issues.

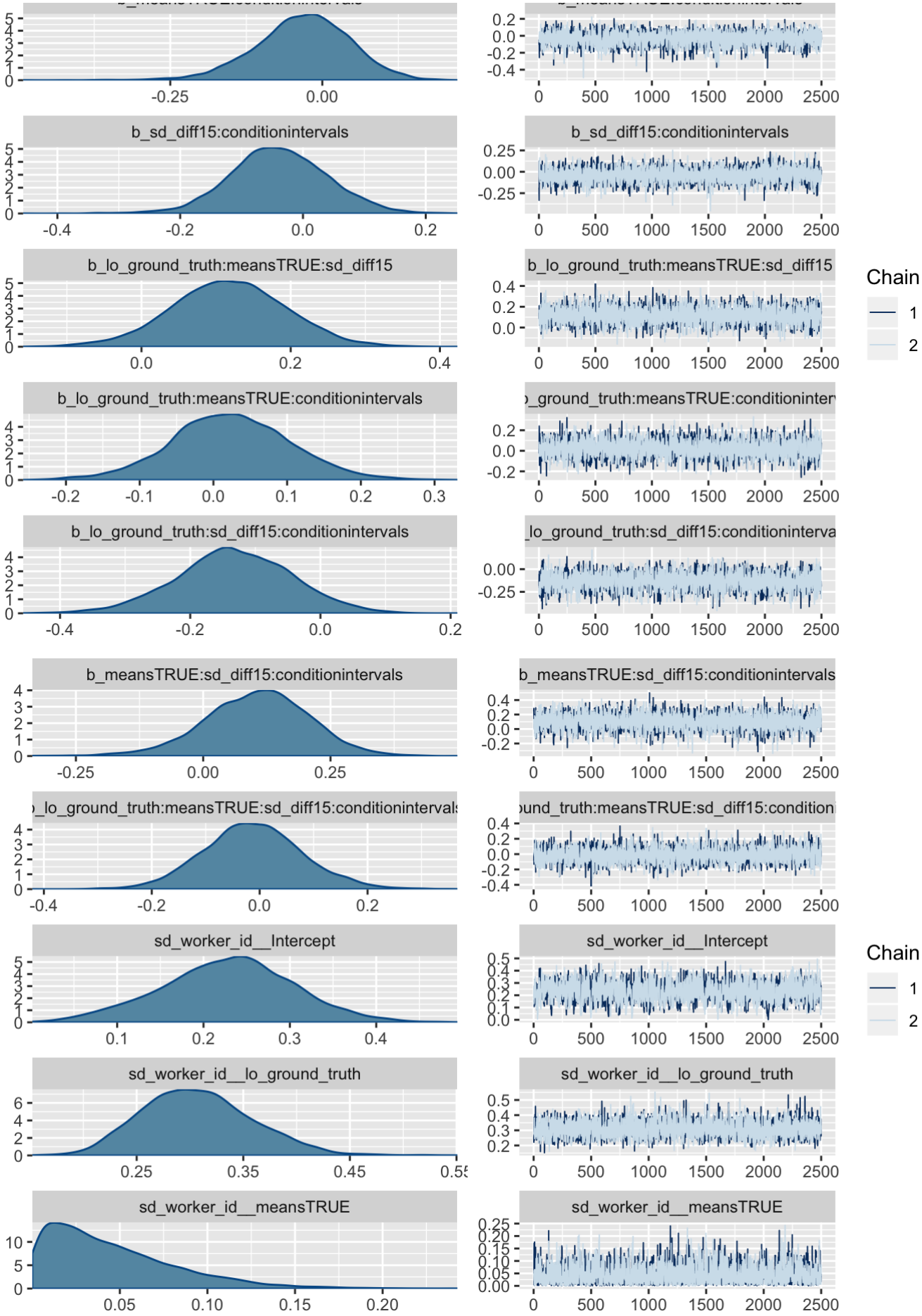
Random Effects for the Interaction of Means and Uncertainty Shown

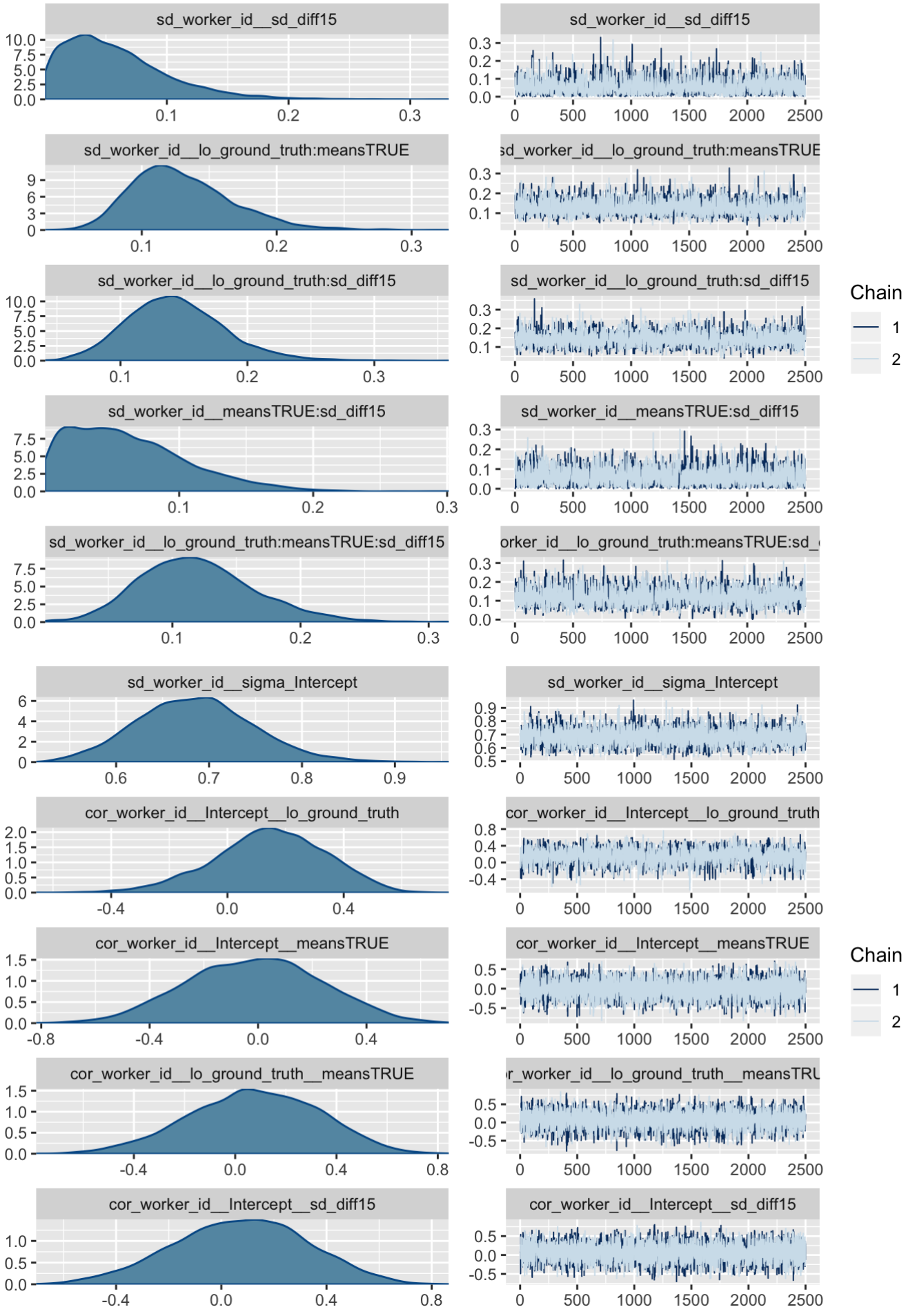
This first model adds random effects for the within-subjects manipulations in our previous model.

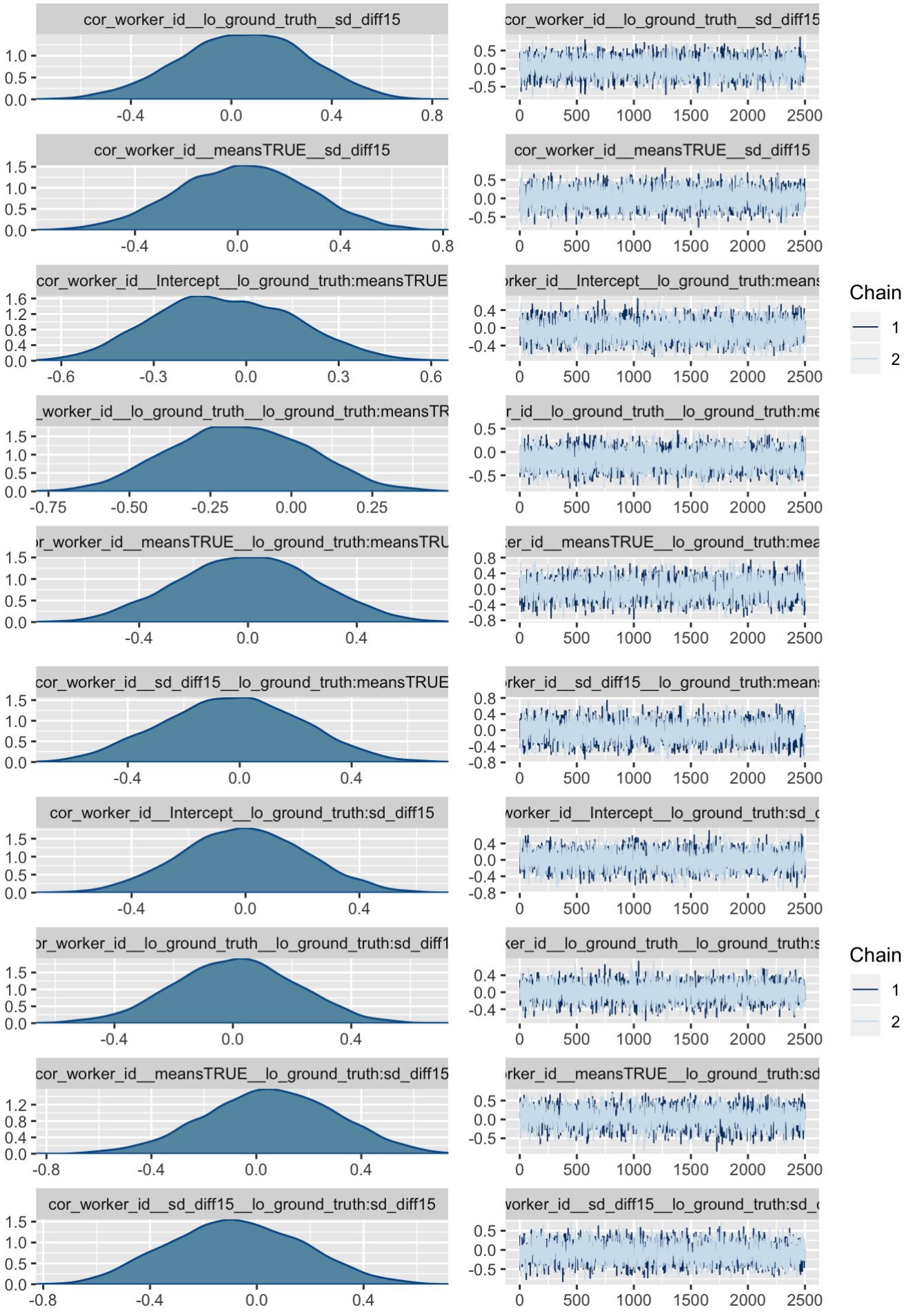
```
# hierarchical LLO model
m.wrkr.means.sd.vis.llo_p_sup.r.means.sd <- brm(data = model_df, family = "gaussian",
                                                formula = bf(lo_p_sup ~  (1 + lo_ground_
truth*means*sd_diff|sharecor|worker_id) + lo_ground_truth*means*sd_diff*condition,
                                                             sigma ~ (1|sharecor|worker_i
d)),
                                                prior = c(prior(normal(1, 0.5), class =
 b),
                                                          prior(normal(1.3, 1), class =
 Intercept),
                                                          prior(normal(0, 0.15), class =
sd, group = worker_id),
                                                          # prior(normal(0, 0.3), class
 = b, dpar = sigma),
                                                          prior(normal(0, 0.15), class =
sd, dpar = sigma),
                                                          prior(lkj(4), class = cor)),
                                                iter = 3000, warmup = 500, chains = 2, c
ores = 2,
                                                control = list(adapt_delta = 0.99, max_t
reedepth = 12),
                                                file = "model-fits/llo_mdl-wrkr_means_sd
_vis-r_means_sd")
```

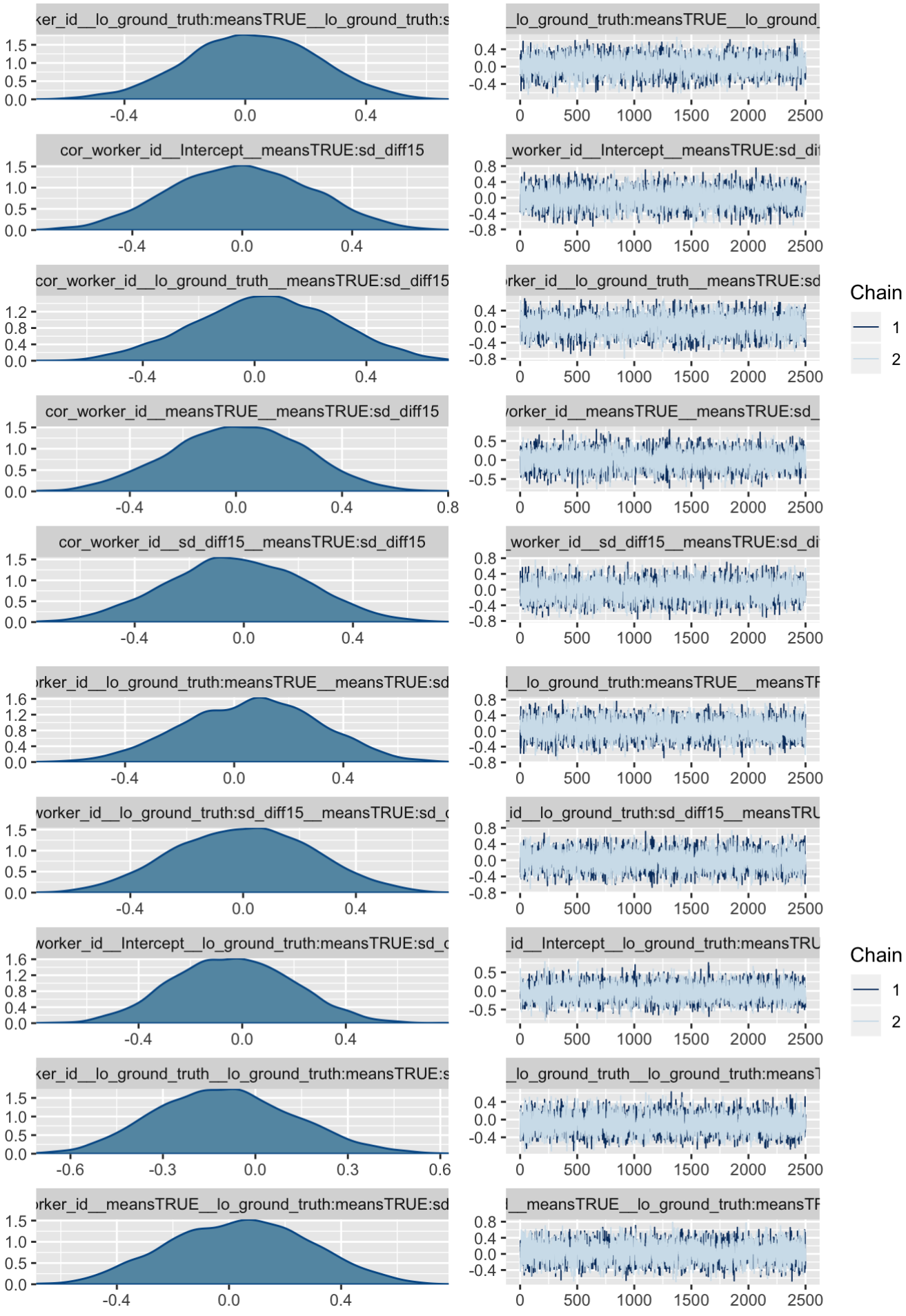
```
plot(m.wrkr.means.sd.vis.llo_p_sup.r.means.sd)
```

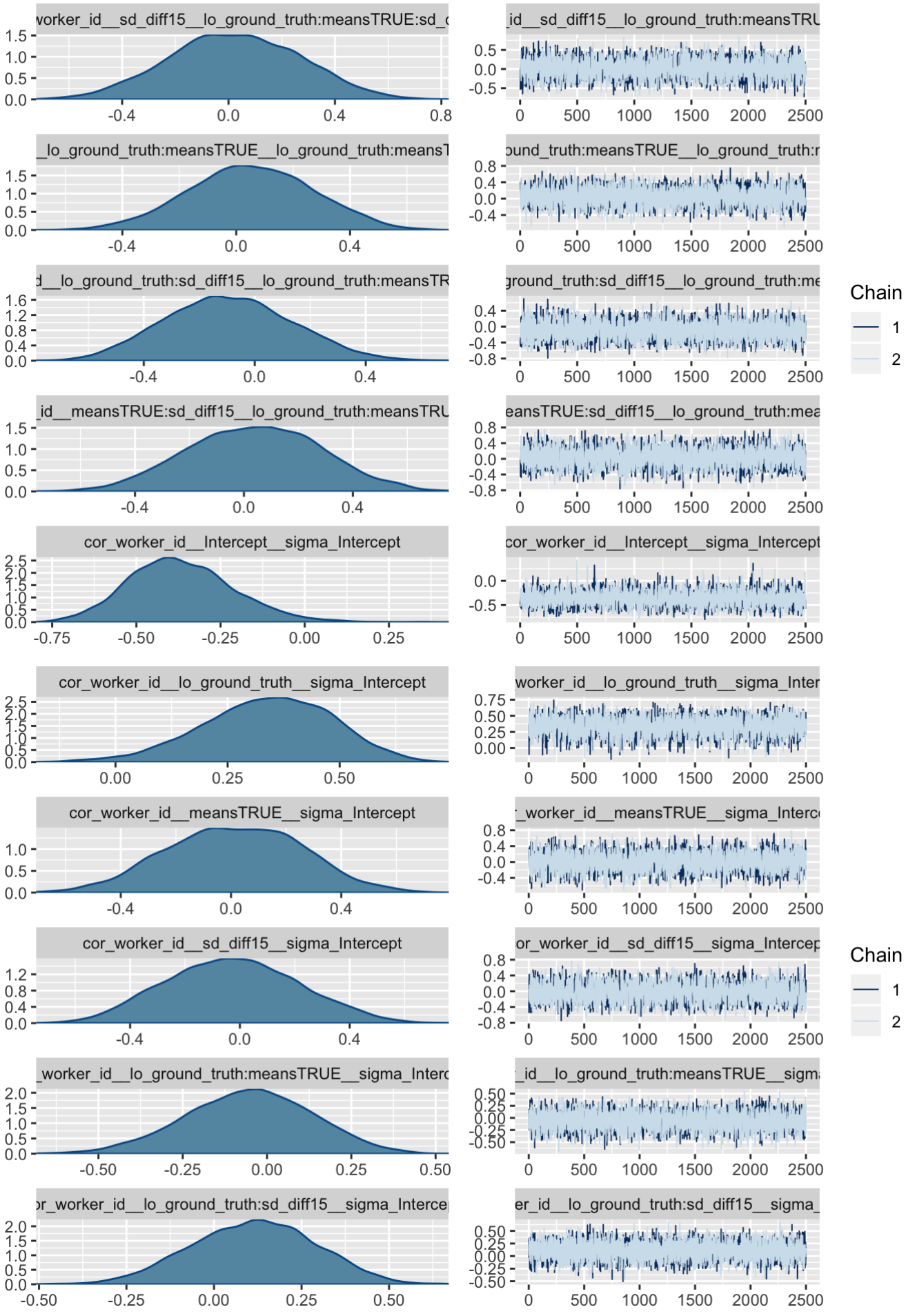

b_Intercept
b_sigma_Intercept
b_lo_ground_truth
b_meansTRUE
b_sd_diff15
b_conditionintervals
b_lo_ground_truth:meansTRUE
b_lo_ground_truth:sd_diff15
b_meansTRUE:sd_diff15
b_lo_ground_truth:conditionintervals
b_meansTRUE:conditionintervals
Chain
1
2

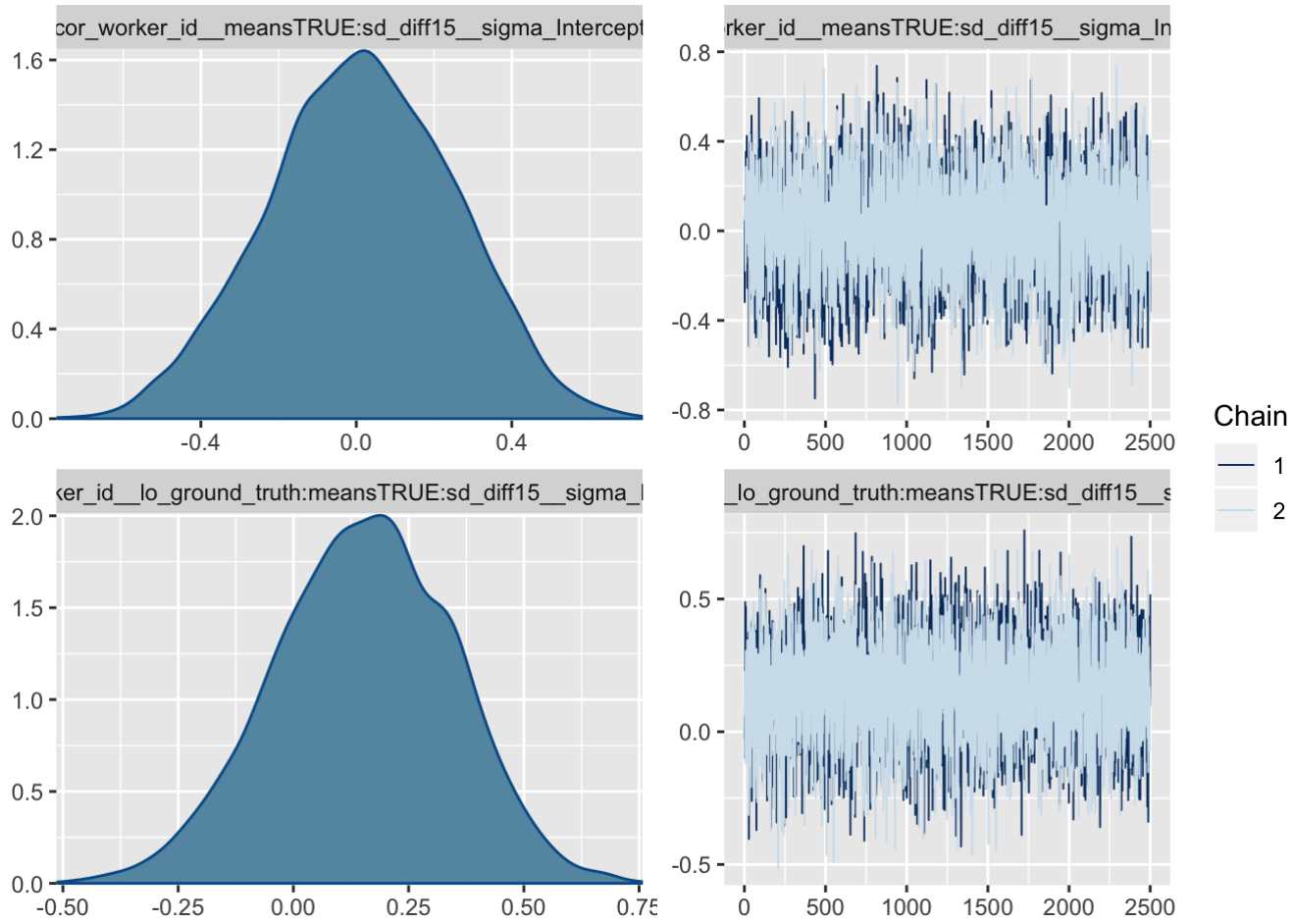

b_sd_diff15:conditionintervals
b_lo_ground_truth:meansTRUE:sd_diff15
b_lo_ground_truth:meansTRUE:conditionintervals
b_lo_ground_truth:sd_diff15:conditionintervals
b_meansTRUE:sd_diff15:conditionintervals
sd_worker_id__Intercept
sd_worker_id__lo_ground_truth
sd_worker_id__meansTRUE
Chain
1
2


sd_worker_id__sd_diff15
sd_worker_id__lo_ground_truth:meansTRUE
sd_worker_id__lo_ground_truth:sd_diff15
sd_worker_id__meansTRUE:sd_diff15
sd_worker_id__lo_ground_truth:meansTRUE:sd_diff15
sd_worker_id__sigma_Intercept
cor_worker_id__Intercept__lo_ground_truth
cor_worker_id__Intercept__meansTRUE
cor_worker_id__lo_ground_truth__meansTRUE
cor_worker_id__Intercept__sd_diff15
Chain
1
2


cor_worker_id__lo_ground_truth__sd_diff15
cor_worker_id__meansTRUE__sd_diff15
cor_worker_id__Intercept__lo_ground_truth:meansTRUE
_worker_id__lo_ground_truth__lo_ground_truth:meansTR
r_worker_id__meansTRUE__lo_ground_truth:meansTRU
cor_worker_id__sd_diff15__lo_ground_truth:meansTRUE
cor_worker_id__Intercept__lo_ground_truth:sd_diff15
r_worker_id__lo_ground_truth__lo_ground_truth:sd_diff1
cor_worker_id__meansTRUE__lo_ground_truth:sd_diff15
cor_worker_id__sd_diff15__lo_ground_truth:sd_diff15
Chain
1
2


cor_worker_id__Intercept__meansTRUE:sd_diff15
cor_worker_id__lo_ground_truth__meansTRUE:sd_diff15
cor_worker_id__meansTRUE__meansTRUE:sd_diff15
cor_worker_id__sd_diff15__meansTRUE:sd_diff15
Chain
1
2


worker_id__sd_diff15__lo_ground_truth:meansTRUE:sd_
id__sd_diff15__lo_ground_truth:meansTRU
_lo_ground_truth:meansTRUE__lo_ground_truth:meansT
ound_truth:meansTRUE__lo_ground_truth:r
d__lo_ground_truth:sd_diff15__lo_ground_truth:meansTR
ground_truth:sd_diff15__lo_ground_truth:me
_id__meansTRUE:sd_diff15__lo_ground_truth:meansTRU
eansTRUE:sd_diff15__lo_ground_truth:mea
cor_worker_id__Intercept__sigma_Intercept
cor_worker_id__Intercept__sigma_Intercept
cor_worker_id__lo_ground_truth__sigma_Intercept
worker_id__lo_ground_truth__sigma_Inter
cor_worker_id__meansTRUE__sigma_Intercept
_worker_id__meansTRUE__sigma_Interc
cor_worker_id__sd_diff15__sigma_Intercept
or_worker_id__sd_diff15__sigma_Intercep
_worker_id__lo_ground_truth:meansTRUE__sigma_Interc
_id__lo_ground_truth:meansTRUE__sigm
or_worker_id__lo_ground_truth:sd_diff15__sigma_Interce
er_id__lo_ground_truth:sd_diff15__sigma_
Chain
1
2
Chain
1
2

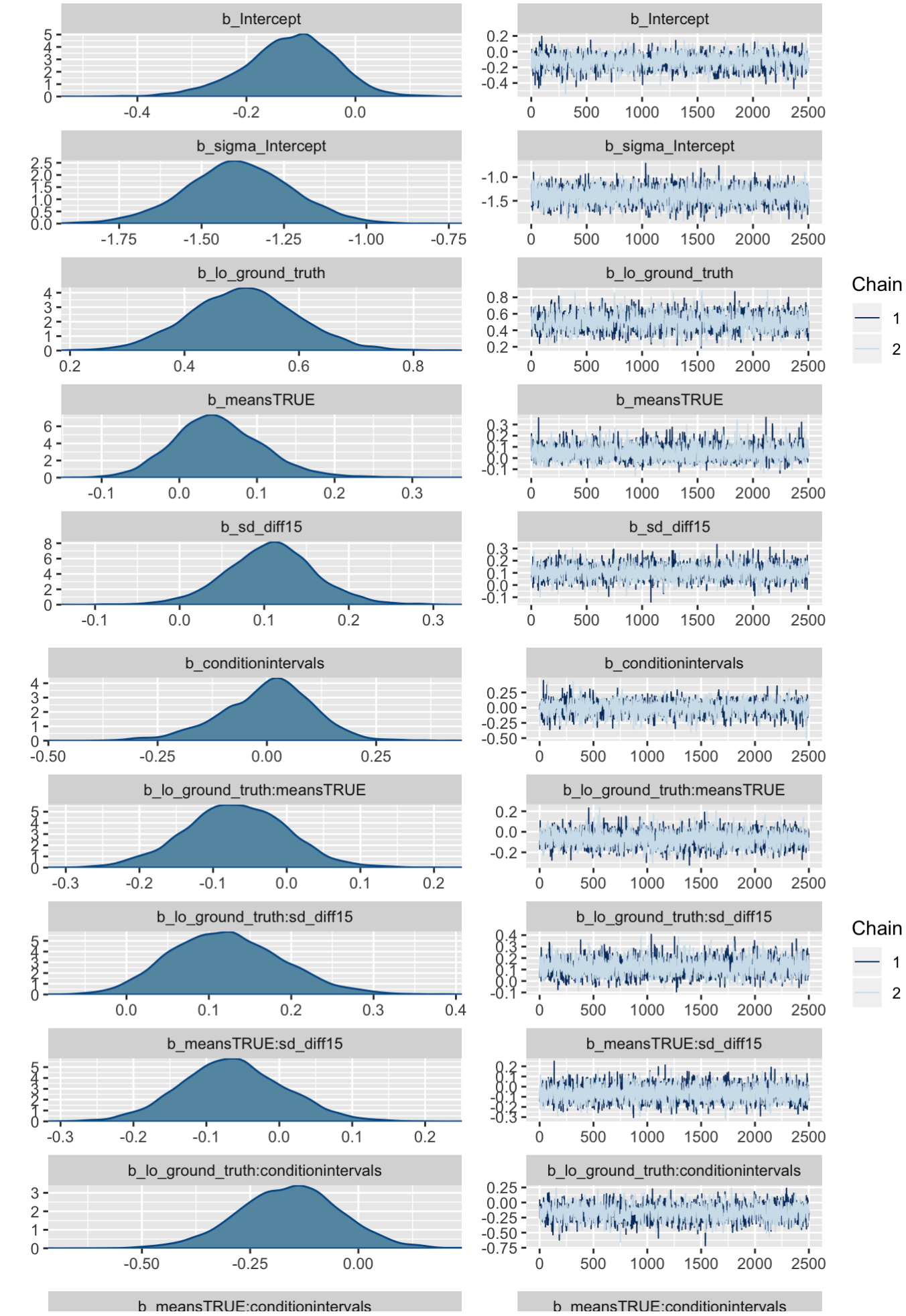


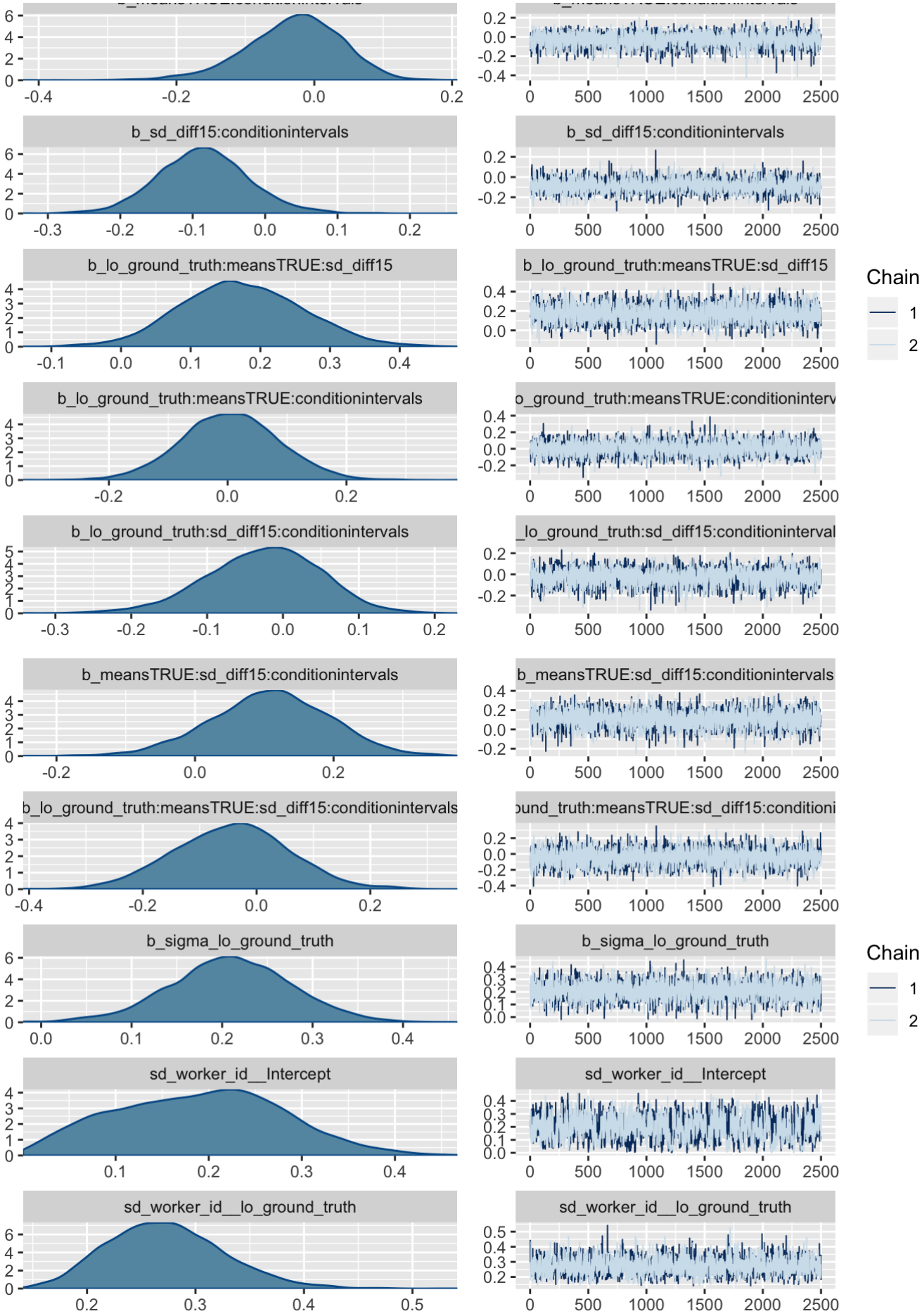
Mixed Effects of Ground Truth on Sigma

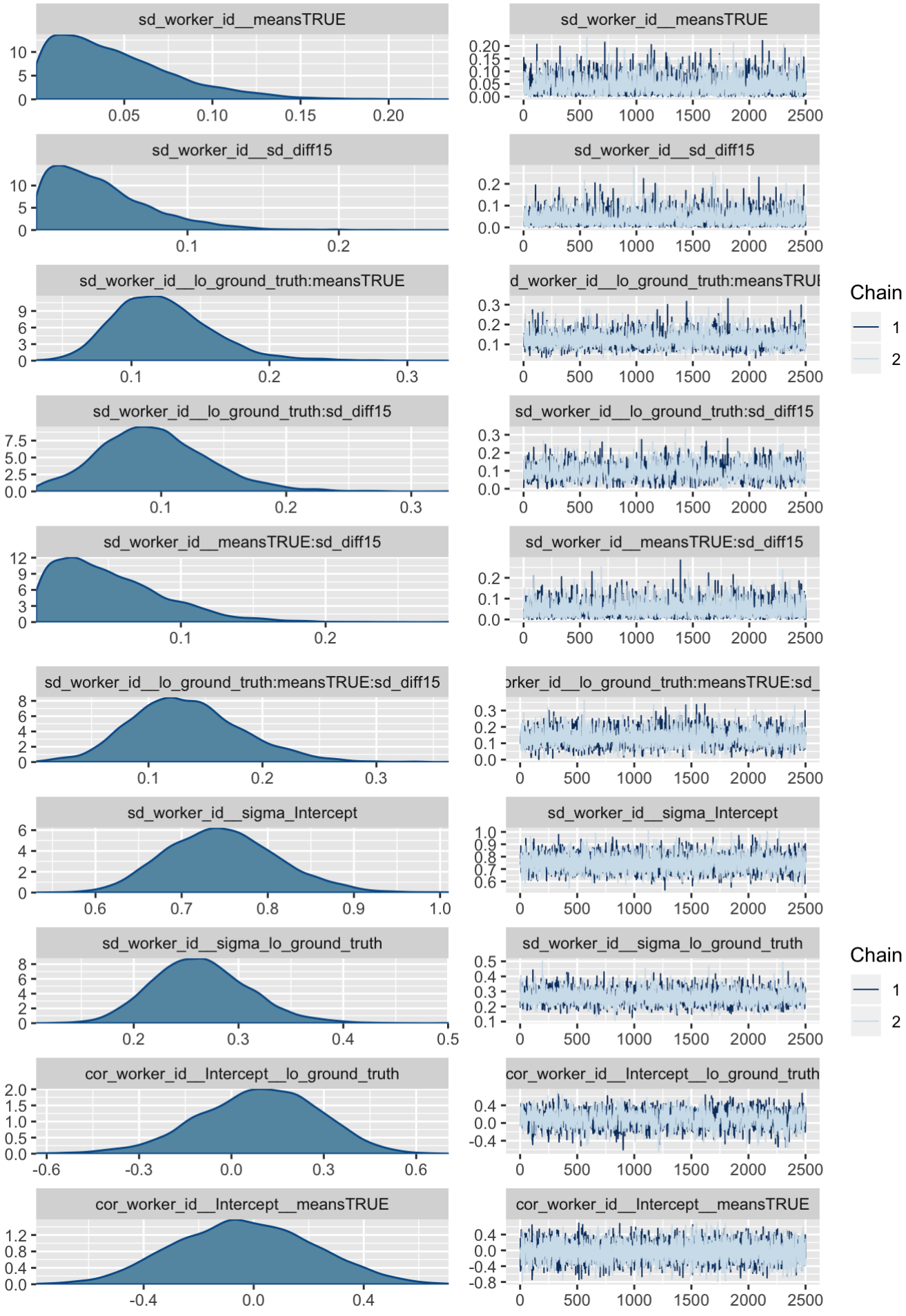
In this model, we add fixed and random effects of ground truth to our sigma submodel. This seems appropriate based on the empirical distribution of the individual responses vs ground truth. We add a conservative but informative prior in order to model fixed effects on sigma.

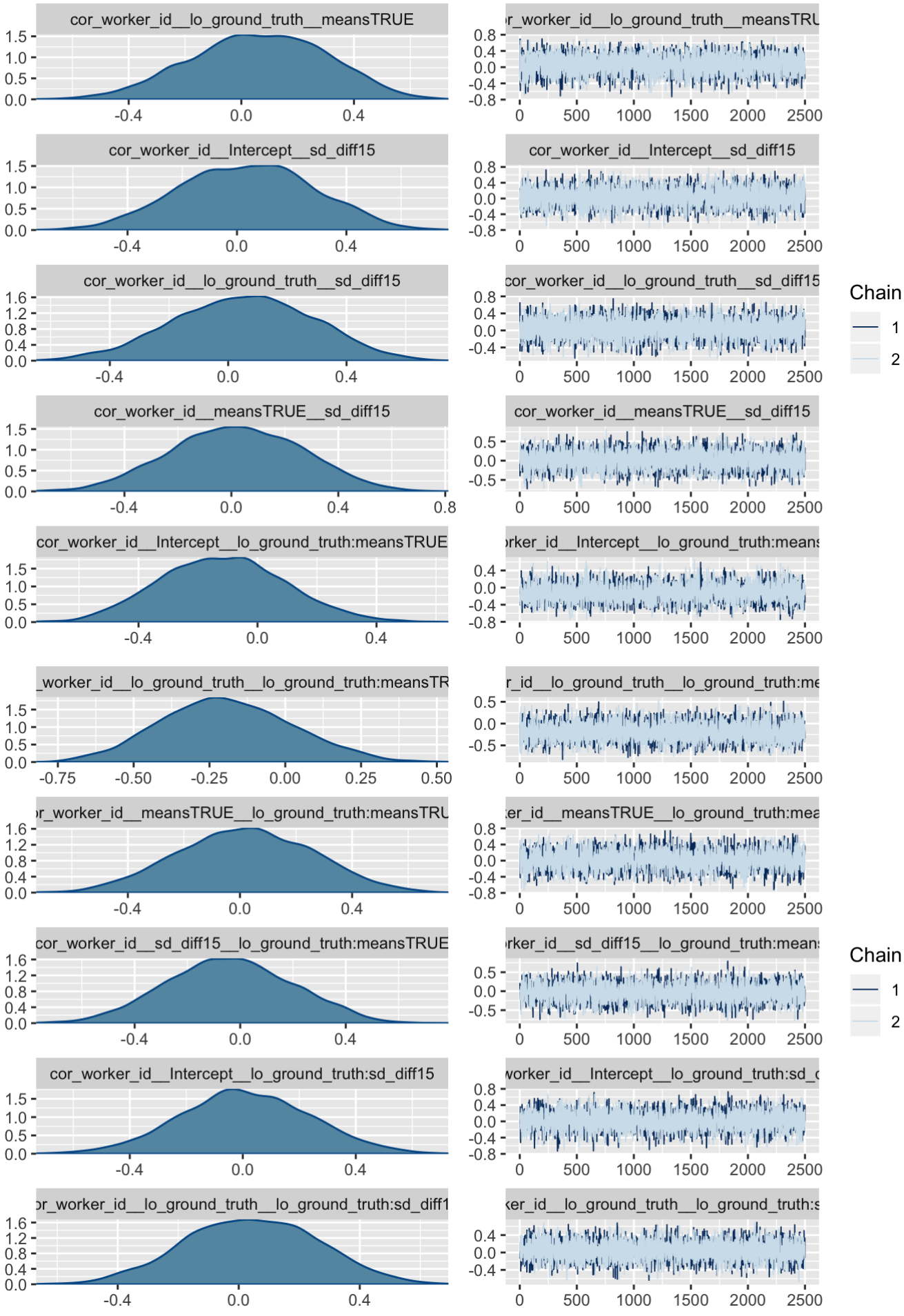
```
# hierarchical LLO model
m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.sigma.gt <- brm(data = model_df, family = "gaus
sian",
                                                         formula = bf(lo_p_sup ~  (1 + l
o_ground_truth*means*sd_diff|sharecor|worker_id) + lo_ground_truth*means*sd_diff*conditi
on,
                                                                      sigma ~ (1 + lo_gr
ound_truth|sharecor|worker_id) + lo_ground_truth),
                                                         prior = c(prior(normal(1, 0.5),
class = b),
                                                                   prior(normal(1.3, 1),
class = Intercept),
                                                                   prior(normal(0, 0.15
), class = sd, group = worker_id),
                                                                   prior(normal(0, 0.3),
class = b, dpar = sigma),
                                                                   prior(normal(0, 0.15
), class = sd, dpar = sigma),
                                                                   prior(lkj(4), class =
cor)),
                                                         iter = 3000, warmup = 500, chai
ns = 2, cores = 2,
                                                         control = list(adapt_delta = 0.
99, max_treedepth = 12),
                                                         file = "model-fits/llo_mdl-wrkr
_means_sd_vis-r_means_sd_sigma_gt")
```

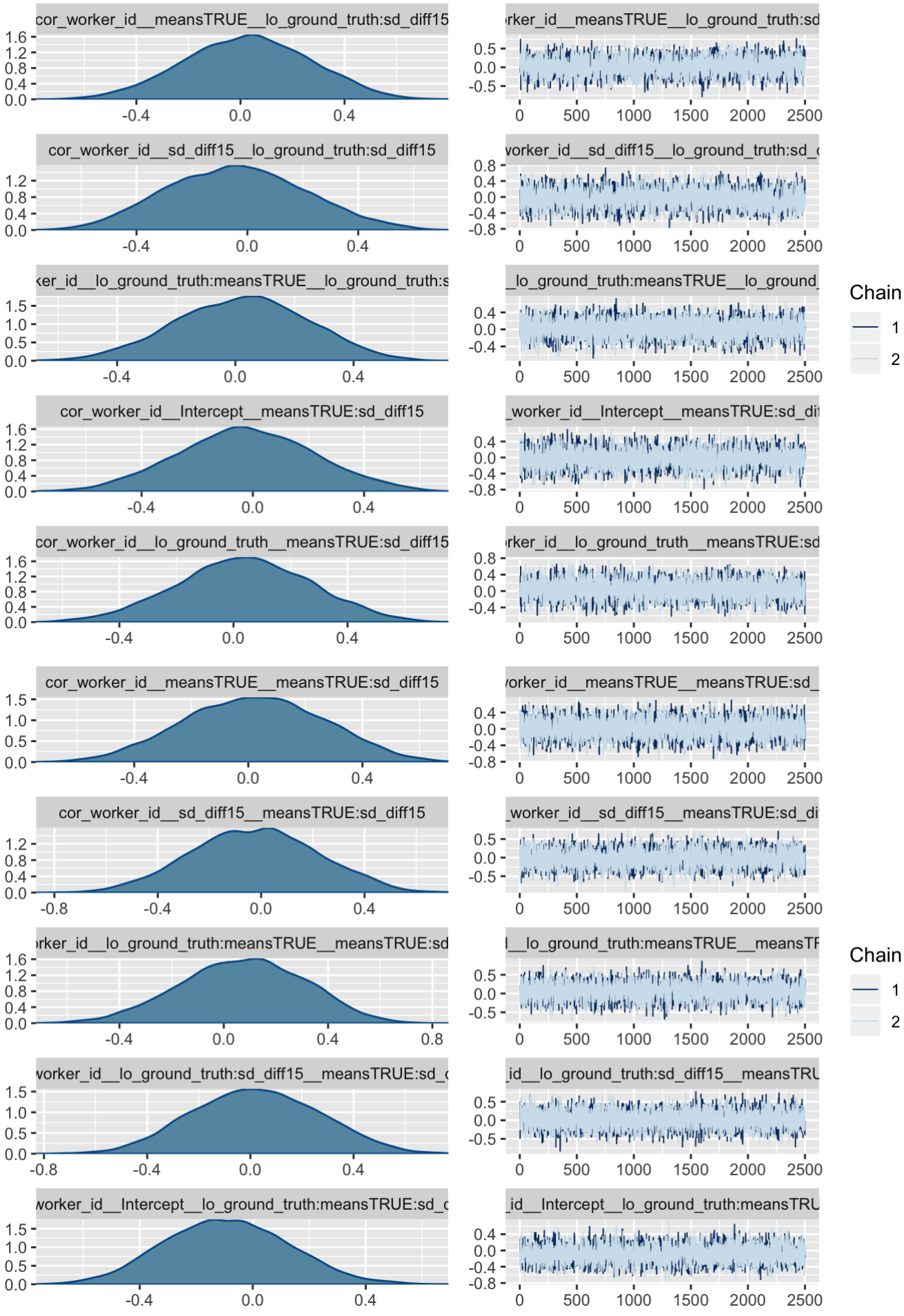
```
plot(m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.sigma.gt)
```

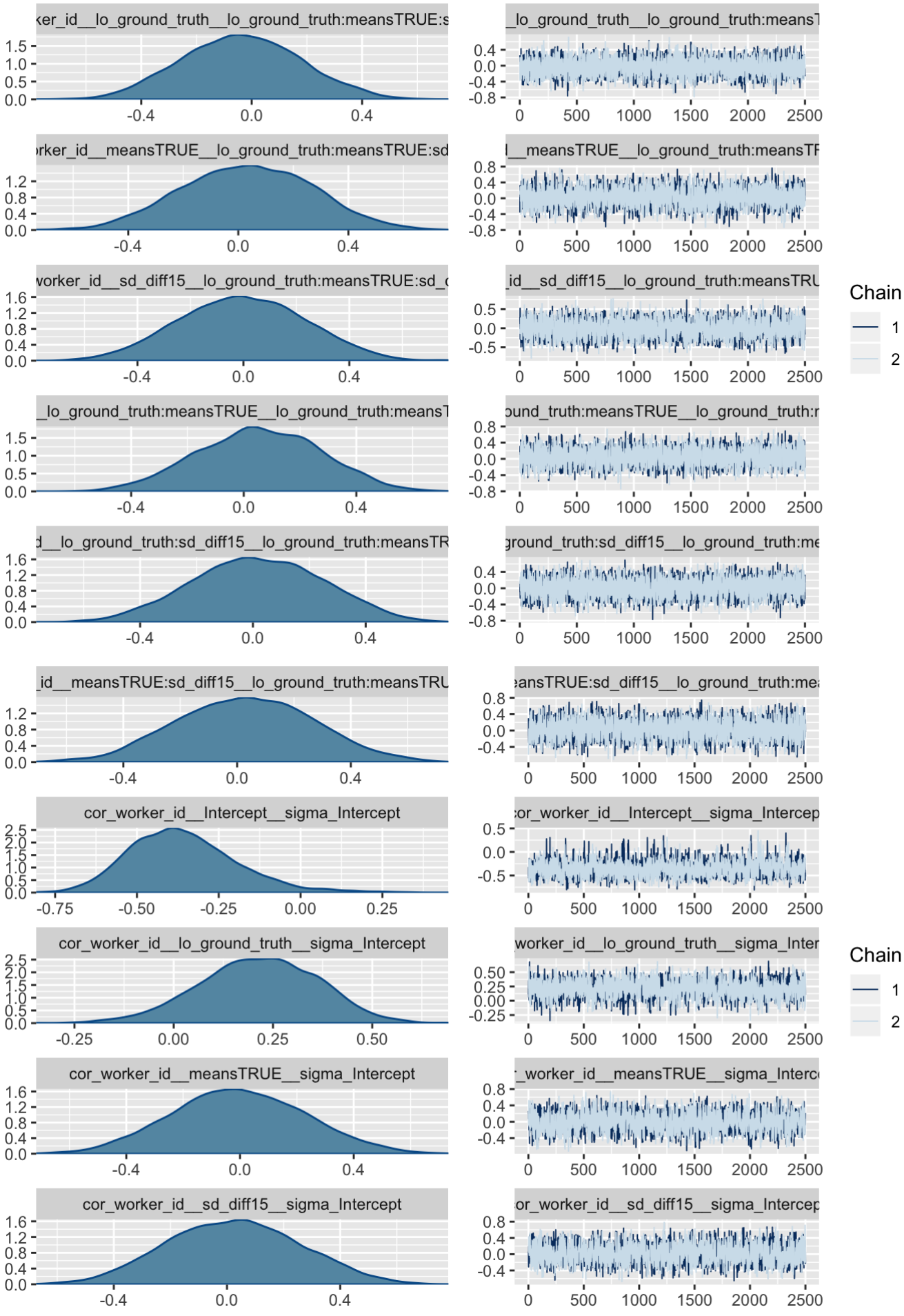

b_Intercept
b_sigma_Intercept
b_lo_ground_truth
b_meansTRUE
b_sd_diff15
b_conditionintervals
b_lo_ground_truth:meansTRUE
b_lo_ground_truth:sd_diff15
b_meansTRUE:sd_diff15
b_lo_ground_truth:conditionintervals
b_meansTRUE:conditionintervals
Chain
1
2

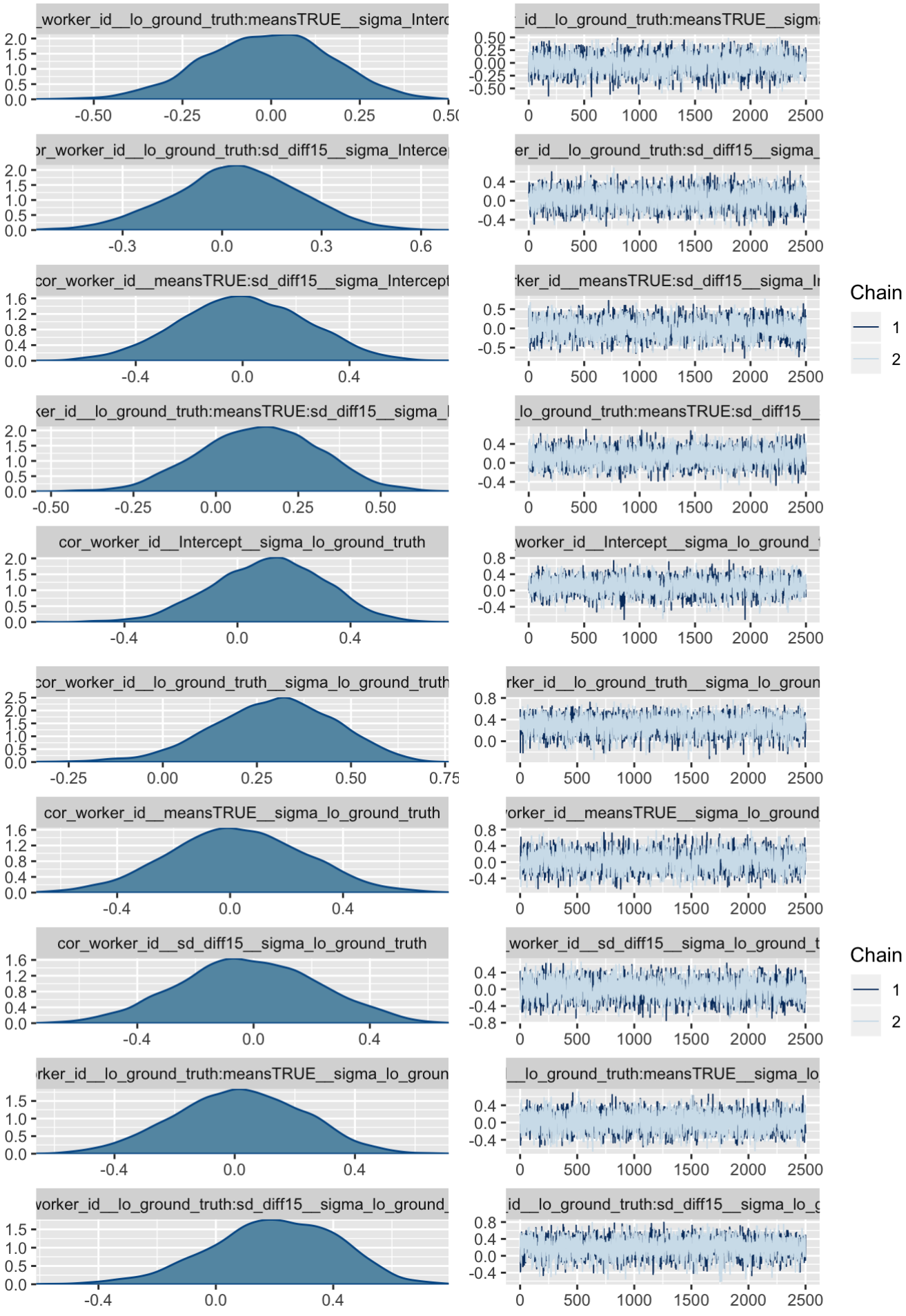

b_sd_diff15:conditionintervals
b_lo_ground_truth:meansTRUE:sd_diff15
b_lo_ground_truth:meansTRUE:conditionintervals
b_lo_ground_truth:sd_diff15:conditionintervals
b_meansTRUE:sd_diff15:conditionintervals
b_lo_ground_truth:meansTRUE:sd_diff15:conditionintervals
b_sigma_lo_ground_truth
sd_worker_id__Intercept
sd_worker_id__lo_ground_truth
Chain
1
2

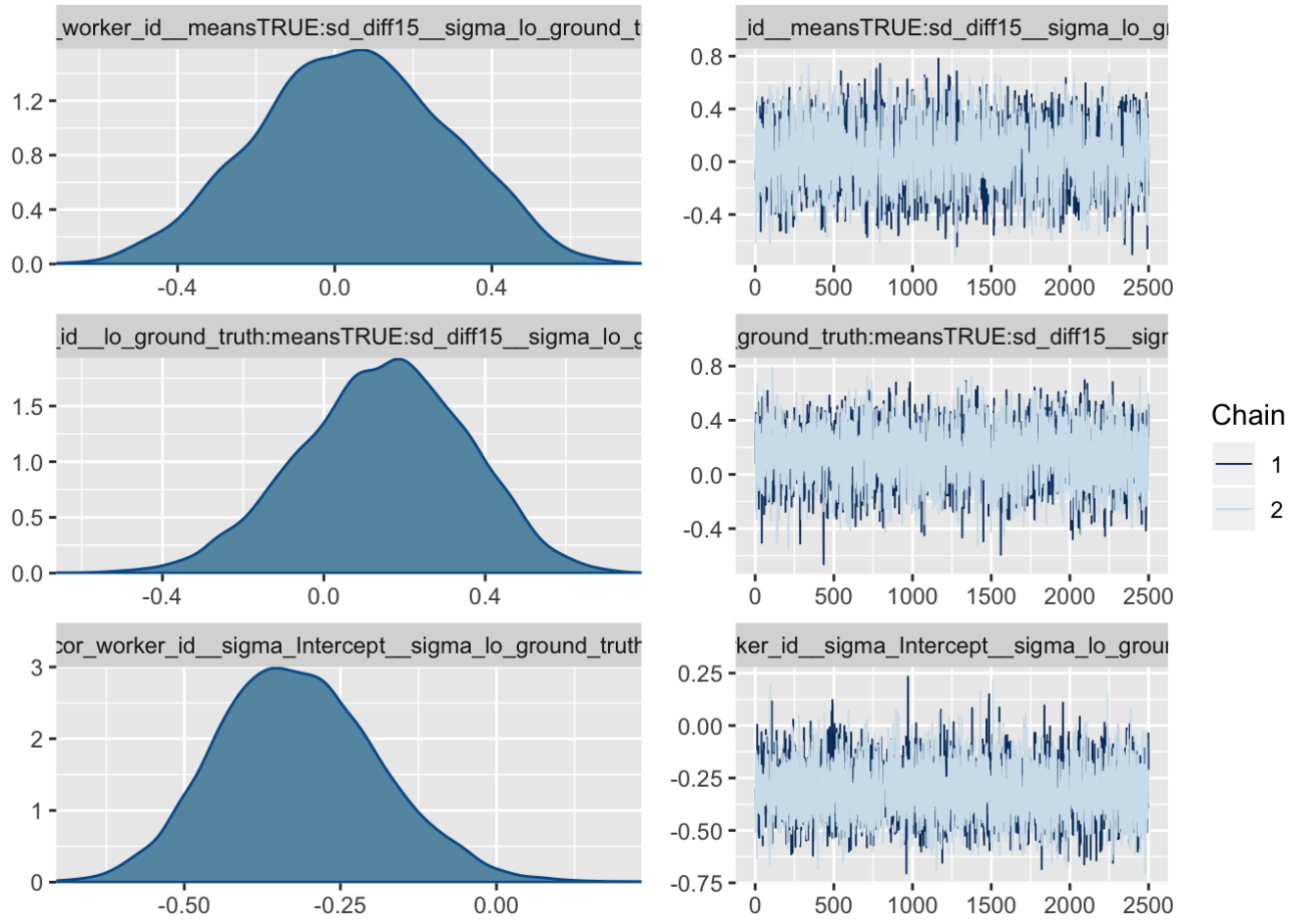

sd_worker_id__meansTRUE
sd_worker_id__sd_diff15
sd_worker_id__lo_ground_truth:meansTRUE
sd_worker_id__lo_ground_truth:sd_diff15
sd_worker_id__meansTRUE:sd_diff15
sd_worker_id__lo_ground_truth:meansTRUE:sd_diff15
sd_worker_id__sigma_Intercept
sd_worker_id__sigma_lo_ground_truth
cor_worker_id__Intercept__lo_ground_truth
cor_worker_id__Intercept__meansTRUE
Chain
1
2


cor_worker_id__lo_ground_truth__meansTRUE
cor_worker_id__Intercept__sd_diff15
cor_worker_id__lo_ground_truth__sd_diff15
cor_worker_id__meansTRUE__sd_diff15
cor_worker_id__Intercept__lo_ground_truth:meansTRUE
_worker_id__lo_ground_truth__lo_ground_truth:meansTR
r_worker_id__meansTRUE__lo_ground_truth:meansTRU
cor_worker_id__sd_diff15__lo_ground_truth:meansTRUE
cor_worker_id__Intercept__lo_ground_truth:sd_diff15
r_worker_id__lo_ground_truth__lo_ground_truth:sd_diff1
Chain
1
2


cor_worker_id__meansTRUE__lo_ground_truth:sd_diff15
cor_worker_id__sd_diff15__lo_ground_truth:sd_diff15
cor_worker_id__Intercept__meansTRUE:sd_diff15
cor_worker_id__lo_ground_truth__meansTRUE:sd_diff15
cor_worker_id__meansTRUE__meansTRUE:sd_diff15
cor_worker_id__sd_diff15__meansTRUE:sd_diff15
cor_worker_id__lo_ground_truth:sd_diff15__meansTRUE:sd_diff15
cor_worker_id__Intercept__lo_ground_truth:meansTRUE:sd_diff15
Chain
1
2


cor_worker_id__Intercept__sigma_Intercept
cor_worker_id__lo_ground_truth__sigma_Intercept
cor_worker_id__meansTRUE__sigma_Intercept
cor_worker_id__sd_diff15__sigma_Intercept
Chain
1
2


_worker_id__lo_ground_truth:meansTRUE__sigma_Interc
_id__lo_ground_truth:meansTRUE__sigm
or_worker_id__lo_ground_truth:sd_diff15__sigma_Interce
er_id__lo_ground_truth:sd_diff15__sigma_
cor_worker_id__meansTRUE:sd_diff15__sigma_Intercept
ker_id__meansTRUE:sd_diff15__sigma_I
ker_id__lo_ground_truth:meansTRUE:sd_diff15__sigma_
_lo_ground_truth:meansTRUE:sd_diff15__
cor_worker_id__Intercept__sigma_lo_ground_truth
worker_id__Intercept__sigma_lo_ground_
cor_worker_id__lo_ground_truth__sigma_lo_ground_truth
rker_id__lo_ground_truth__sigma_lo_groun
cor_worker_id__meansTRUE__sigma_lo_ground_truth
orker_id__meansTRUE__sigma_lo_ground
cor_worker_id__sd_diff15__sigma_lo_ground_truth
_worker_id__sd_diff15__sigma_lo_ground_t
rker_id__lo_ground_truth:meansTRUE__sigma_groun
__lo_ground_truth:meansTRUE__sigma_lo
worker_id__lo_ground_truth:sd_diff15__sigma_lo_ground_
id__lo_ground_truth:sd_diff15__sigma_lo_g
Chain
1
2
Chain
1
2

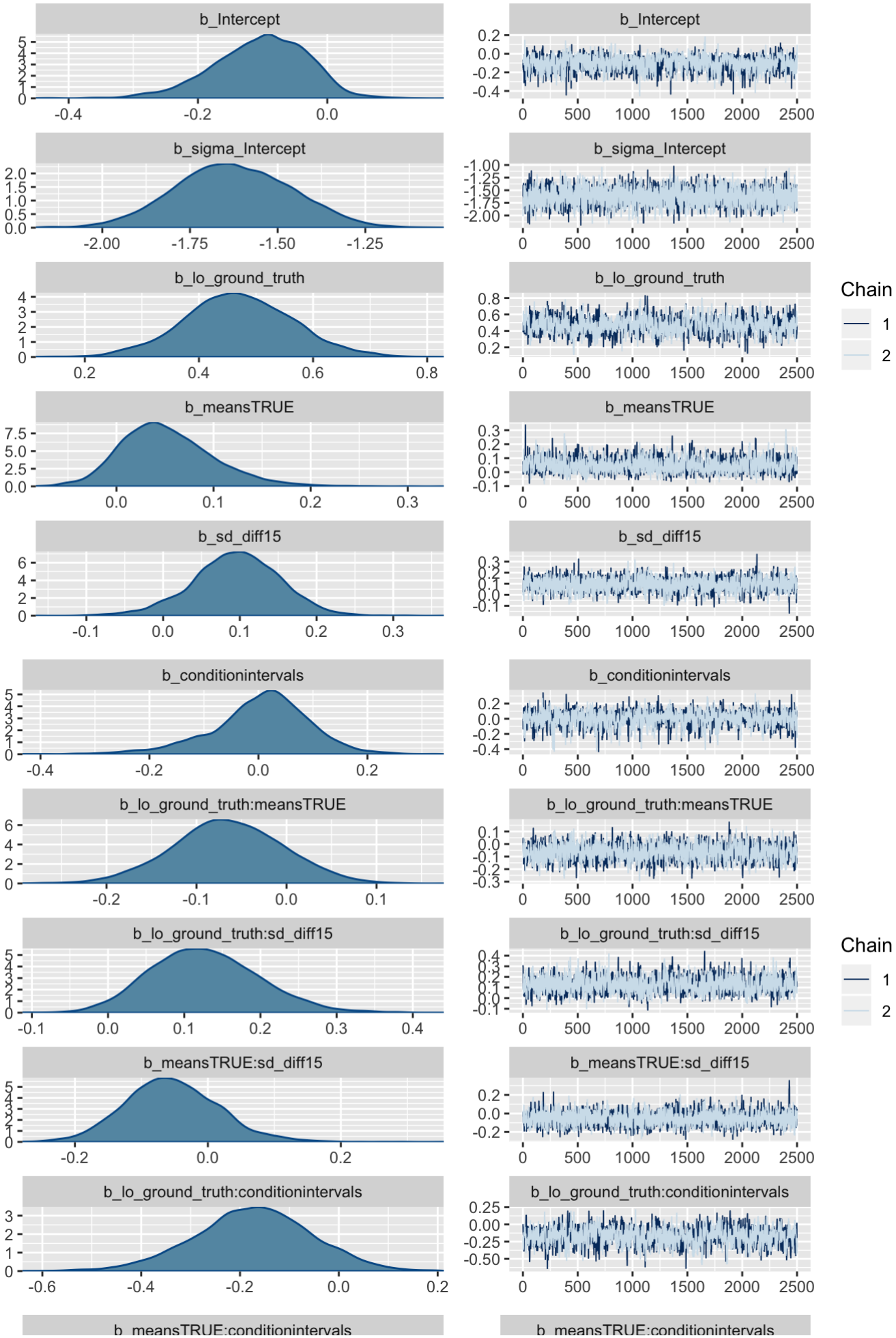


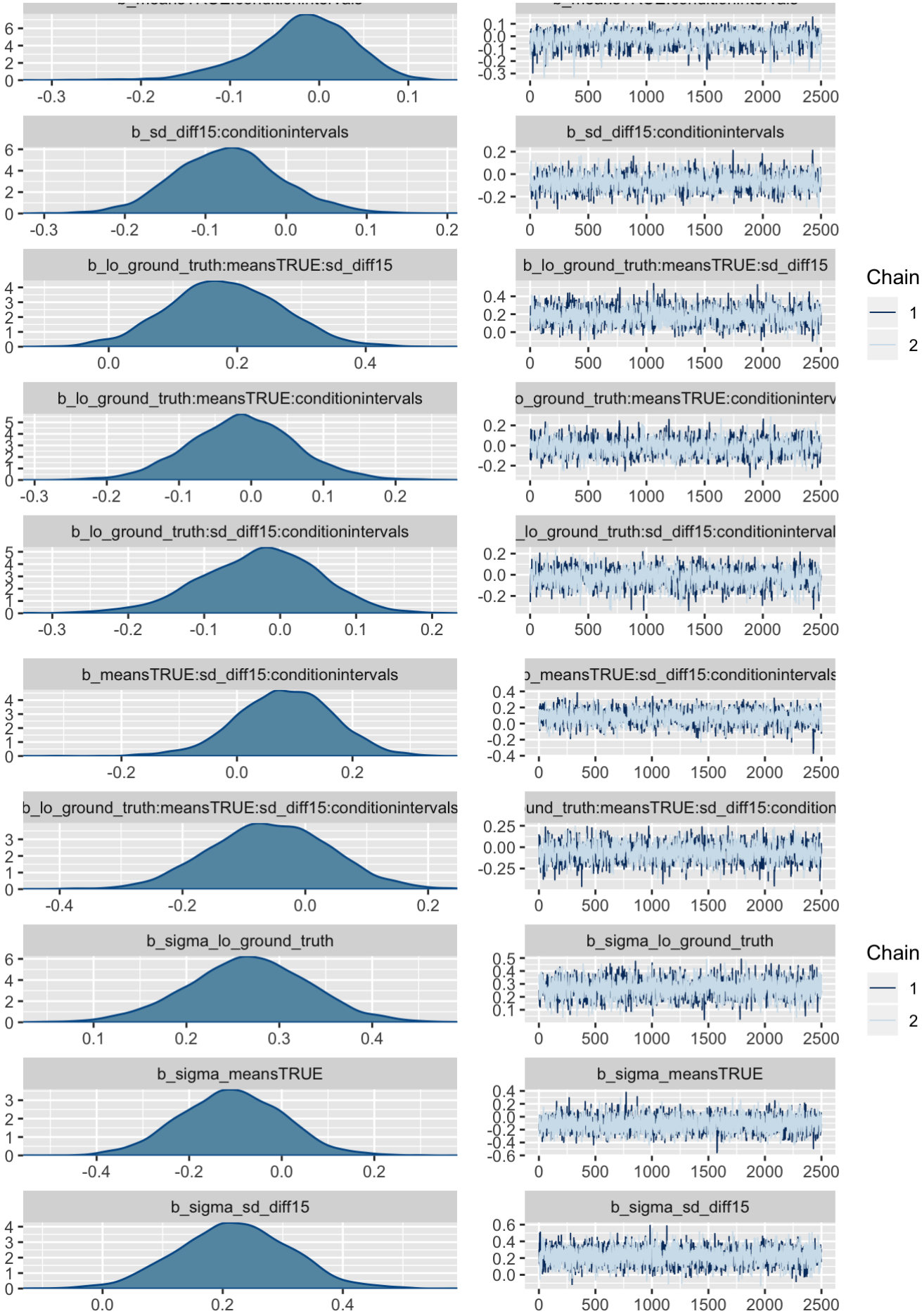
Mixed Effects of the Interaction of Means and Uncertainty Shown on Sigma

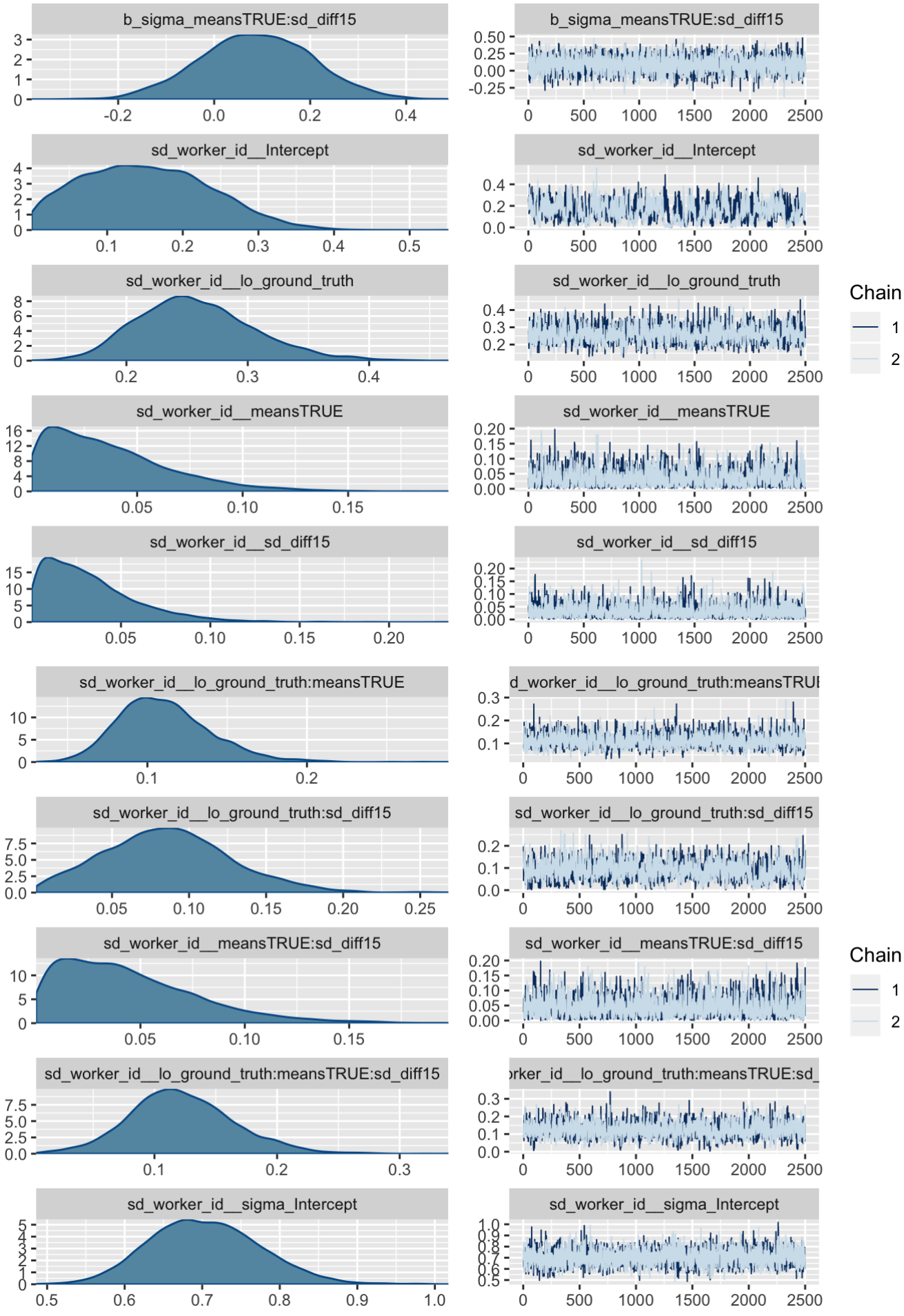
This model adds mixed effects on residual variance (sigma) for the interaction of extrinsic means and uncertainty shown. Note that we do not model this as a three way interaction with the ground truth because of fit issues.

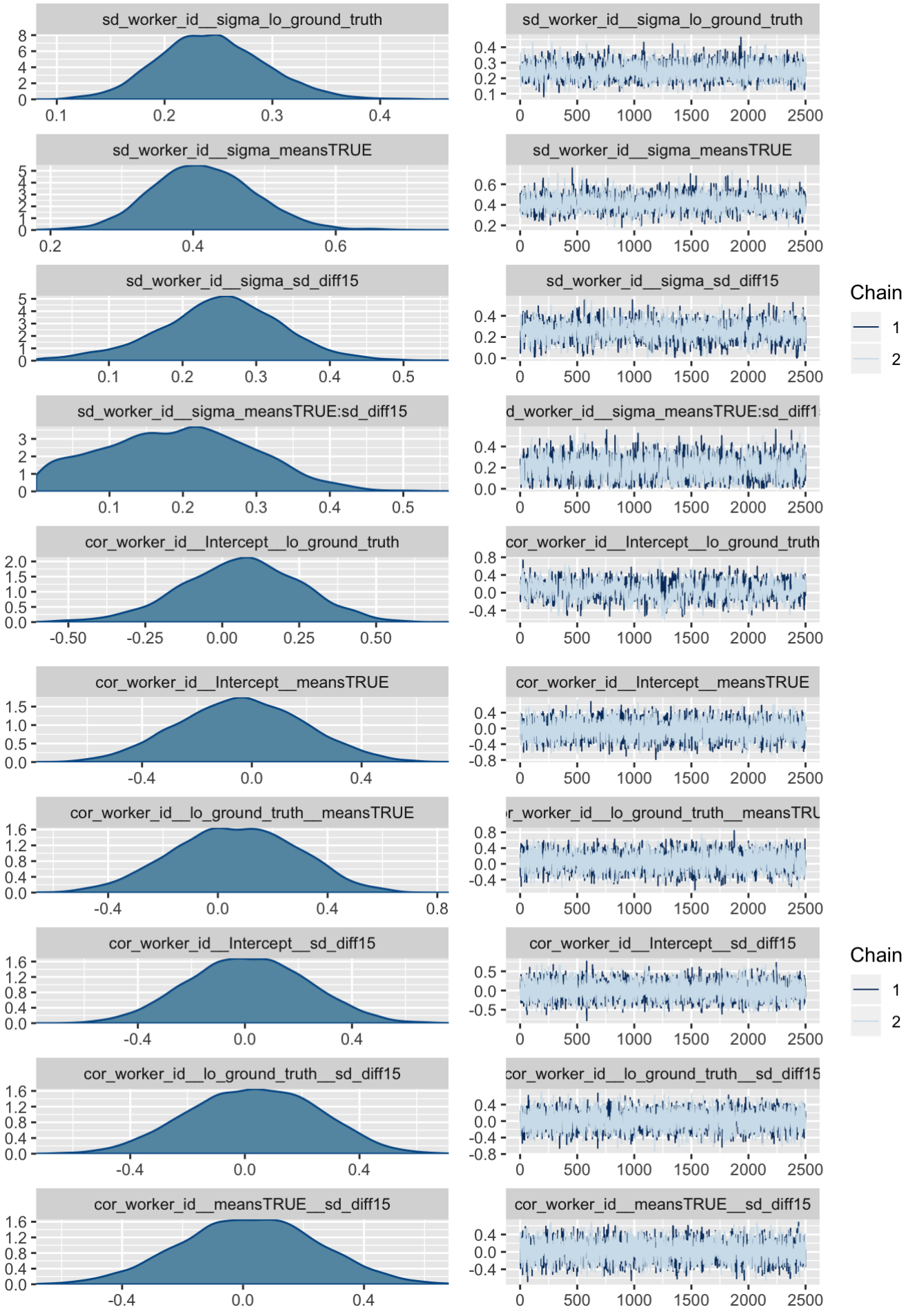
```
# hierarchical LLO model
m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.sigma.gt.means.sd <- brm(
  data = model_df, family = "gaussian",
  formula = bf(lo_p_sup ~  (1 + lo_ground_truth*means*sd_diff|sharecor|worker_id) + lo_g
round_truth*means*sd_diff*condition,
  sigma ~ (1 + lo_ground_truth + means*sd_diff|sharecor|worker_id) + lo_ground_truth + m
eans*sd_diff),
  prior = c(prior(normal(1, 0.5), class = b),
            prior(normal(1.3, 1), class = Intercept),
            prior(normal(0, 0.15), class = sd, group = worker_id),
            prior(normal(0, 0.3), class = b, dpar = sigma),
            prior(normal(0, 0.15), class = sd, dpar = sigma),
            prior(lkj(4), class = cor)),
  iter = 3000, warmup = 500, chains = 2, cores = 2,
  control = list(adapt_delta = 0.99, max_treedepth = 12),
  file = "model-fits/llo_mdl-wrkr_means_sd_vis-r_means_sd_sigma_gt_means_sd")
```

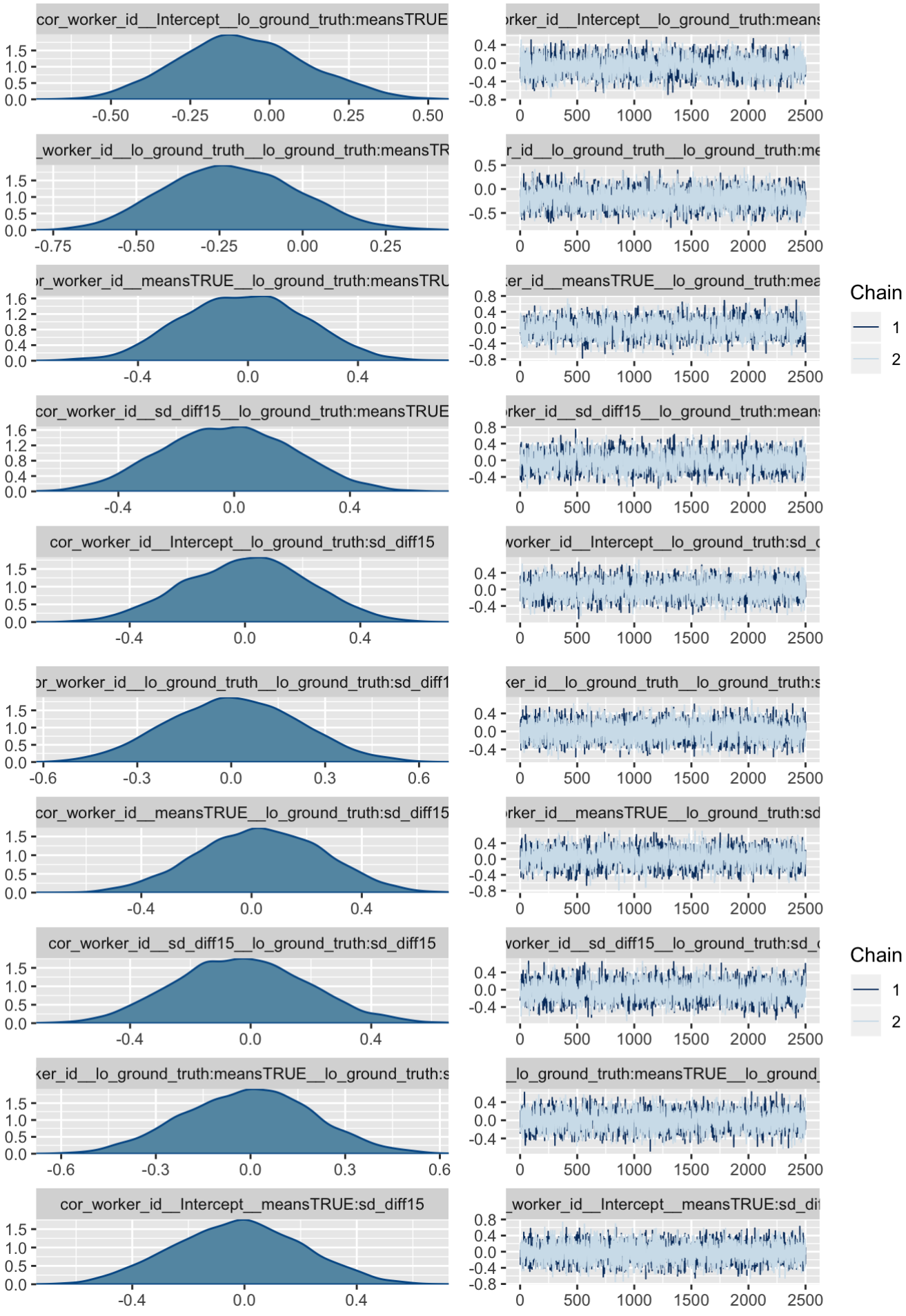
```
plot(m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.sigma.gt.means.sd)
```

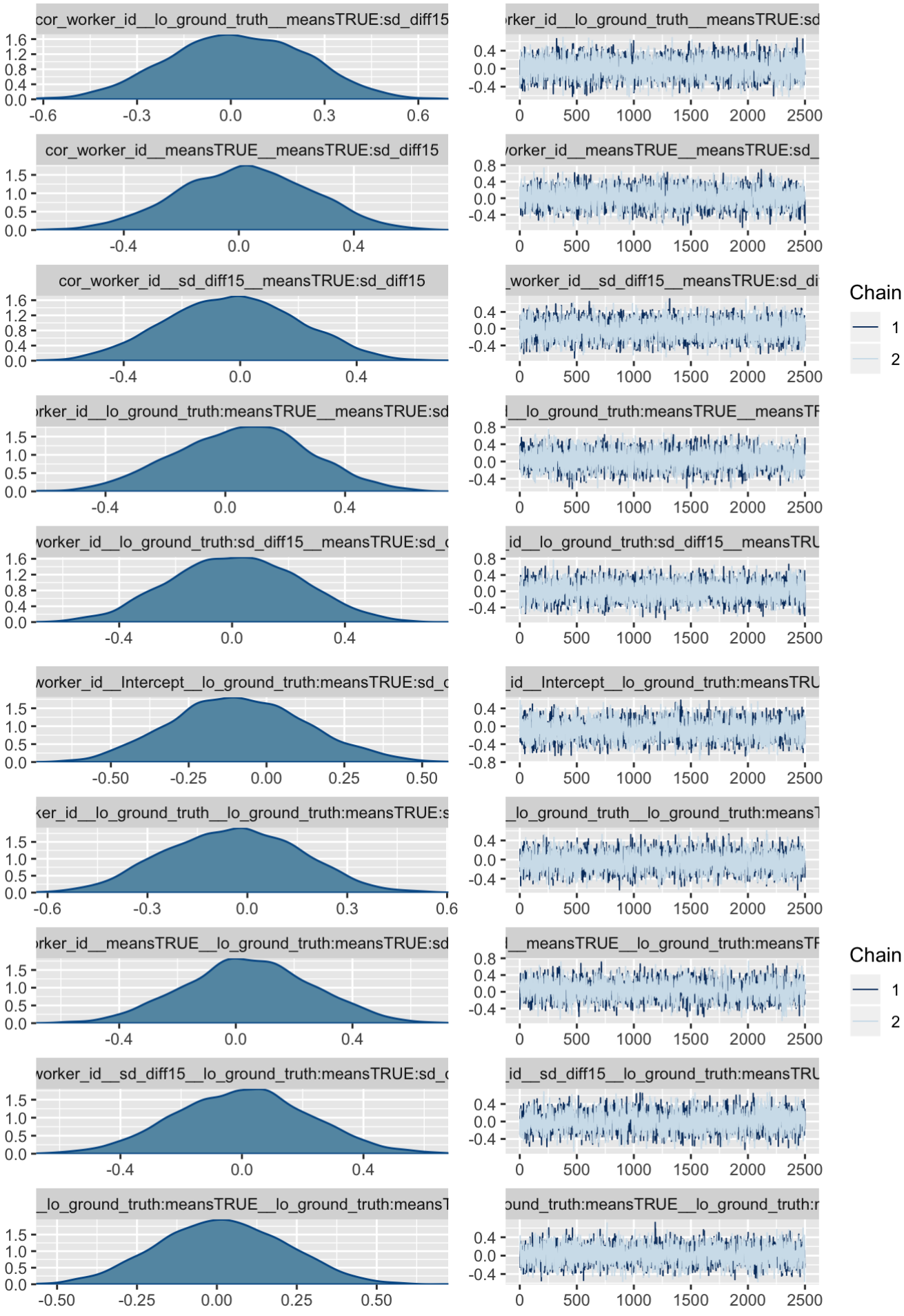

b_Intercept
b_sigma_Intercept
b_lo_ground_truth
b_meansTRUE
b_sd_diff15
b_conditionintervals
b_lo_ground_truth:meansTRUE
b_lo_ground_truth:sd_diff15
b_meansTRUE:sd_diff15
b_lo_ground_truth:conditionintervals
b_meansTRUE:conditionintervals
Chain
1
2

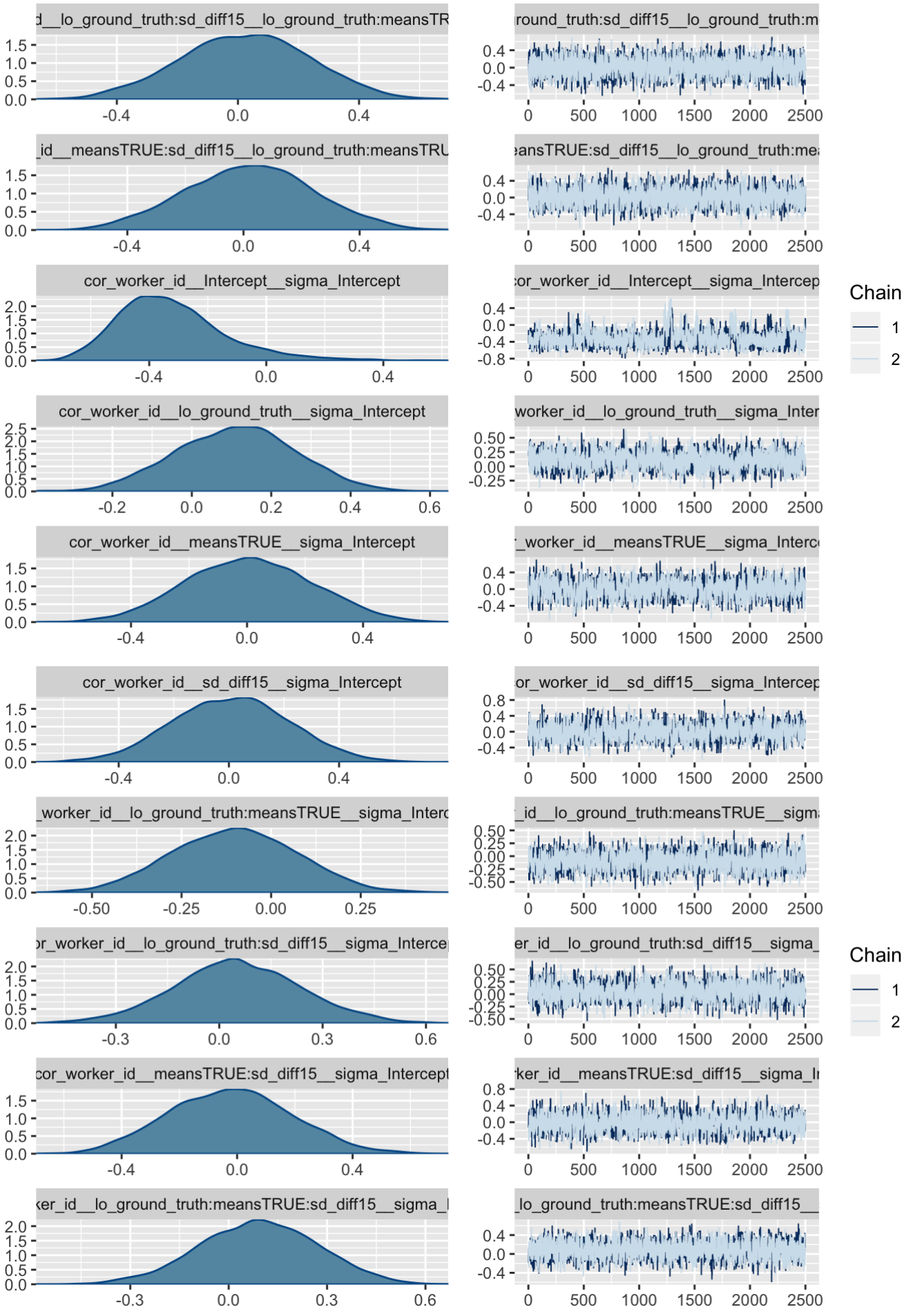

b_sd_diff15:conditionintervals
b_lo_ground_truth:meansTRUE:sd_diff15
b_lo_ground_truth:meansTRUE:conditionintervals
b_lo_ground_truth:sd_diff15:conditionintervals
b_meansTRUE:sd_diff15:conditionintervals
b_lo_ground_truth:meansTRUE:sd_diff15:conditionintervals
b_sigma_lo_ground_truth
b_sigma_meansTRUE
b_sigma_sd_diff15
Chain
1
2

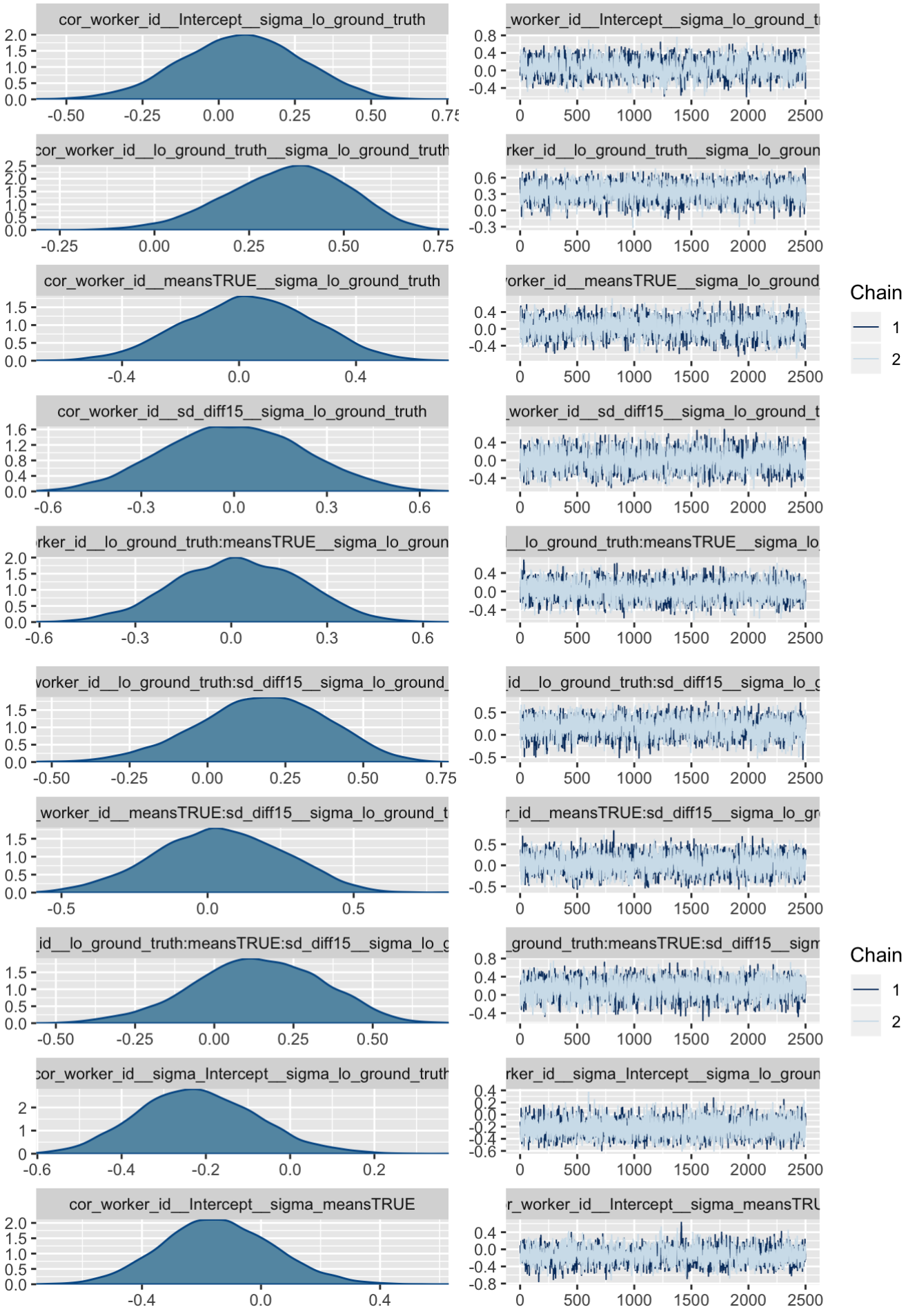

b_sigma_meansTRUE:sd_diff15
sd_worker_id__Intercept
sd_worker_id__lo_ground_truth
sd_worker_id__meansTRUE
sd_worker_id__sd_diff15
sd_worker_id__lo_ground_truth:meansTRUE
sd_worker_id__lo_ground_truth:sd_diff15
sd_worker_id__meansTRUE:sd_diff15
sd_worker_id__lo_ground_truth:meansTRUE:sd_diff15
sd_worker_id__sigma_Intercept
Chain
1
2

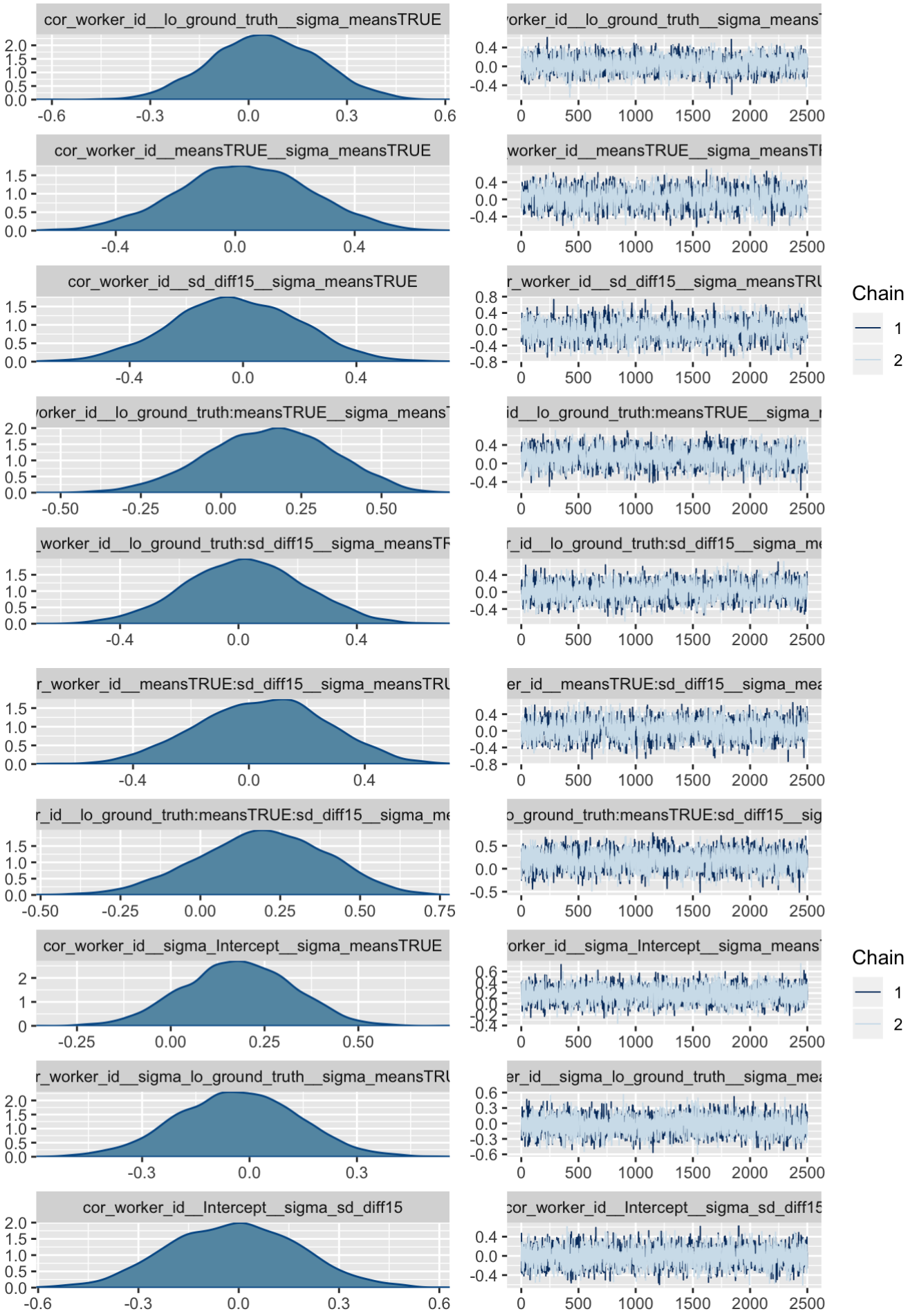

sd_worker_id__sigma_lo_ground_truth
sd_worker_id__sigma_meansTRUE
sd_worker_id__sigma_sd_diff15
sd_worker_id__sigma_meansTRUE:sd_diff15
cor_worker_id__Intercept__lo_ground_truth
cor_worker_id__Intercept__meansTRUE
cor_worker_id__lo_ground_truth__meansTRUE
cor_worker_id__Intercept__sd_diff15
cor_worker_id__lo_ground_truth__sd_diff15
cor_worker_id__meansTRUE__sd_diff15
Chain
1
2

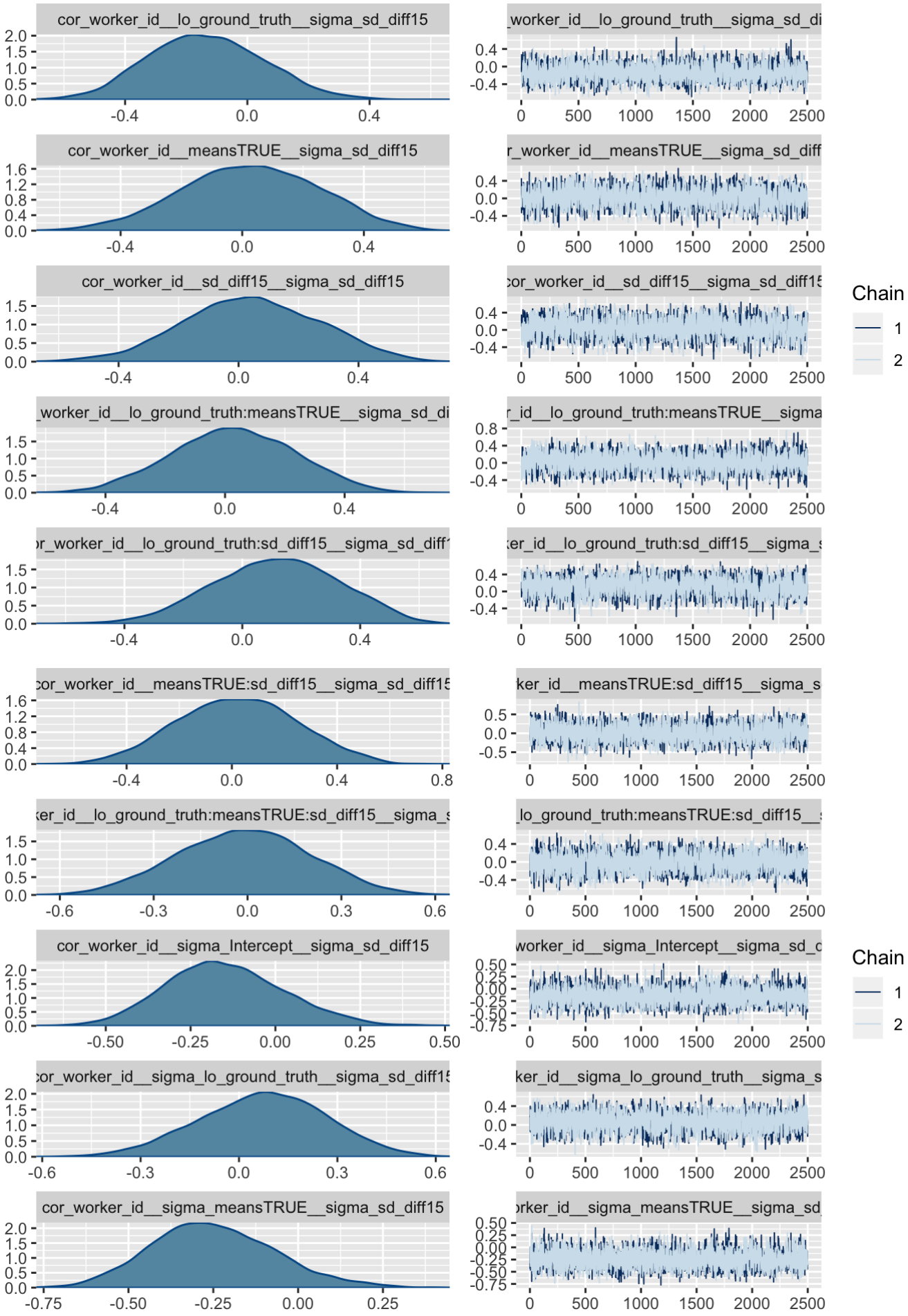

cor_worker_id__Intercept__lo_ground_truth:meansTRUE
_worker_id__lo_ground_truth__lo_ground_truth:meansTR
r_worker_id__meansTRUE__lo_ground_truth:meansTRU
cor_worker_id__sd_diff15__lo_ground_truth:meansTRUE
cor_worker_id__Intercept__lo_ground_truth:sd_diff15
r_worker_id__lo_ground_truth__lo_ground_truth:sd_diff1
cor_worker_id__meansTRUE__lo_ground_truth:sd_diff15
cor_worker_id__sd_diff15__lo_ground_truth:sd_diff15
ker_id__lo_ground_truth:meansTRUE__lo_ground_truth:s
cor_worker_id__Intercept__meansTRUE:sd_diff15
Chain
1
2

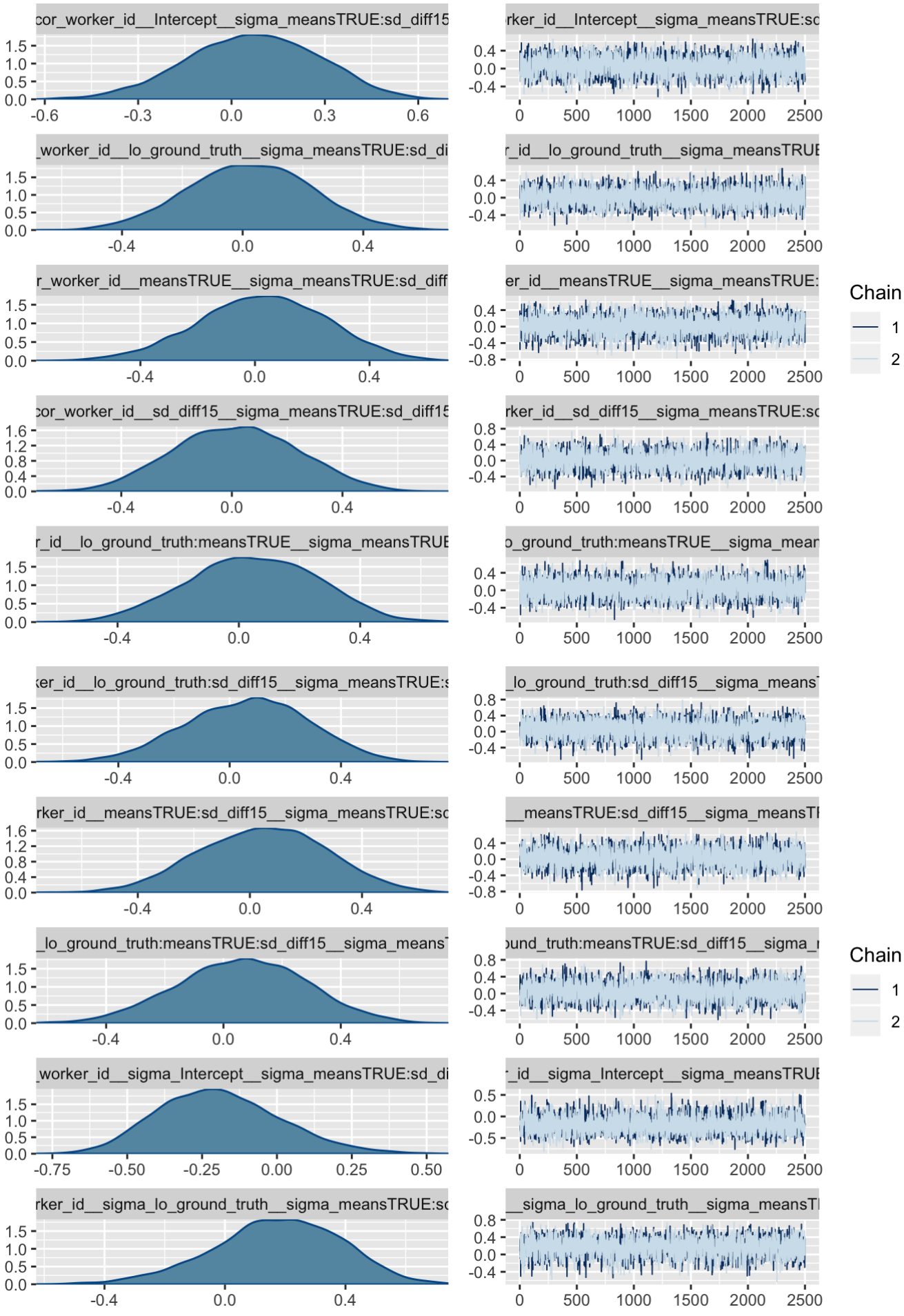

cor_worker_id__lo_ground_truth__meansTRUE:sd_diff15
rker_id__lo_ground_truth__meansTRUE:sd
cor_worker_id__meansTRUE__meansTRUE:sd_diff15
orker_id__meansTRUE__meansTRUE:sd_
cor_worker_id__sd_diff15__meansTRUE:sd_diff15
_worker_id__sd_diff15__meansTRUE:sd_di
rker_id__lo_ground_truth:meansTRUE__meansTRUE:sd
__lo_ground_truth:meansTRUE__meansTR
worker_id__lo_ground_truth:sd_diff15__meansTRUE:sd_
id__lo_ground_truth:sd_diff15__meansTRU
worker_id__Intercept__lo_ground_truth:meansTRUE:sd_
id__Intercept__lo_ground_truth:meansTRU
ker_id__lo_ground_truth__lo_ground_truth:meansTRUE:s
_lo_ground_truth__lo_ground_truth:meansT
rker_id__meansTRUE__lo_ground_truth:meansTRUE:sd
__meansTRUE__lo_ground_truth:meansTR
worker_id__sd_diff15__lo_ground_truth:meansTRUE:sd_
id__sd_diff15__lo_ground_truth:meansTRU
_lo_ground_truth:meansTRUE__lo_ground_truth:meansT
ound_truth:meansTRUE__lo_ground_truth:r
Chain
1
2
Chain
1
2

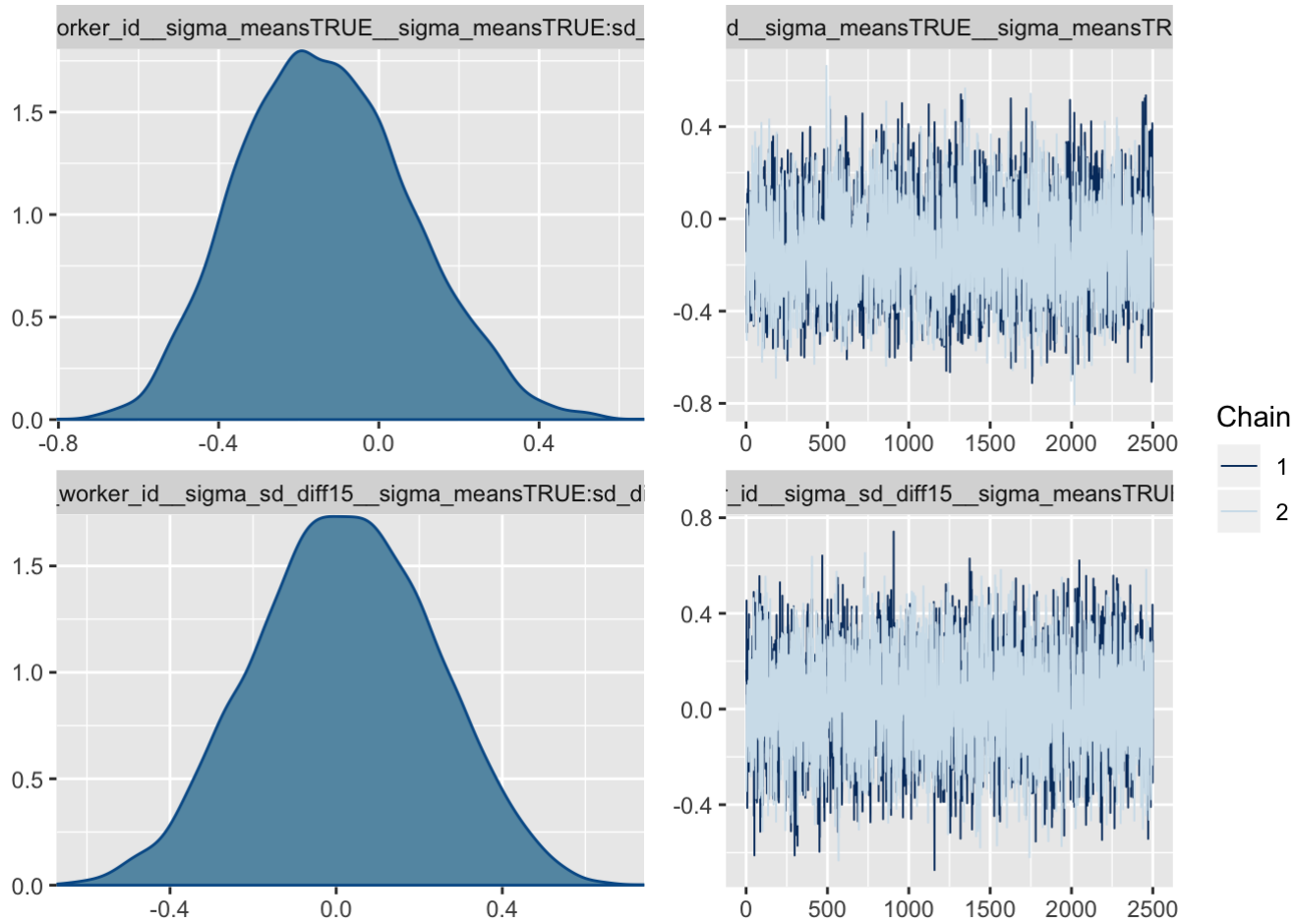

d__lo_ground_truth:sd_diff15__lo_ground_truth:meansTR
round_truth:sd_diff15__lo_ground_truth:m
_id__meansTRUE:sd_diff15__lo_ground_truth:meansTRU
ansTRUE:sd_diff15__lo_ground_truth:mea
cor_worker_id__Intercept__sigma_Intercept
or_worker_id__Intercept__sigma_Intercep
cor_worker_id__lo_ground_truth__sigma_Intercept
worker_id__lo_ground_truth__sigma_Inter
cor_worker_id__meansTRUE__sigma_Intercept
_worker_id__meansTRUE__sigma_Interc
cor_worker_id__sd_diff15__sigma_Intercept
or_worker_id__sd_diff15__sigma_Intercep
_worker_id__lo_ground_truth:meansTRUE__sigma_Interc
_id__lo_ground_truth:meansTRUE__sigm
or_worker_id__lo_ground_truth:sd_diff15__sigma_Interce
er_id__lo_ground_truth:sd_diff15__sigma_
cor_worker_id__meansTRUE:sd_diff15__sigma_Intercept
ker_id__meansTRUE:sd_diff15__sigma_I
ker_id__lo_ground_truth:meansTRUE:sd_diff15__sigma_
_lo_ground_truth:meansTRUE:sd_diff15__
Chain
1
2
Chain
1
2


cor_worker_id__Intercept__sigma_lo_ground_truth
worker_id__Intercept__sigma_lo_ground_t
cor_worker_id__lo_ground_truth__sigma_lo_ground_truth
rker_id__lo_ground_truth__sigma_lo_groun
cor_worker_id__meansTRUE__sigma_lo_ground_truth
orker_id__meansTRUE__sigma_lo_ground
cor_worker_id__sd_diff15__sigma_lo_ground_truth
worker_id__sd_diff15__sigma_lo_ground_t
rker_id__lo_ground_truth:meansTRUE__sigma_groun
__lo_ground_truth:meansTRUE__sigma_lo
worker_id__lo_ground_truth:sd_diff15__sigma_lo_ground_
id__lo_ground_truth:sd_diff15__sigma_lo_g
_worker_id__meansTRUE:sd_diff15__sigma_lo_ground_t
r_id__meansTRUE:sd_diff15__sigma_lo_gr
id__lo_ground_truth:meansTRUE:sd_diff15__sigma_lo_g
_ground_truth:meansTRUE:sd_diff15__sigm
cor_worker_id__sigma_Intercept__sigma_lo_ground_truth
rker_id__sigma_Intercept__sigma_lo_groun
cor_worker_id__Intercept__sigma_meansTRUE
r_worker_id__Intercept__sigma_meansTRU
Chain
1
2
Chain
1
2


cor_worker_id__lo_ground_truth__sigma_meansTRUE
cor_worker_id__meansTRUE__sigma_meansTRUE
cor_worker_id__sd_diff15__sigma_meansTRUE
cor_worker_id__lo_ground_truth:meansTRUE__sigma_meansTRUE
cor_worker_id__lo_ground_truth:sd_diff15__sigma_meansTRUE
cor_worker_id__meansTRUE:sd_diff15__sigma_meansTRUE
cor_worker_id__lo_ground_truth:meansTRUE:sd_diff15__sigma_meansTRUE
cor_worker_id__sigma_Intercept__sigma_meansTRUE
cor_worker_id__sigma_lo_ground_truth__sigma_meansTRUE
cor_worker_id__Intercept__sigma_sd_diff15
Chain
1
2


cor_worker_id__lo_ground_truth__sigma_sd_diff15
cor_worker_id__meansTRUE__sigma_sd_diff15
cor_worker_id__sd_diff15__sigma_sd_diff15
worker_id__lo_ground_truth:meansTRUE__sigma_sd_di
r_worker_id__lo_ground_truth:sd_diff15__sigma_sd_diff
cor_worker_id__meansTRUE:sd_diff15__sigma_sd_diff15
ker_id__lo_ground_truth:meansTRUE:sd_diff15__sigma_s
cor_worker_id__sigma_Intercept__sigma_sd_diff15
cor_worker_id__sigma_lo_ground_truth__sigma_sd_diff15
cor_worker_id__sigma_meansTRUE__sigma_sd_diff15
Chain
1
2


cor_worker_id__Intercept__sigma_meansTRUE:sd_diff15
rker_id__Intercept__sigma_meansTRUE:sc
_worker_id__lo_ground_truth__sigma_meansTRUE:sd_di
r_id__lo_ground_truth__sigma_meansTRUE
r_worker_id__meansTRUE__sigma_meansTRUE:sd_diff
er_id__meansTRUE__sigma_meansTRUE:
cor_worker_id__sd_diff15__sigma_meansTRUE:sd_diff15
rker_id__sd_diff15__sigma_meansTRUE:sc
r_id__lo_ground_truth:meansTRUE__sigma_meansTRUE
o_ground_truth:meansTRUE__sigma_mean
ker_id__lo_ground_truth:sd_diff15__sigma_meansTRUE:s
_lo_ground_truth:sd_diff15__sigma_means
rker_id__meansTRUE:sd_diff15__sigma_meansTRUE:sc
__meansTRUE:sd_diff15__sigma_meansTR
_lo_ground_truth:meansTRUE:sd_diff15__sigma_means
ound_truth:meansTRUE:sd_diff15__sigma_
_worker_id__sigma_Intercept__sigma_meansTRUE:sd_di
_id__sigma_Intercept__sigma_meansTRUE
rker_id__sigma_lo_ground_truth__sigma_meansTRUE:sc
__sigma_lo_ground_truth__sigma_meansTR
Chain
1
2
Chain
1
2

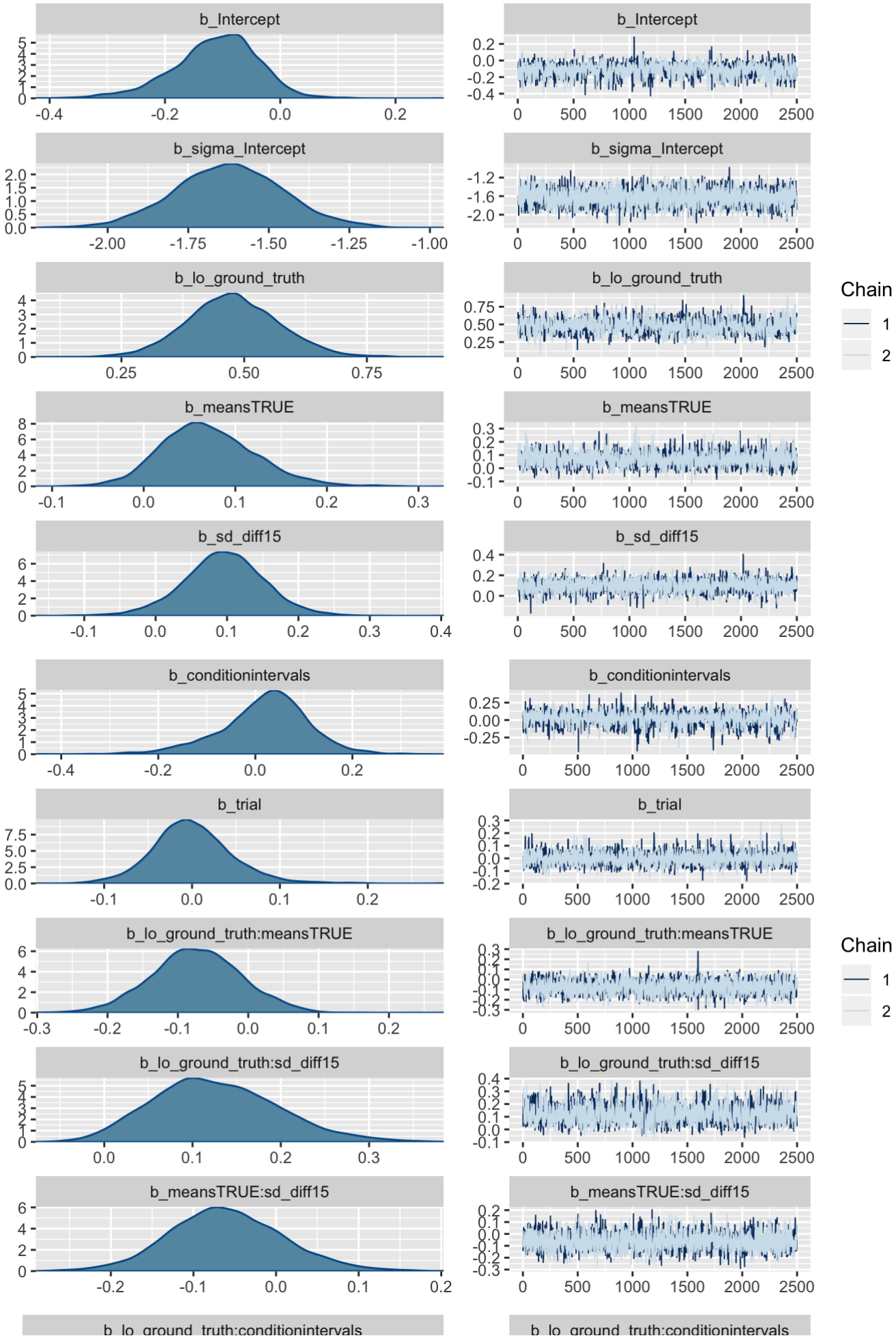


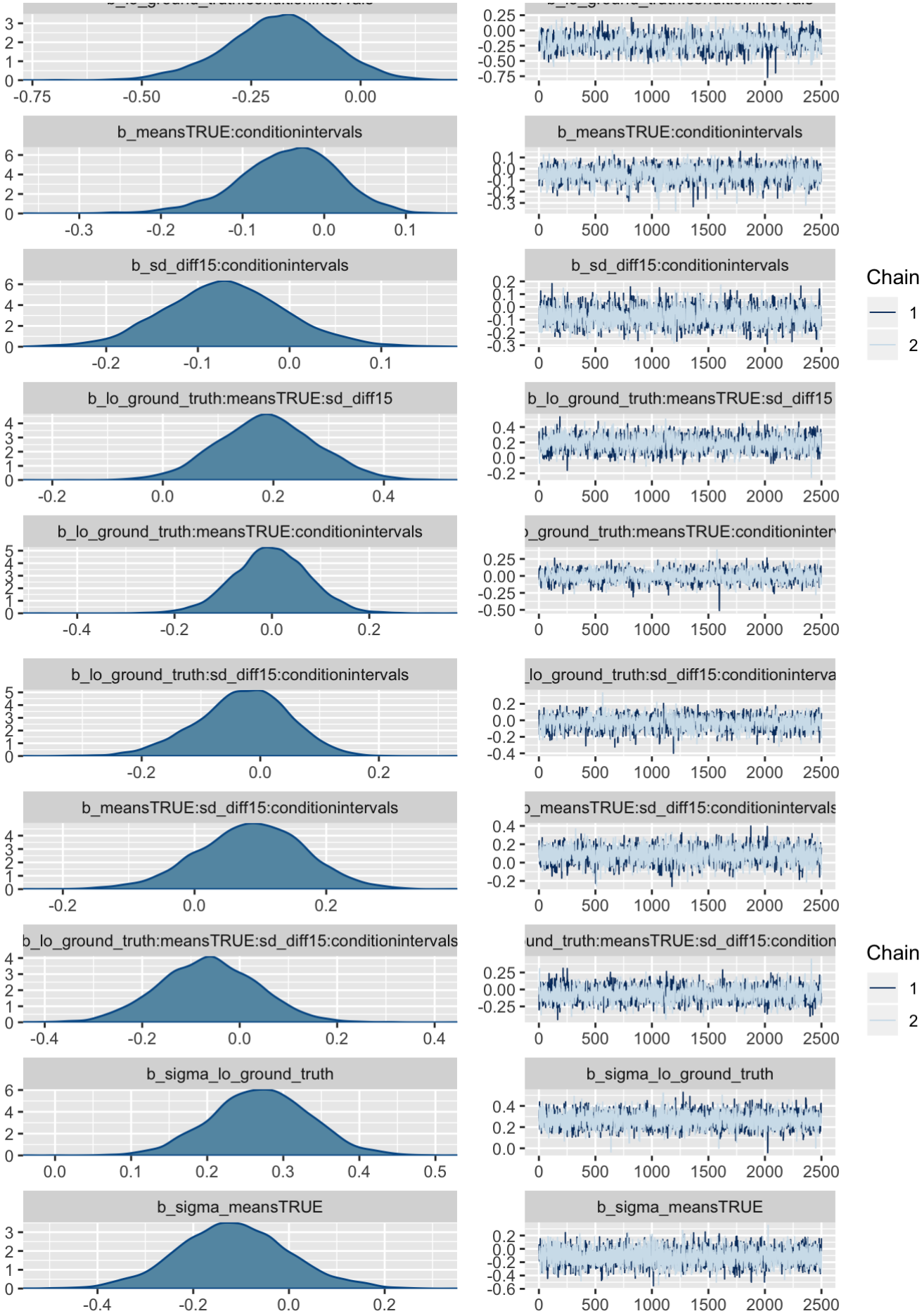
Mixed Effects of Trial Order on Mean Response

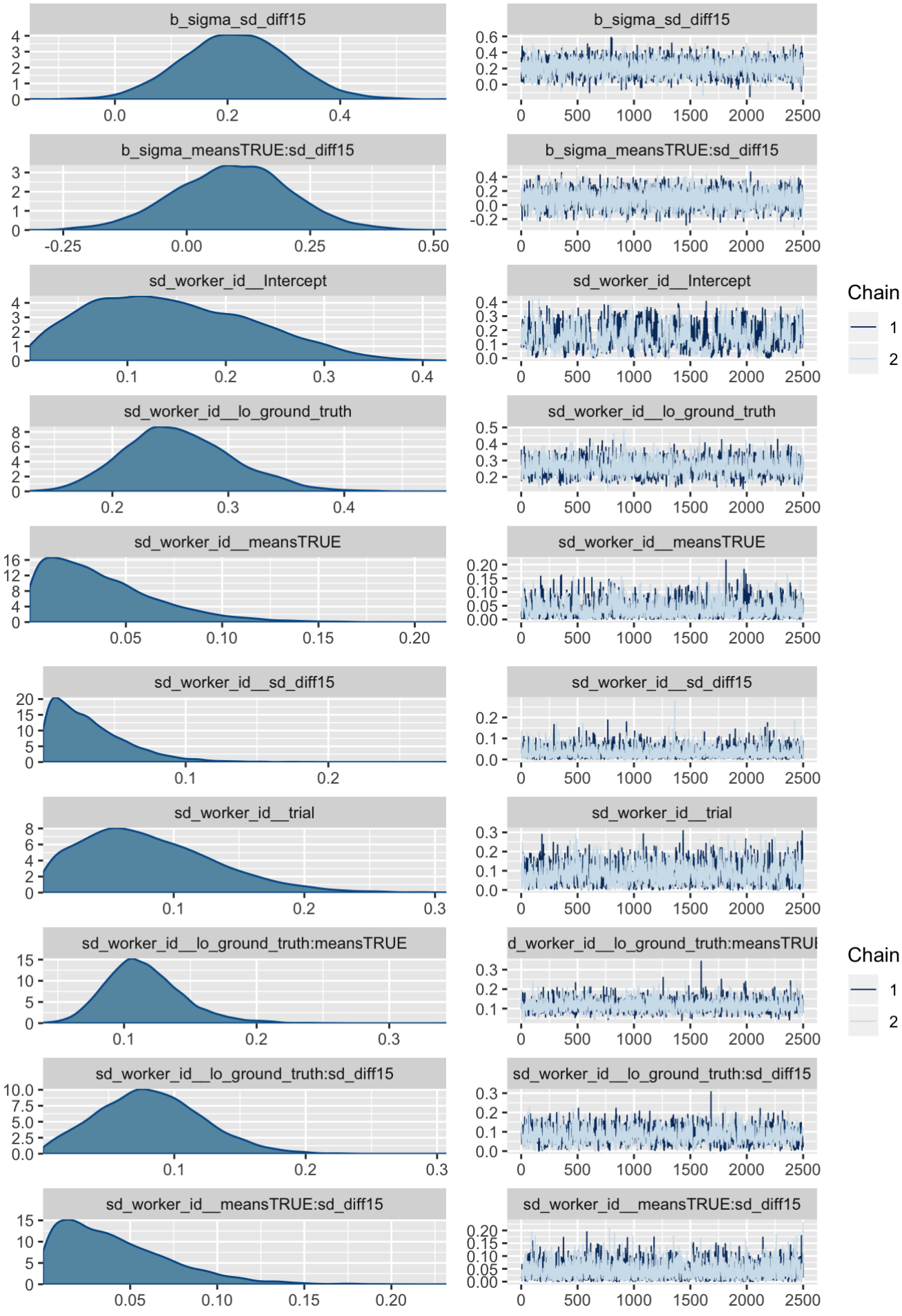
This model adds mixed effects of trial order to the LLO model. This is effectively modeling a learning effect on the pattern of mean response at each level of ground truth.

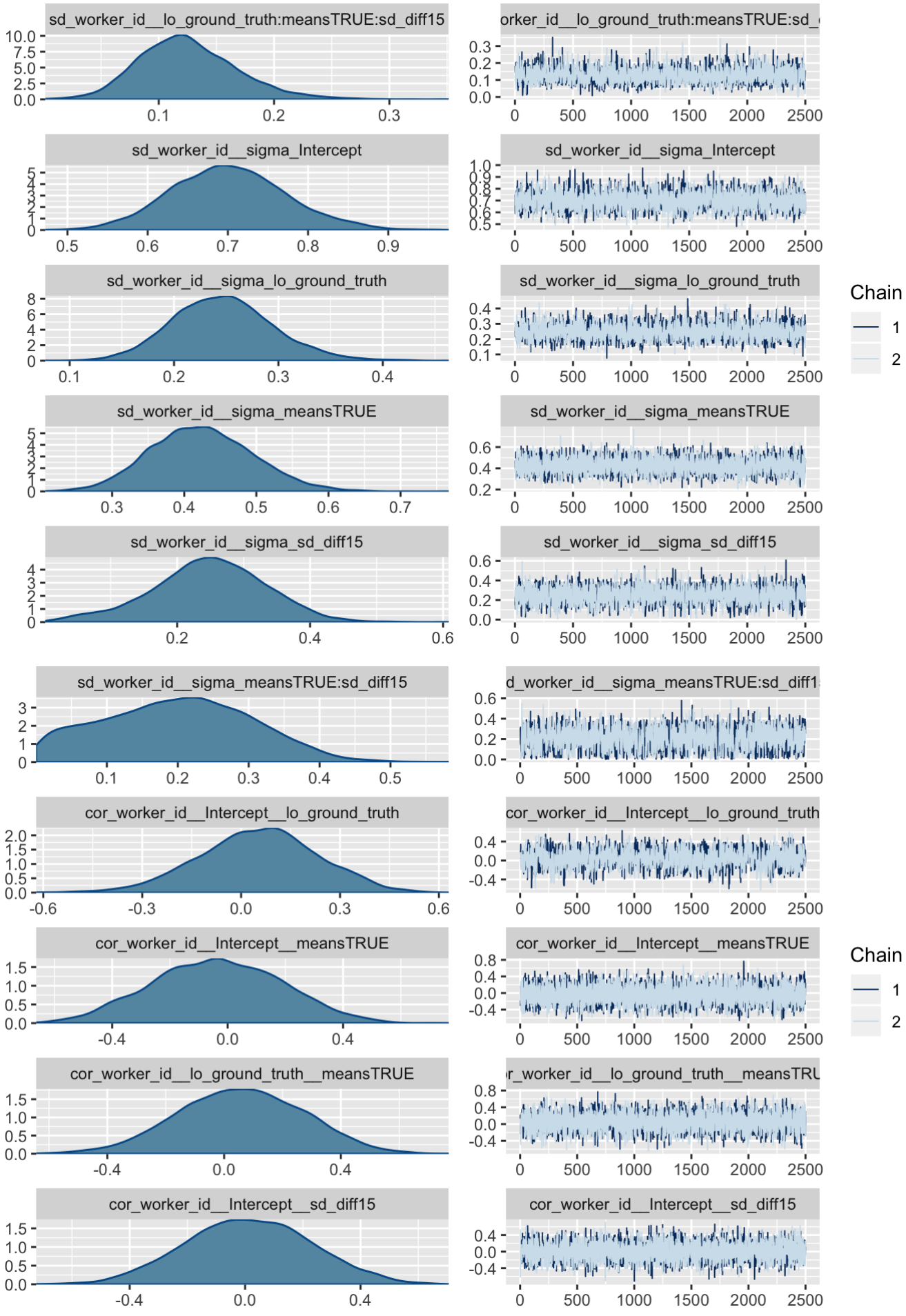
```
# hierarchical LLO model
m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.trial.sigma.gt.means.sd <- brm(
  data = model_df, family = "gaussian",
  formula = bf(lo_p_sup ~  (1 + lo_ground_truth*means*sd_diff + trial|sharecor|worker_i
d) + lo_ground_truth*means*sd_diff*condition + trial,
               sigma ~ (1 + lo_ground_truth + means*sd_diff|sharecor|worker_id) + lo_gro
und_truth + means*sd_diff),
  prior = c(prior(normal(1, 0.5), class = b),
            prior(normal(1.3, 1), class = Intercept),
            prior(normal(0, 0.15), class = sd, group = worker_id),
            prior(normal(0, 0.3), class = b, dpar = sigma),
            prior(normal(0, 0.15), class = sd, dpar = sigma),
            prior(lkj(4), class = cor)),
  iter = 3000, warmup = 500, chains = 2, cores = 2,
  control = list(adapt_delta = 0.99, max_treedepth = 12),
  file = "model-fits/llo_mdl-wrkr_means_sd_vis-r_means_sd_trial_sigma_gt_means_sd")
```

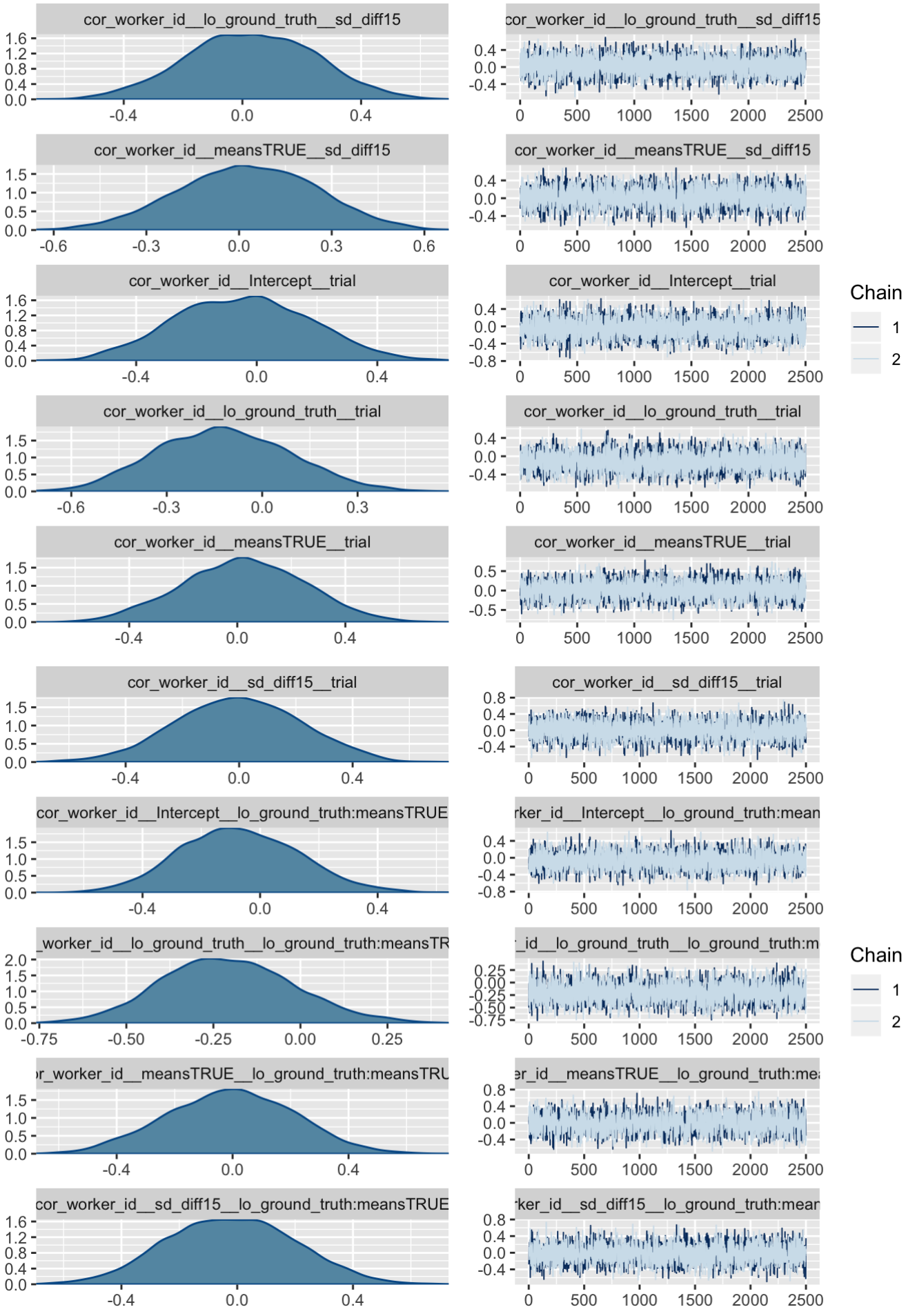
```
plot(m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.trial.sigma.gt.means.sd)
```

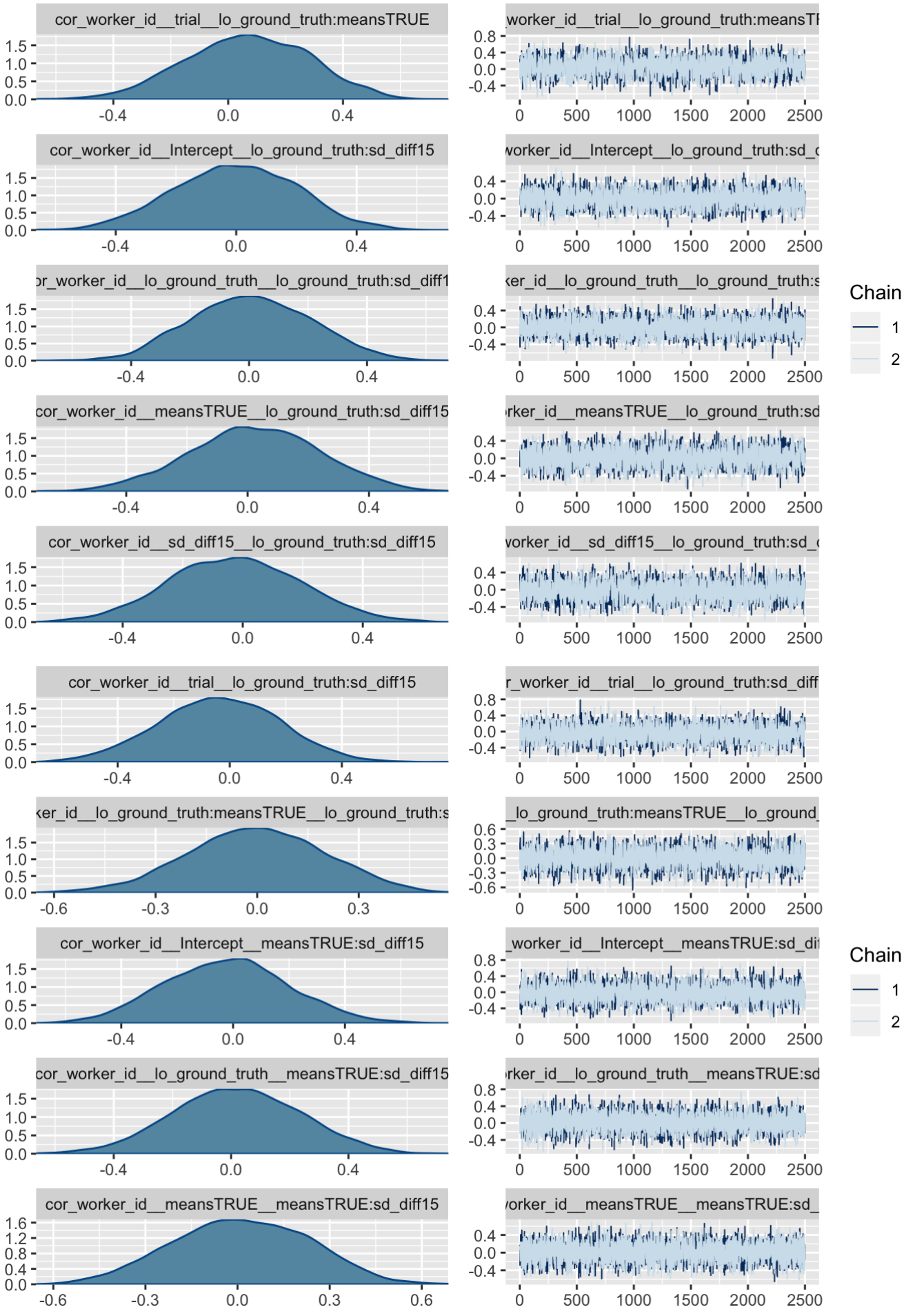

b_Intercept
b_sigma_Intercept
b_lo_ground_truth
b_meansTRUE
b_sd_diff15
b_conditionintervals
b_trial
b_lo_ground_truth:meansTRUE
b_lo_ground_truth:sd_diff15
b_meansTRUE:sd_diff15
b_lo_ground_truth:conditionintervals
Chain
1
2

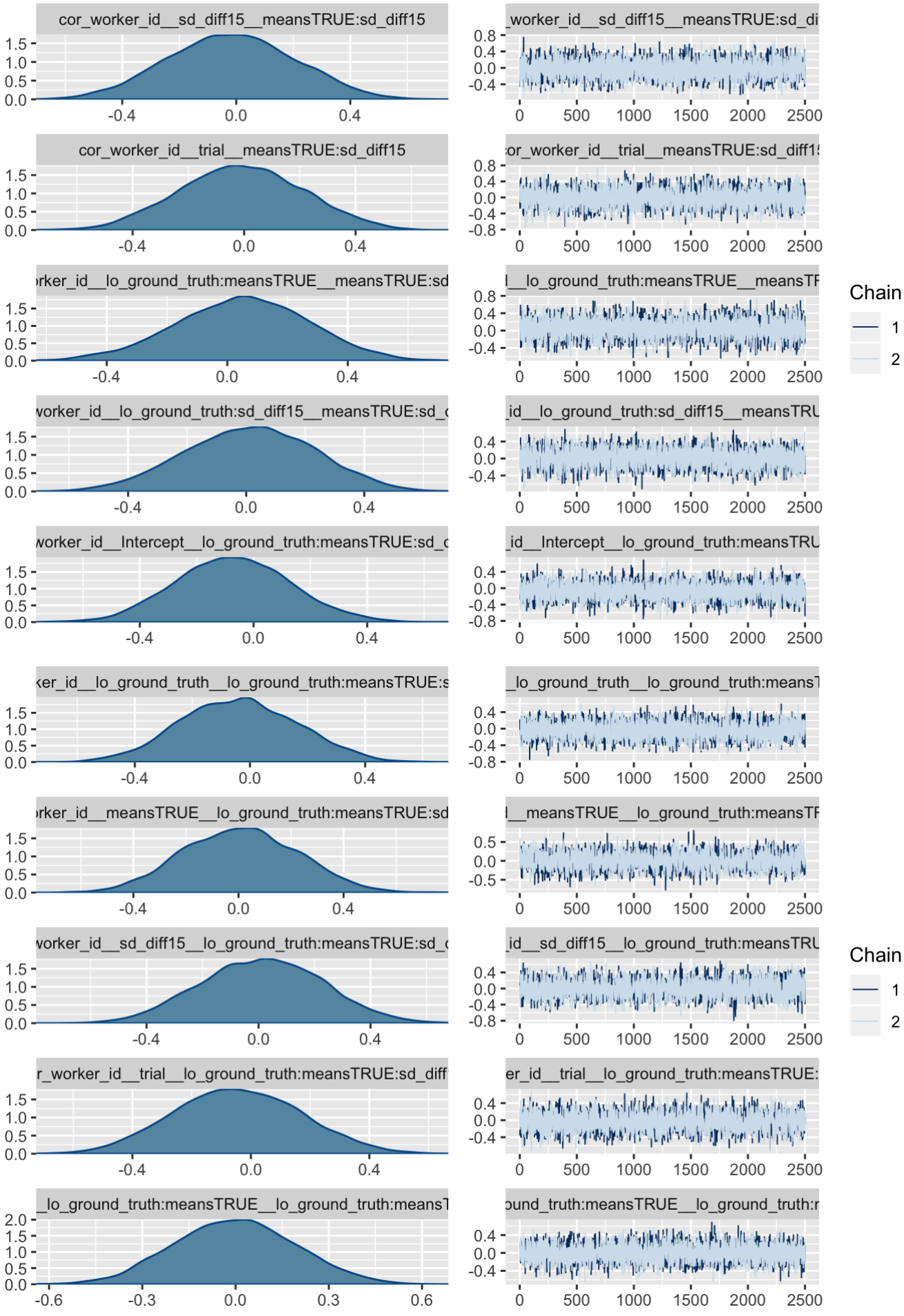

b_meansTRUE:conditionintervals
b_sd_diff15:conditionintervals
b_lo_ground_truth:meansTRUE:sd_diff15
b_lo_ground_truth:meansTRUE:conditionintervals
b_lo_ground_truth:sd_diff15:conditionintervals
b_meansTRUE:sd_diff15:conditionintervals
b_lo_ground_truth:meansTRUE:sd_diff15:conditionintervals
b_sigma_lo_ground_truth
b_sigma_meansTRUE
Chain
1
2

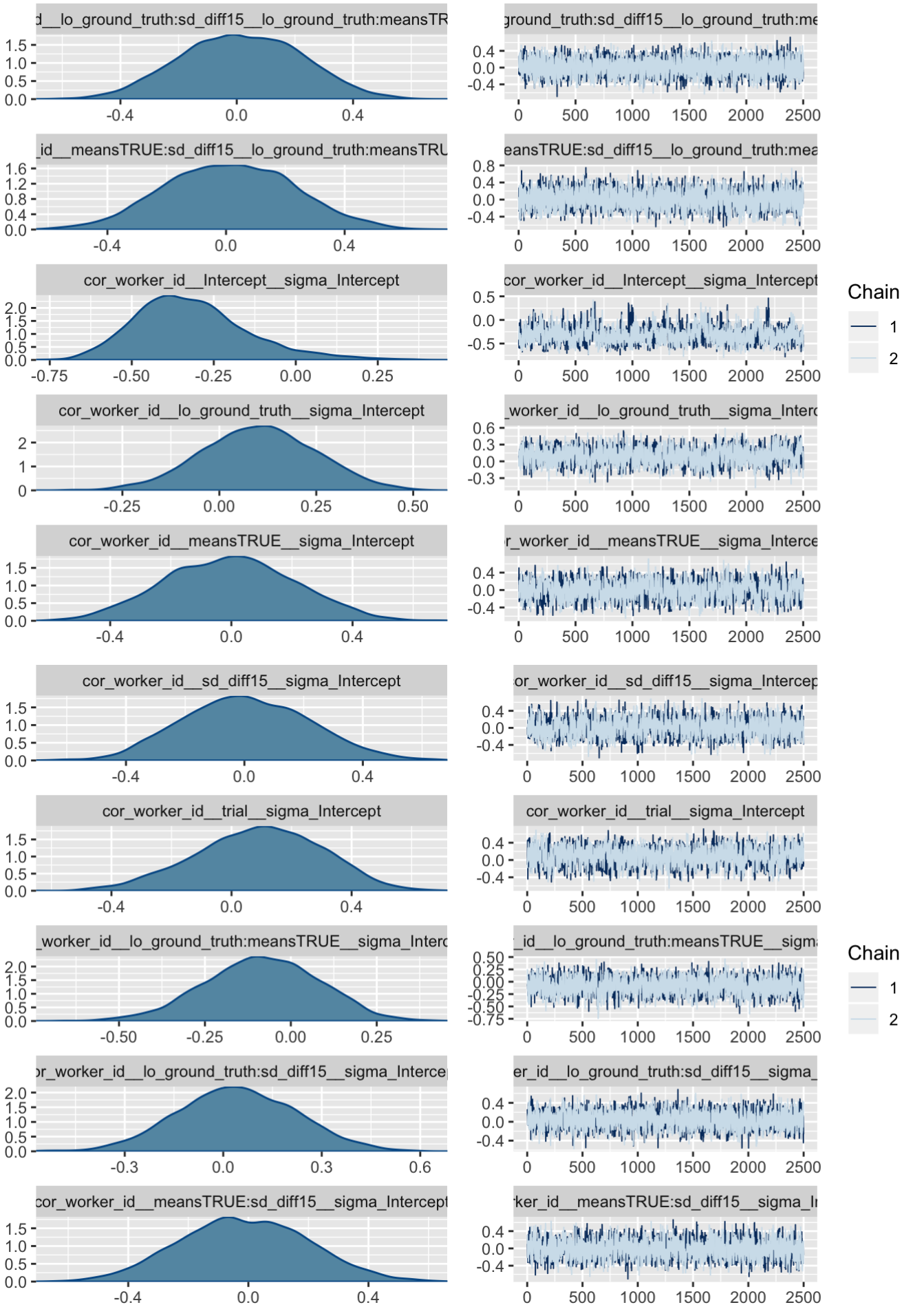

b_sigma_sd_diff15
b_sigma_meansTRUE:sd_diff15
sd_worker_id__Intercept
sd_worker_id__lo_ground_truth
sd_worker_id__meansTRUE
sd_worker_id__sd_diff15
sd_worker_id__trial
sd_worker_id__lo_ground_truth:meansTRUE
sd_worker_id__lo_ground_truth:sd_diff15
sd_worker_id__meansTRUE:sd_diff15
Chain
1
2

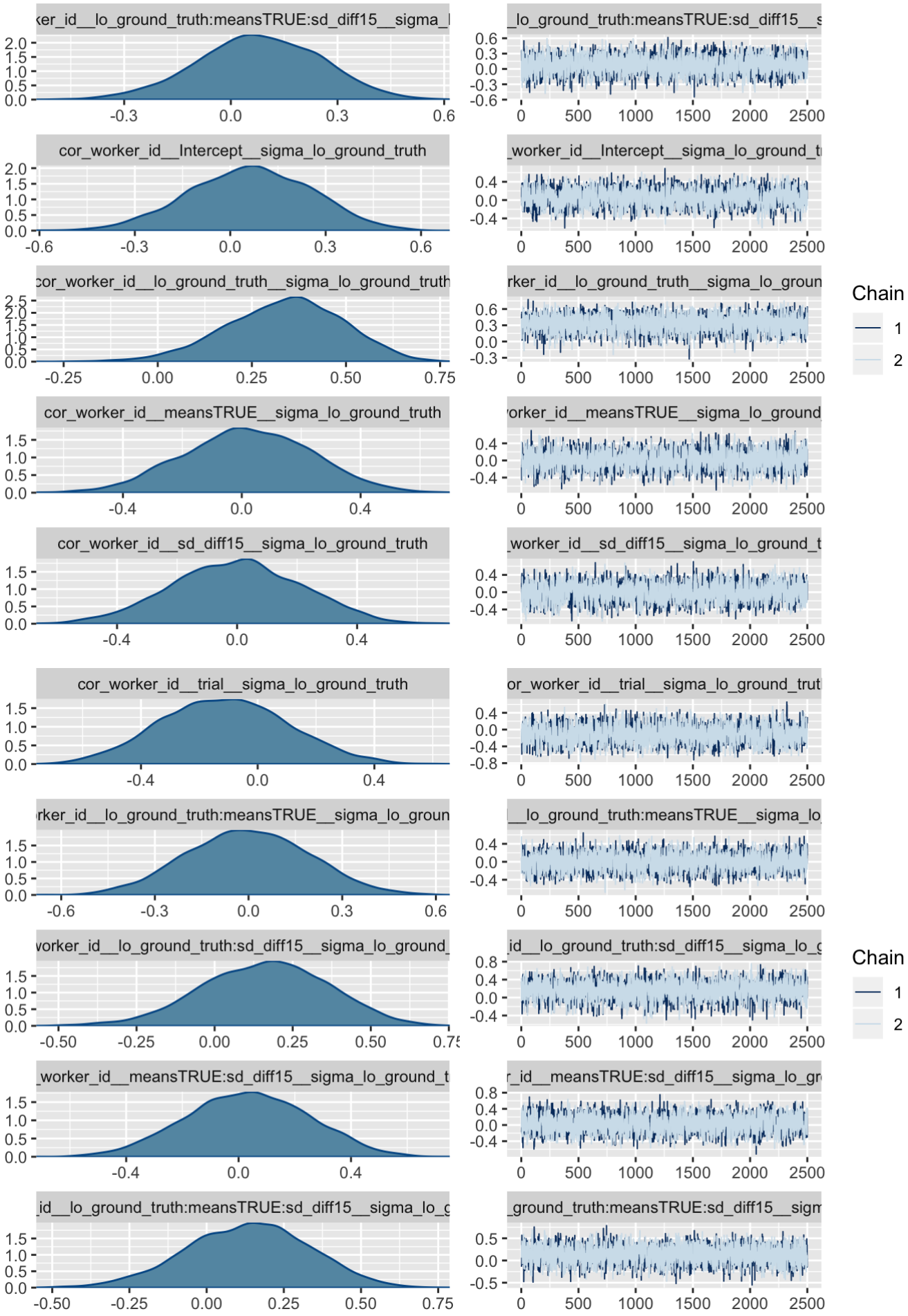

sd_worker_id__lo_ground_truth:meansTRUE:sd_diff15
sd_worker_id__sigma_Intercept
sd_worker_id__sigma_lo_ground_truth
sd_worker_id__sigma_meansTRUE
sd_worker_id__sigma_sd_diff15
sd_worker_id__sigma_meansTRUE:sd_diff15
cor_worker_id__Intercept__lo_ground_truth
cor_worker_id__Intercept__meansTRUE
cor_worker_id__lo_ground_truth__meansTRUE
cor_worker_id__Intercept__sd_diff15
Chain
1
2

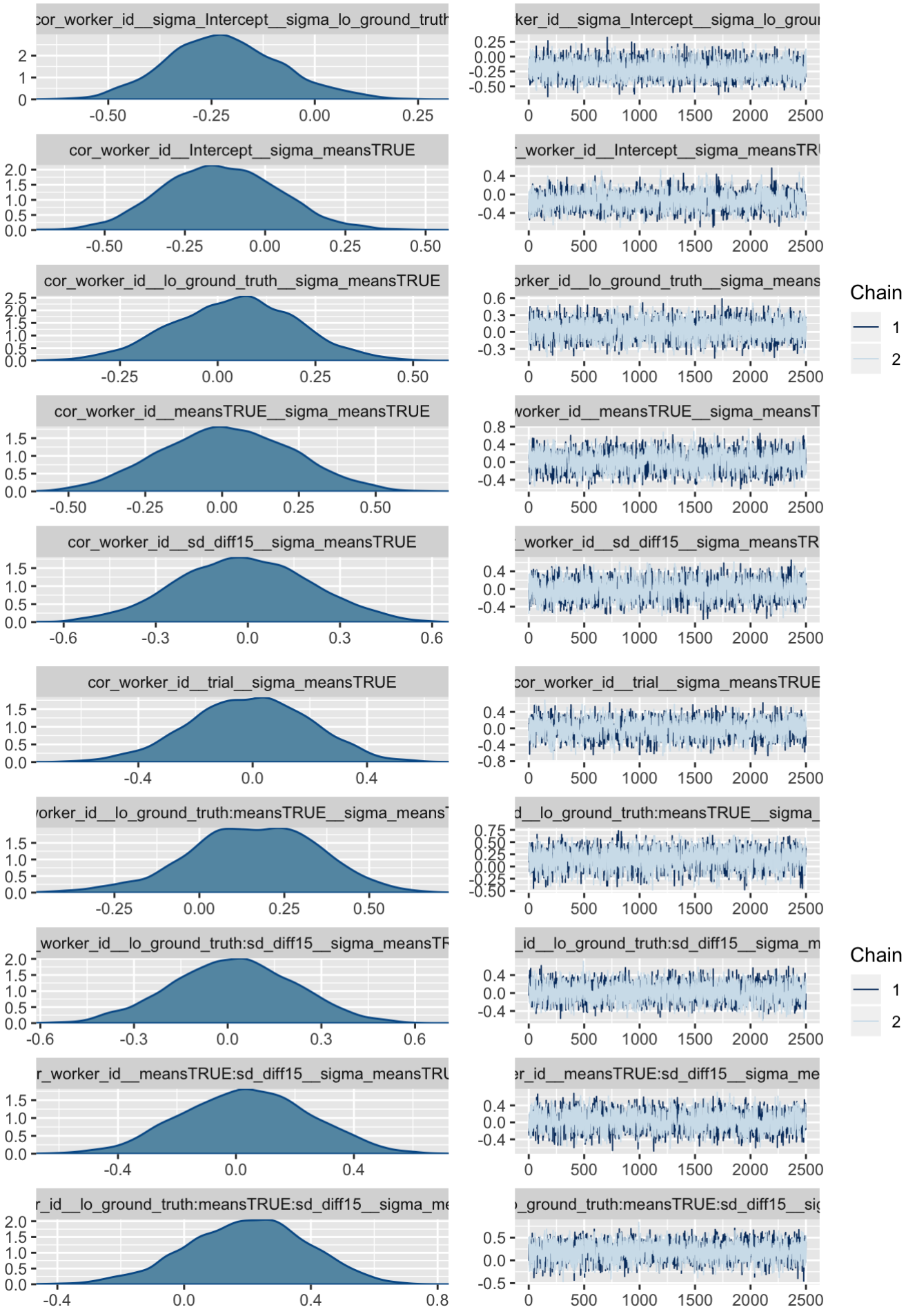

cor_worker_id__lo_ground_truth__sd_diff15
cor_worker_id__meansTRUE__sd_diff15
cor_worker_id__Intercept__trial
cor_worker_id__lo_ground_truth__trial
cor_worker_id__meansTRUE__trial
cor_worker_id__sd_diff15__trial
cor_worker_id__Intercept__lo_ground_truth:meansTRUE
_worker_id__lo_ground_truth__lo_ground_truth:meansTR
r_worker_id__meansTRUE__lo_ground_truth:meansTRU
cor_worker_id__sd_diff15__lo_ground_truth:meansTRUE
Chain
1
2

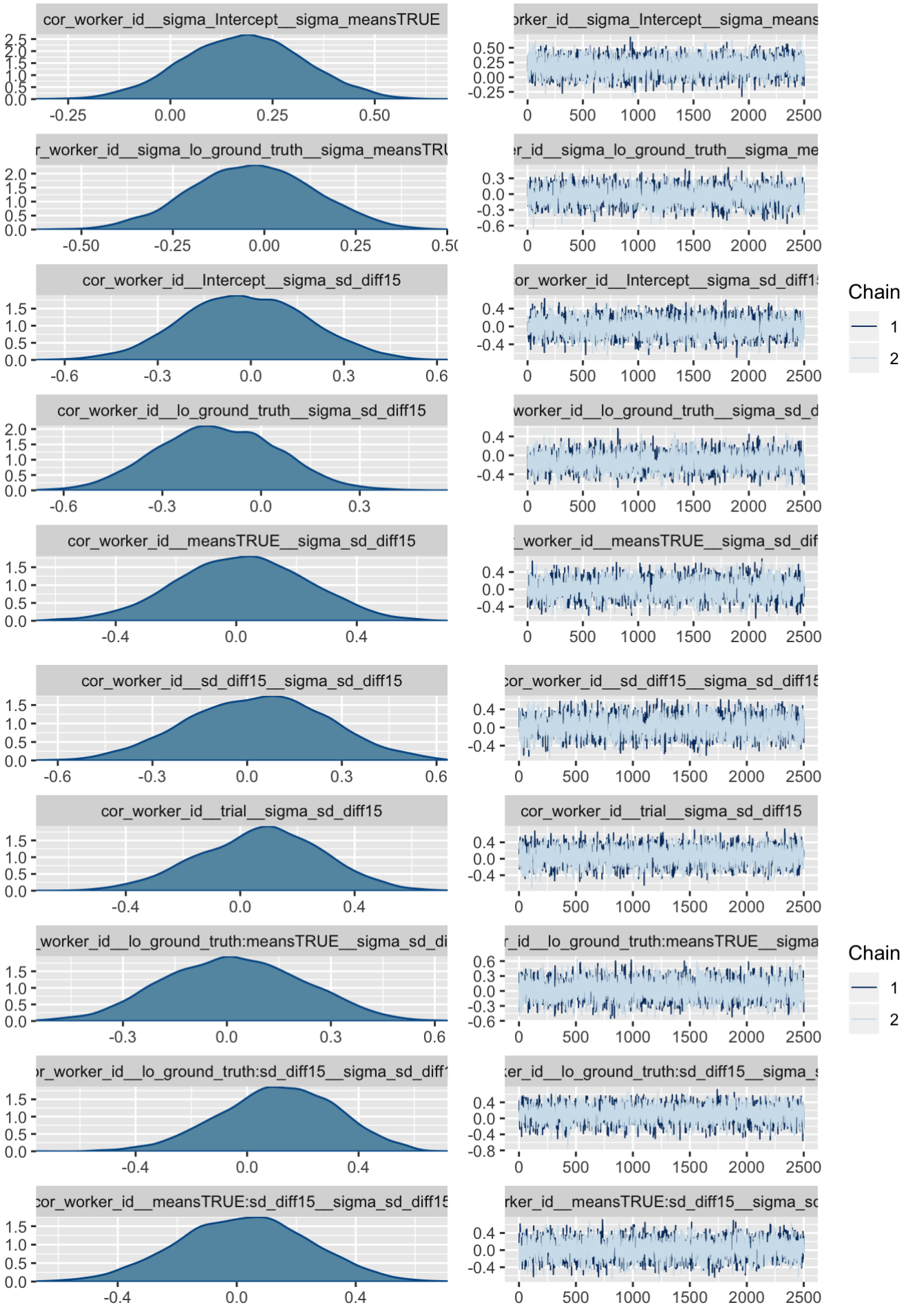

cor_worker_id__trial__lo_ground_truth:meansTRUE
cor_worker_id__Intercept__lo_ground_truth:sd_diff15
cor_worker_id__lo_ground_truth__lo_ground_truth:sd_diff1
cor_worker_id__meansTRUE__lo_ground_truth:sd_diff15
cor_worker_id__sd_diff15__lo_ground_truth:sd_diff15
cor_worker_id__trial__lo_ground_truth:sd_diff15
cor_worker_id__lo_ground_truth:meansTRUE__lo_ground_truth:s
cor_worker_id__Intercept__meansTRUE:sd_diff15
cor_worker_id__lo_ground_truth__meansTRUE:sd_diff15
cor_worker_id__meansTRUE__meansTRUE:sd_diff15
Chain
1
2

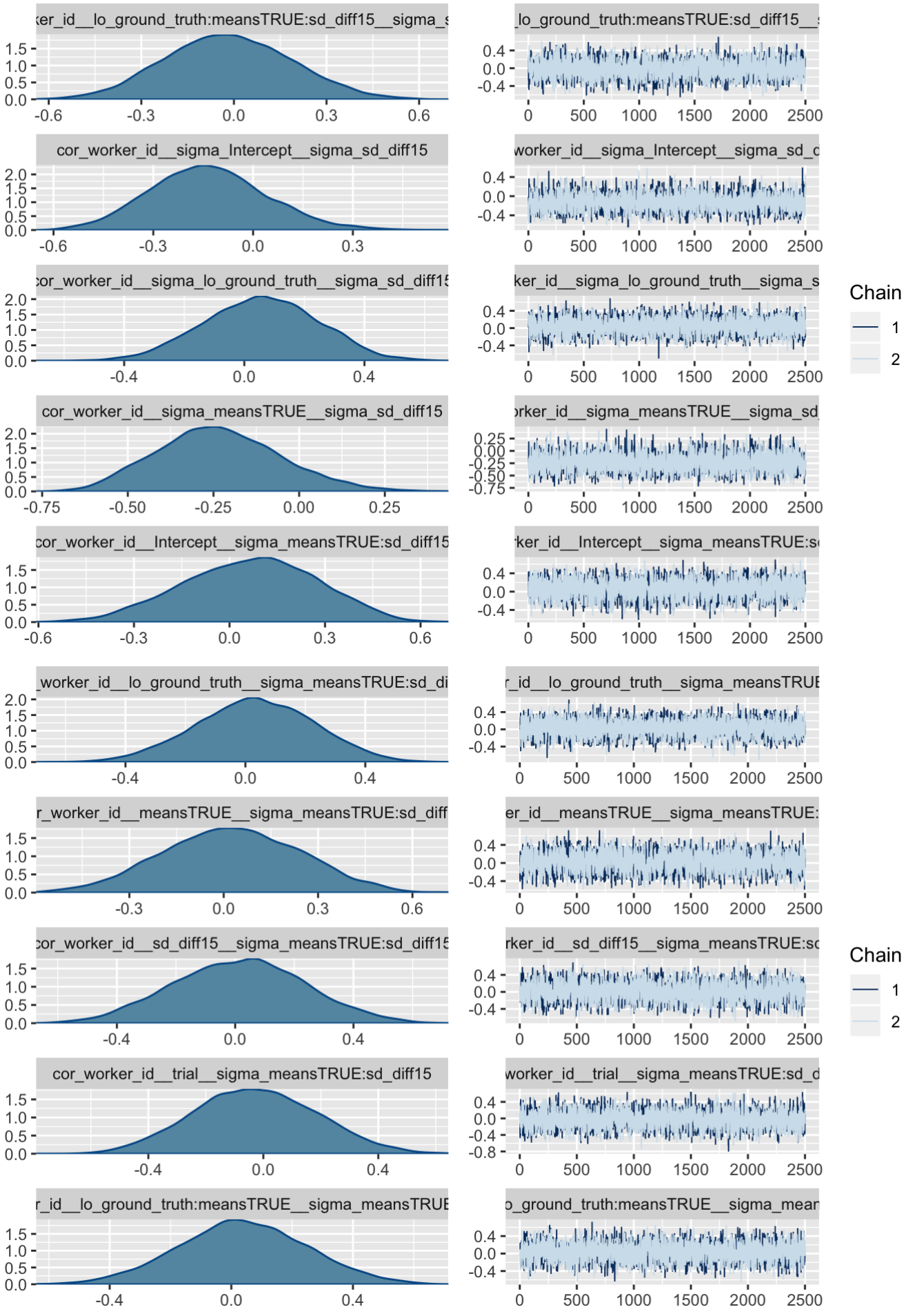

cor_worker_id__sd_diff15__meansTRUE:sd_diff15
cor_worker_id__trial__meansTRUE:sd_diff15
Chain
1
2

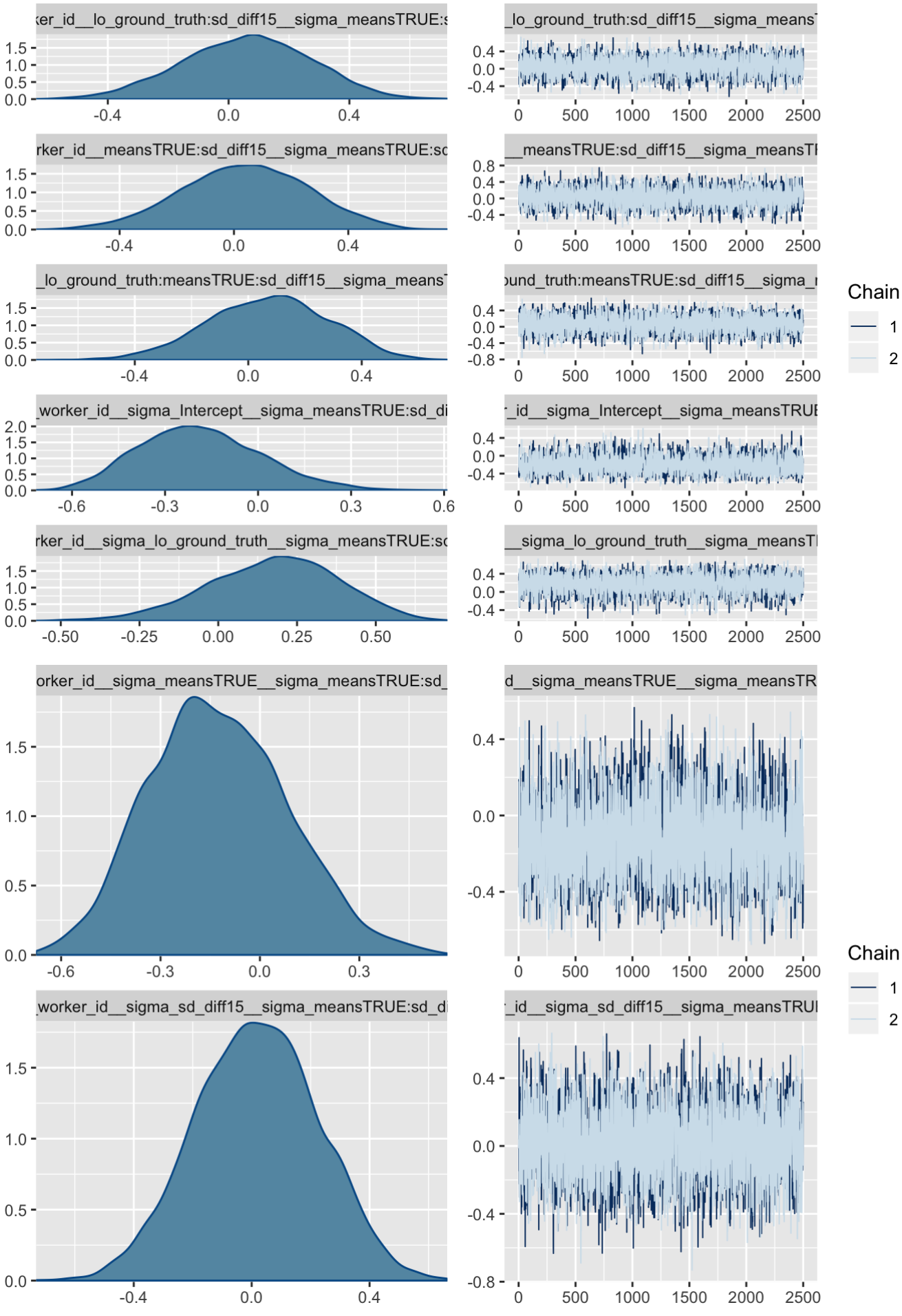

d__lo_ground_truth:sd_diff15__lo_ground_truth:meansTR
ground_truth:sd_diff15__lo_ground_truth:me
_id__meansTRUE:sd_diff15__lo_ground_truth:meansTRU
eansTRUE:sd_diff15__lo_ground_truth:mea
cor_worker_id__Intercept__sigma_Intercept
cor_worker_id__Intercept__sigma_Intercept
cor_worker_id__lo_ground_truth__sigma_Intercept
_worker_id__lo_ground_truth__sigma_Interc
cor_worker_id__meansTRUE__sigma_Intercept
r_worker_id__meansTRUE__sigma_Interce
cor_worker_id__sd_diff15__sigma_Intercept
or_worker_id__sd_diff15__sigma_Intercep
cor_worker_id__trial__sigma_Intercept
cor_worker_id__trial__sigma_Intercept
_worker_id__lo_ground_truth:meansTRUE__sigma_Interc
_id__lo_ground_truth:meansTRUE__sigma
or_worker_id__lo_ground_truth:sd_diff15__sigma_Interce
er_id__lo_ground_truth:sd_diff15__sigma_
cor_worker_id__meansTRUE:sd_diff15__sigma_Intercept
ker_id__meansTRUE:sd_diff15__sigma_I
Chain
1
2
Chain
1
2


cor_worker_id__Intercept__sigma_lo_ground_truth
cor_worker_id__lo_ground_truth__sigma_lo_ground_truth
cor_worker_id__meansTRUE__sigma_lo_ground_truth
cor_worker_id__sd_diff15__sigma_lo_ground_truth
cor_worker_id__trial__sigma_lo_ground_truth
Chain
1
2


cor_worker_id__sigma_Intercept__sigma_lo_ground_truth
cor_worker_id__Intercept__sigma_meansTRUE
cor_worker_id__lo_ground_truth__sigma_meansTRUE
cor_worker_id__meansTRUE__sigma_meansTRUE
cor_worker_id__sd_diff15__sigma_meansTRUE
cor_worker_id__trial__sigma_meansTRUE
worker_id__lo_ground_truth:meansTRUE__sigma_meansT
_worker_id__lo_ground_truth:sd_diff15__sigma_meansTR
r_worker_id__meansTRUE:sd_diff15__sigma_meansTRU
r_id__lo_ground_truth:meansTRUE:sd_diff15__sigma_me
Chain
1
2


cor_worker_id__sigma_Intercept__sigma_meansTRUE
cor_worker_id__sigma_lo_ground_truth__sigma_meansTRU
cor_worker_id__Intercept__sigma_sd_diff15
cor_worker_id__lo_ground_truth__sigma_sd_diff15
cor_worker_id__meansTRUE__sigma_sd_diff15
cor_worker_id__sd_diff15__sigma_sd_diff15
cor_worker_id__trial__sigma_sd_diff15
worker_id__lo_ground_truth:meansTRUE__sigma_sd_di
or_worker_id__lo_ground_truth:sd_diff15__sigma_sd_diff
cor_worker_id__meansTRUE:sd_diff15__sigma_sd_diff15
Chain
1
2


cor_worker_id__sigma_Intercept__sigma_sd_diff15
cor_worker_id__sigma_meansTRUE__sigma_sd_diff15
cor_worker_id__Intercept__sigma_meansTRUE:sd_diff15
cor_worker_id__trial__sigma_meansTRUE:sd_diff15
Chain
1
2

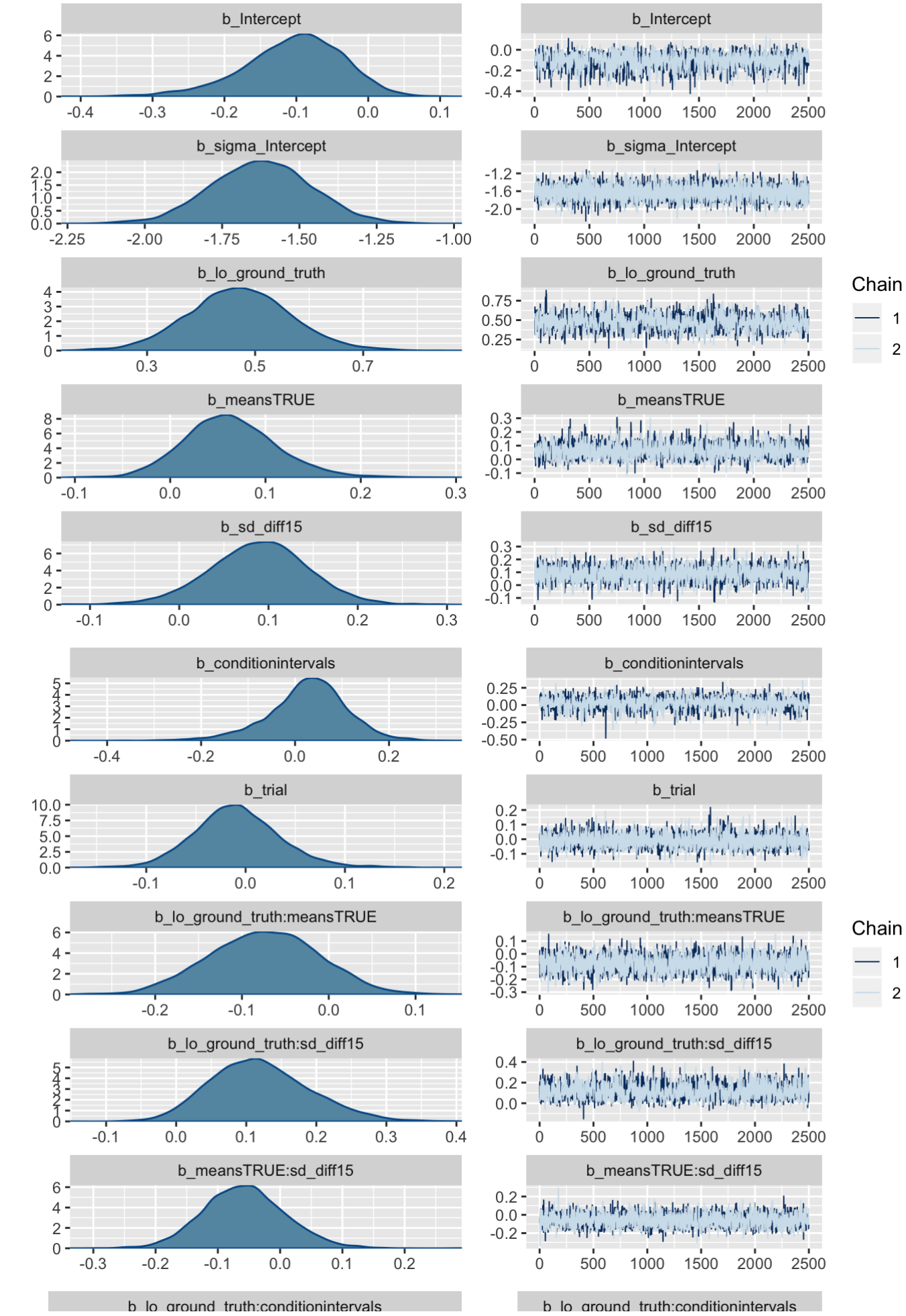

er_id__lo_ground_truth:sd_diff15__sigma_meansTRUE:s
_lo_ground_truth:sd_diff15__sigma_means
rker_id__meansTRUE:sd_diff15__sigma_meansTRUE:sc
__meansTRUE:sd_diff15__sigma_meansTI
_lo_ground_truth:meansTRUE:sd_diff15__sigma_means
ound_truth:meansTRUE:sd_diff15__sigma_
_worker_id__sigma_Intercept__sigma_meansTRUE:sd_di
_id__sigma_Intercept__sigma_meansTRUE
rker_id__sigma_lo_ground_truth__sigma_meansTRUE:sc
__sigma_lo_ground_truth__sigma_meansTI
orker_id__sigma_meansTRUE__sigma_meansTRUE:sd_
d__sigma_meansTRUE__sigma_meansTR
_worker_id__sigma_sd_diff15__sigma_meansTRUE:sd_di
_id__sigma_sd_diff15__sigma_meansTRUI
Chain
1
2
Chain
1
2

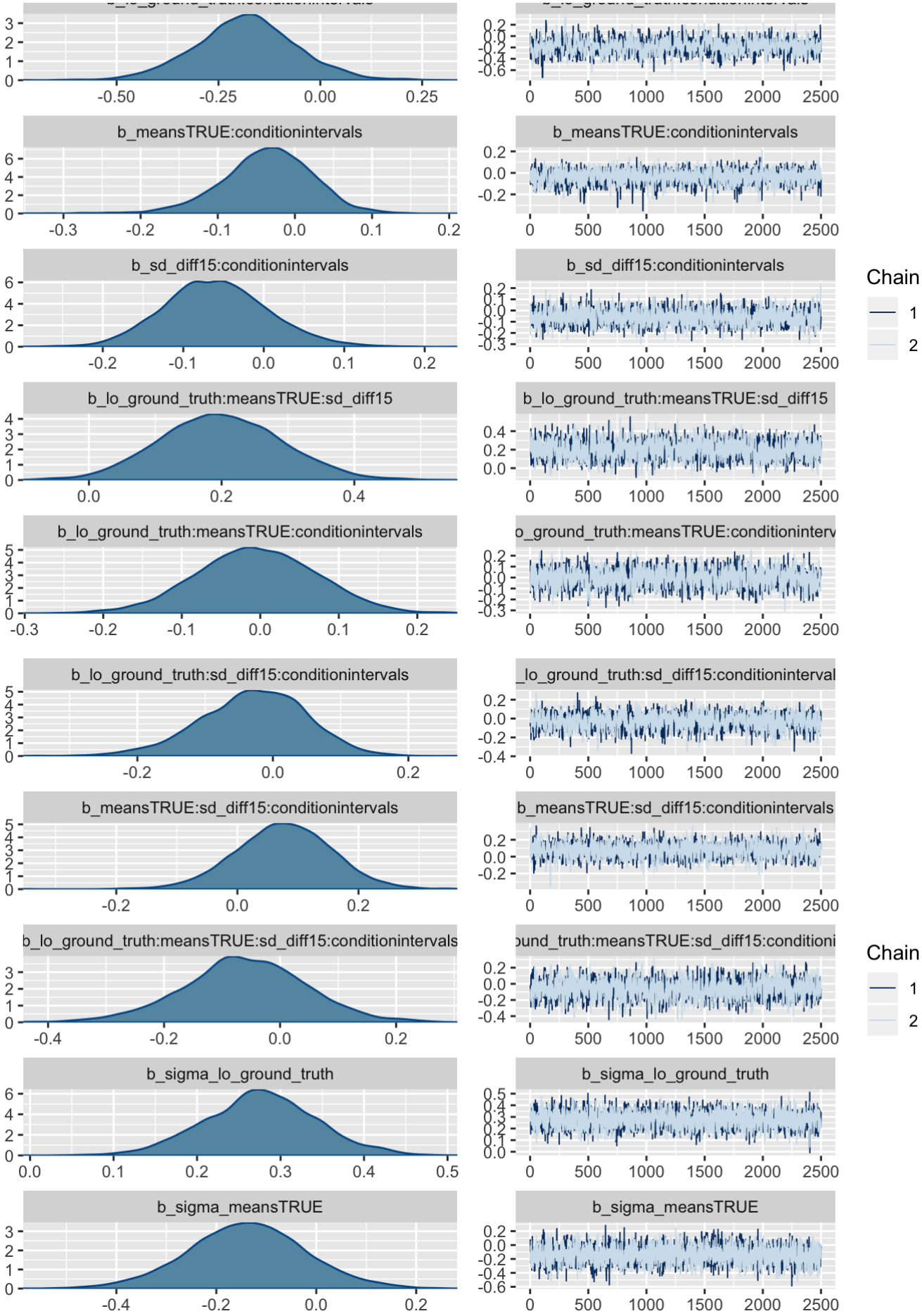
Mixed Effects of Trial Order on Sigma

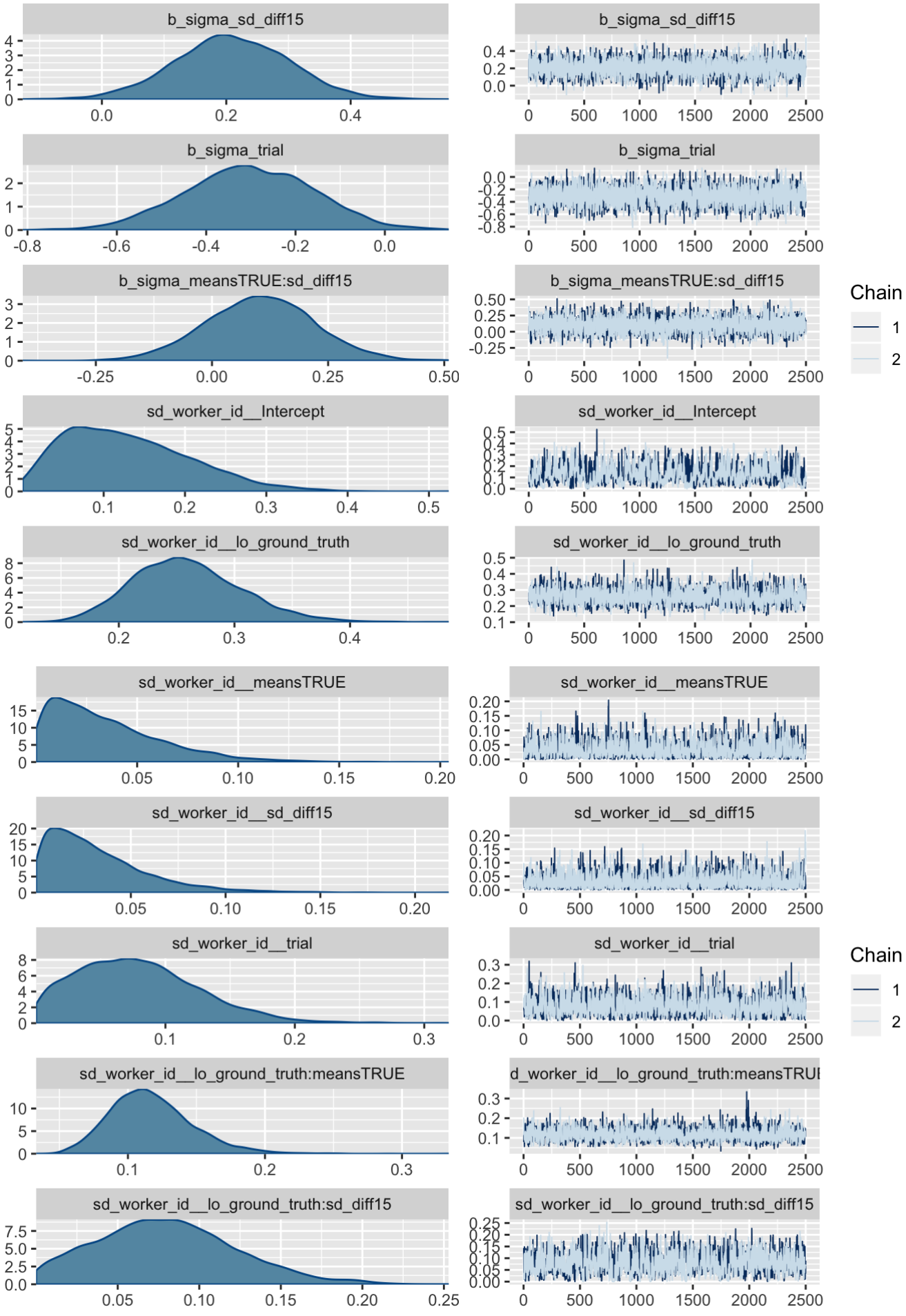
This model adds fixed and random intercepts of trial order to sigma submodel. This is a a learning effect on residual variance.

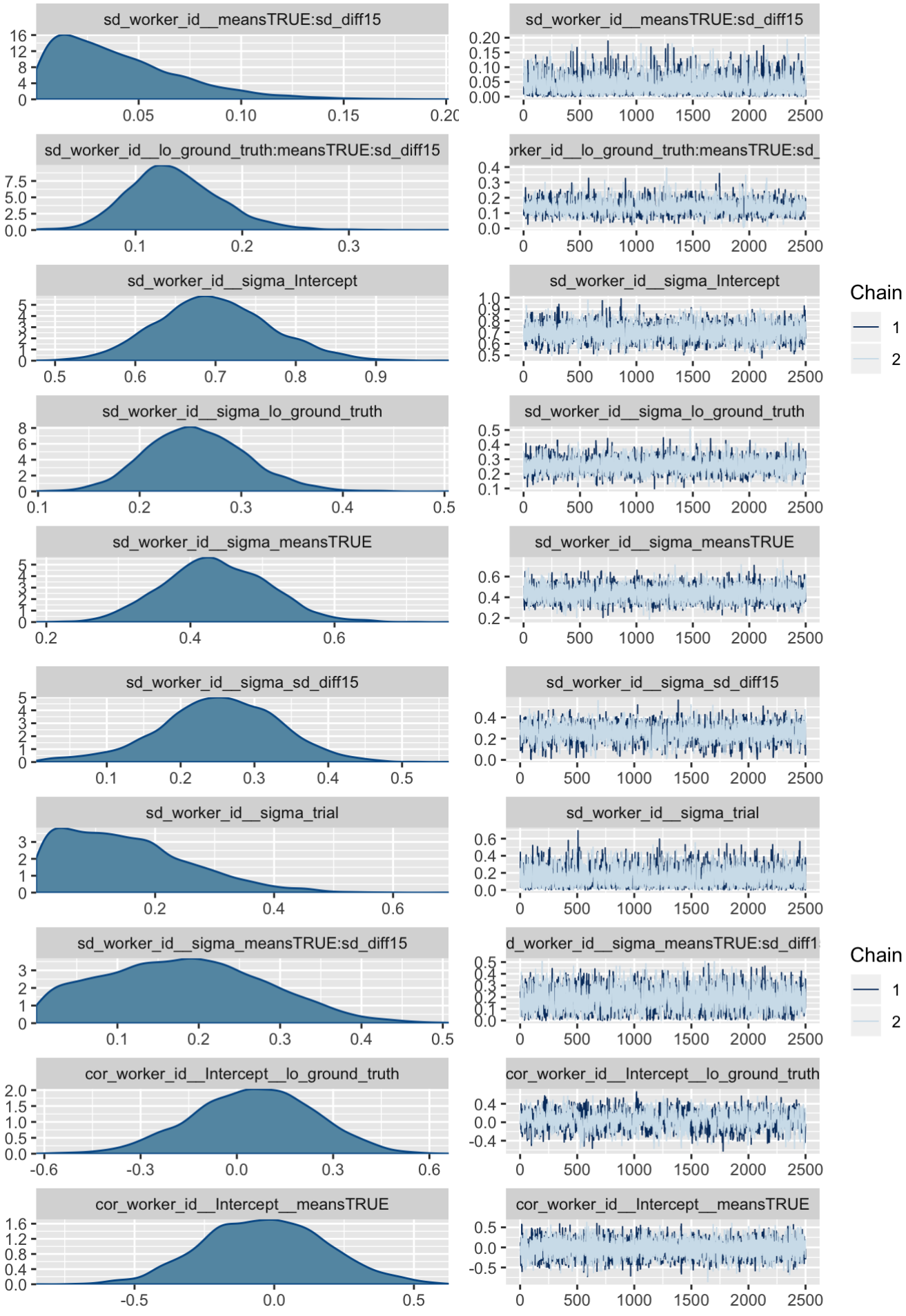
```
# hierarchical LLO model
m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.trial.sigma.gt.means.sd.trial <- brm(
  data = model_df, family = "gaussian",
  formula = bf(lo_p_sup ~  (1 + lo_ground_truth*means*sd_diff + trial|sharecor|worker_i
d) + lo_ground_truth*means*sd_diff*condition + trial,
               sigma ~ (1 + lo_ground_truth + means*sd_diff + trial|sharecor|worker_id)
 + lo_ground_truth + means*sd_diff + trial),
  prior = c(prior(normal(1, 0.5), class = b),
            prior(normal(1.3, 1), class = Intercept),
            prior(normal(0, 0.15), class = sd, group = worker_id),
            prior(normal(0, 0.3), class = b, dpar = sigma),
            prior(normal(0, 0.15), class = sd, dpar = sigma),
            prior(lkj(4), class = cor)),
  iter = 3000, warmup = 500, chains = 2, cores = 2,
  control = list(adapt_delta = 0.99, max_treedepth = 12),
  file = "model-fits/llo_mdl-wrkr_means_sd_vis-r_means_sd_trial_sigma_gt_means_sd_trial"
)
```

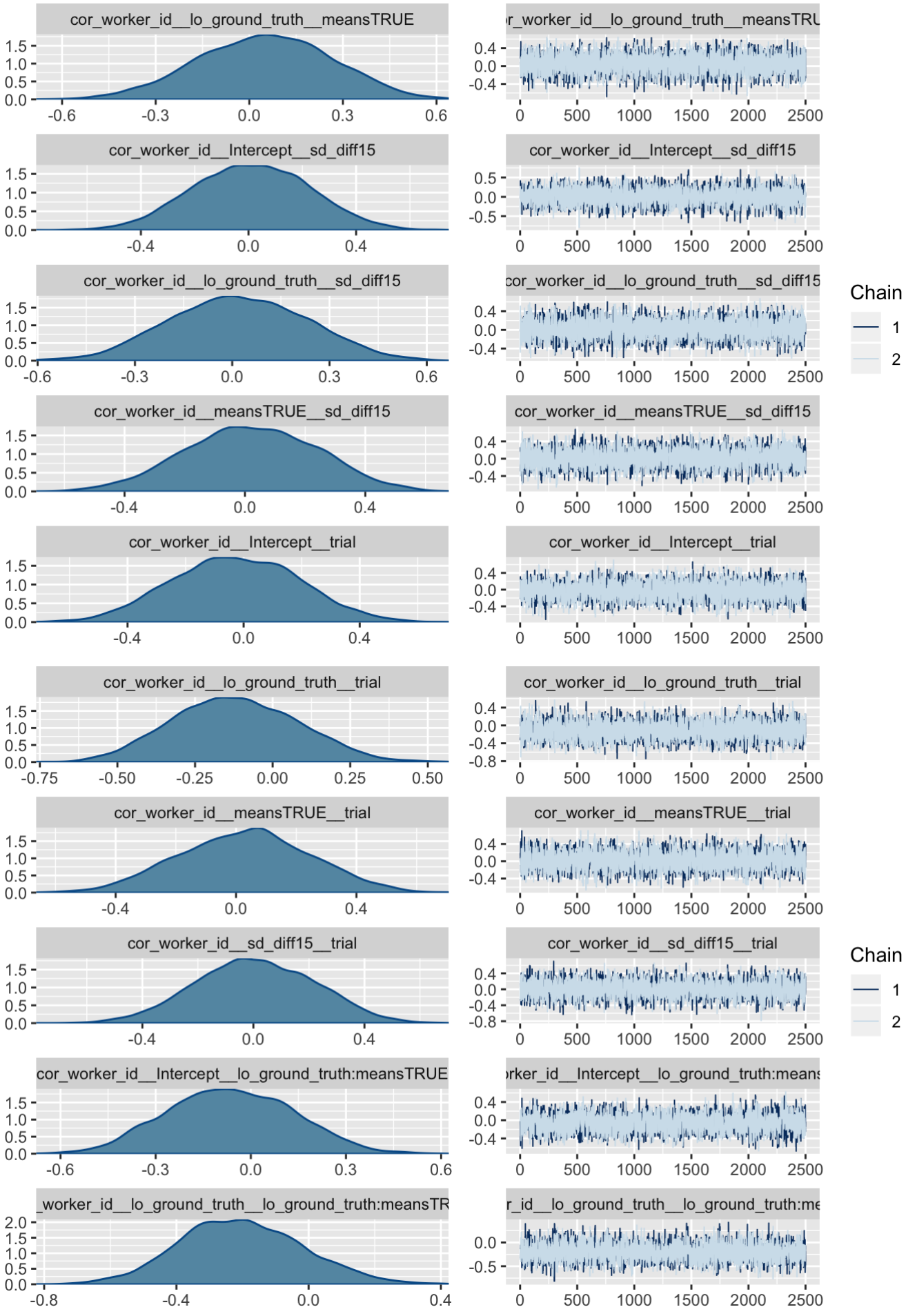
```
plot(m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.trial.sigma.gt.means.sd.trial)
```

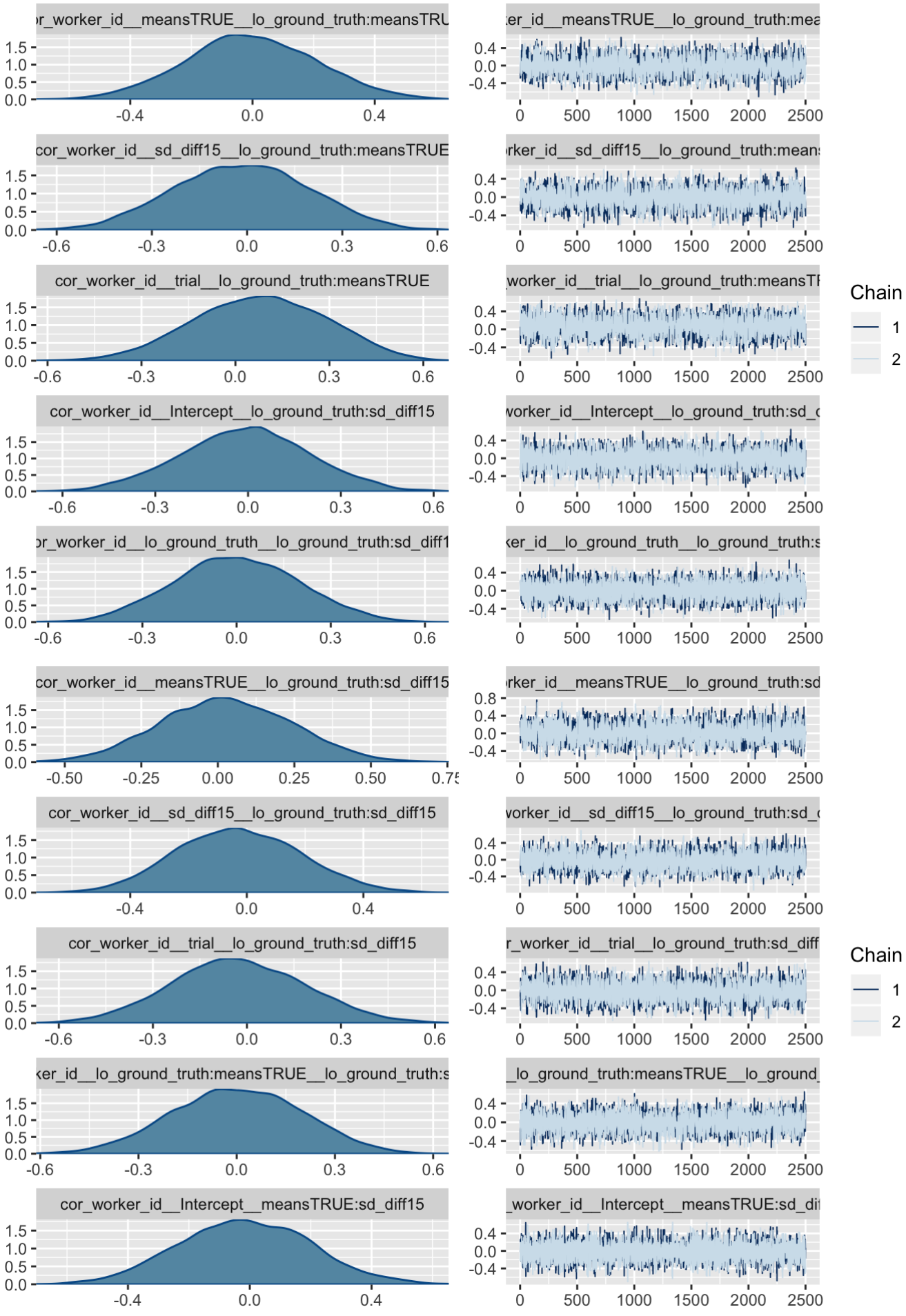

b_Intercept
b_sigma_Intercept
b_lo_ground_truth
b_meansTRUE
b_sd_diff15
b_conditionintervals
b_trial
b_lo_ground_truth:meansTRUE
b_lo_ground_truth:sd_diff15
b_meansTRUE:sd_diff15
b_lo_ground_truth:conditionintervals
Chain
1
2

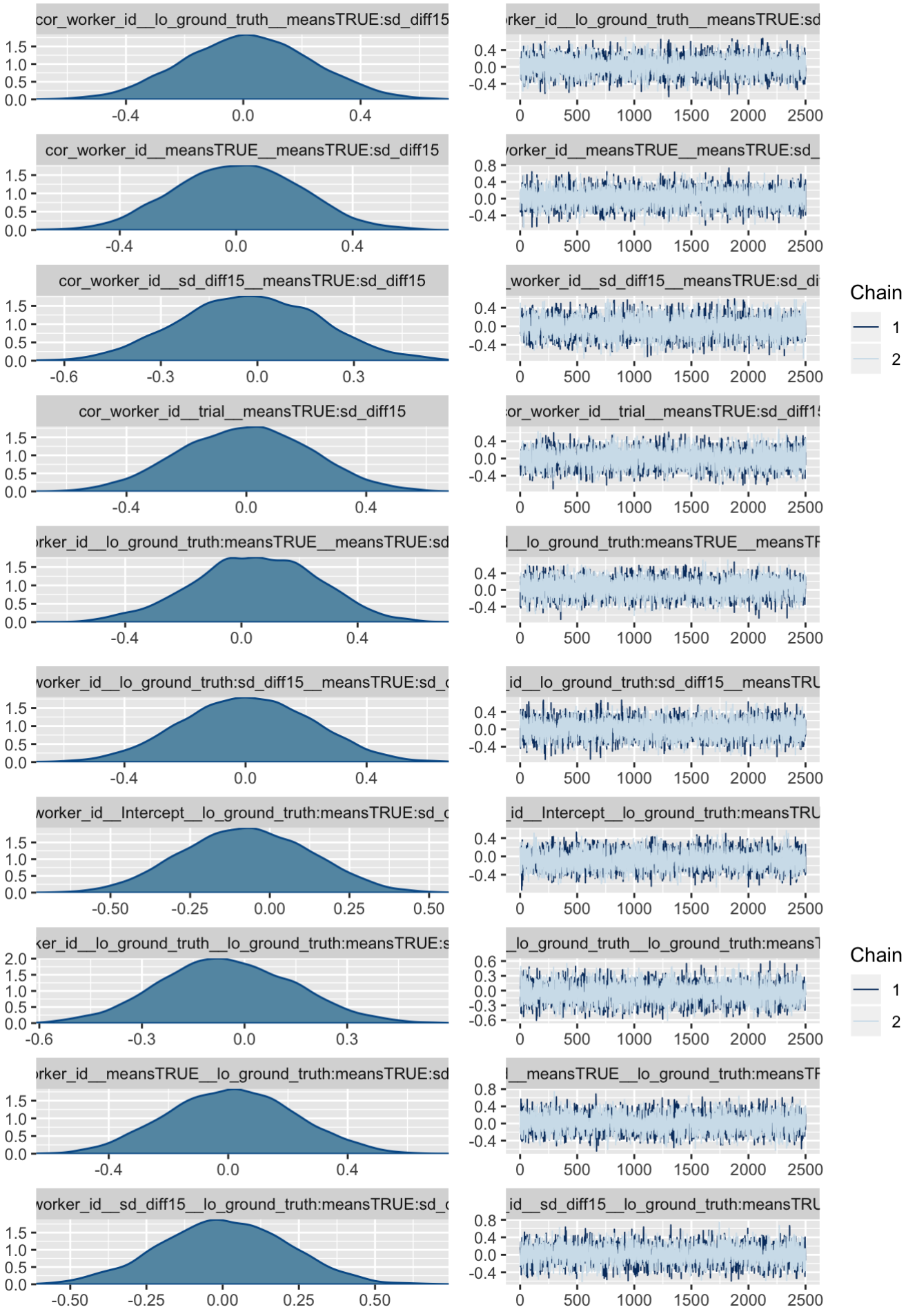

b_meansTRUE:conditionintervals
b_sd_diff15:conditionintervals
b_lo_ground_truth:meansTRUE:sd_diff15
b_lo_ground_truth:meansTRUE:conditionintervals
b_lo_ground_truth:sd_diff15:conditionintervals
b_meansTRUE:sd_diff15:conditionintervals
b_lo_ground_truth:meansTRUE:sd_diff15:conditionintervals
b_sigma_lo_ground_truth
b_sigma_meansTRUE
Chain
1
2

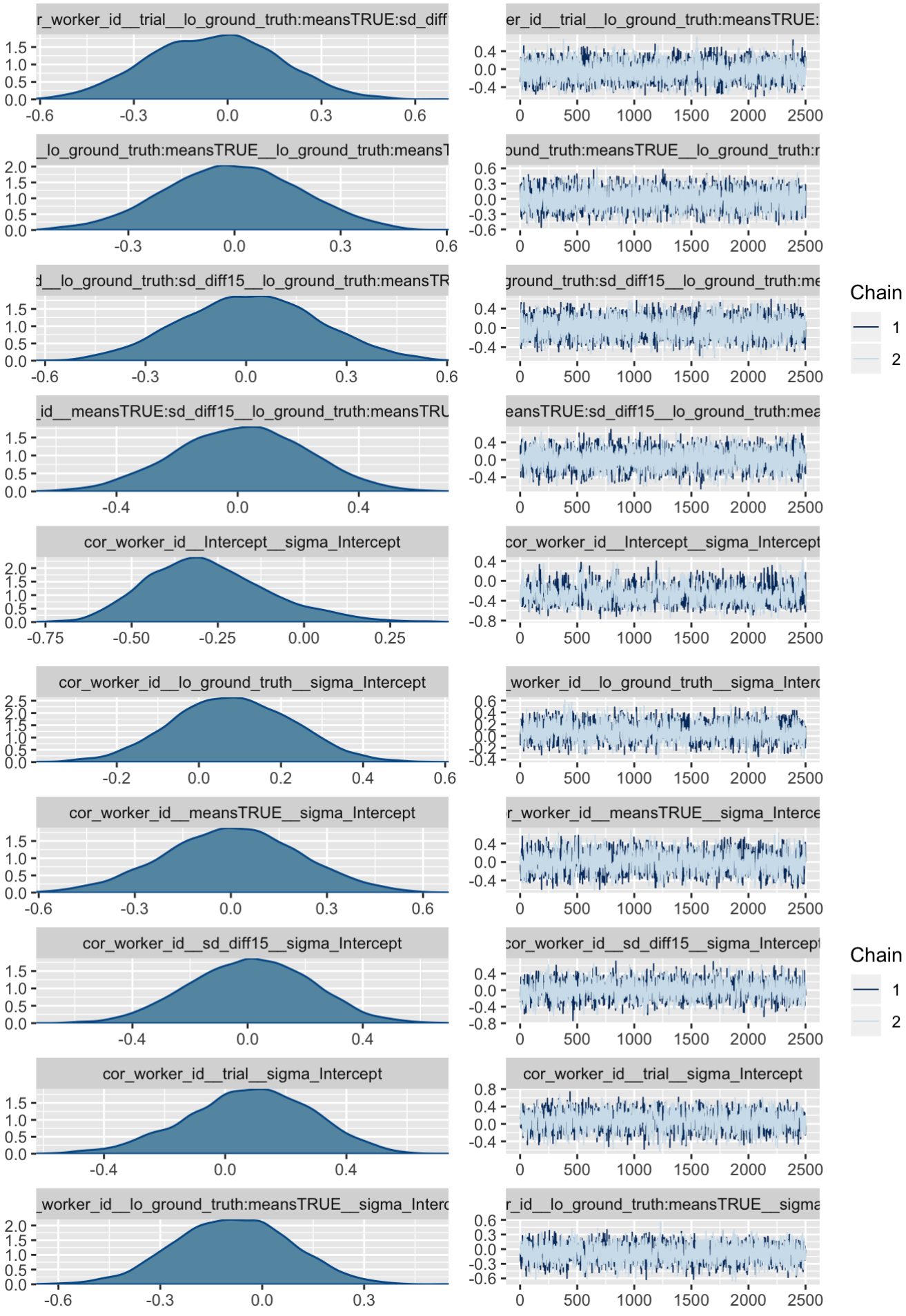

b_sigma_sd_diff15
b_sigma_trial
b_sigma_meansTRUE:sd_diff15
sd_worker_id__Intercept
sd_worker_id__lo_ground_truth
sd_worker_id__meansTRUE
sd_worker_id__sd_diff15
sd_worker_id__trial
sd_worker_id__lo_ground_truth:meansTRUE
sd_worker_id__lo_ground_truth:sd_diff15
Chain
1
2

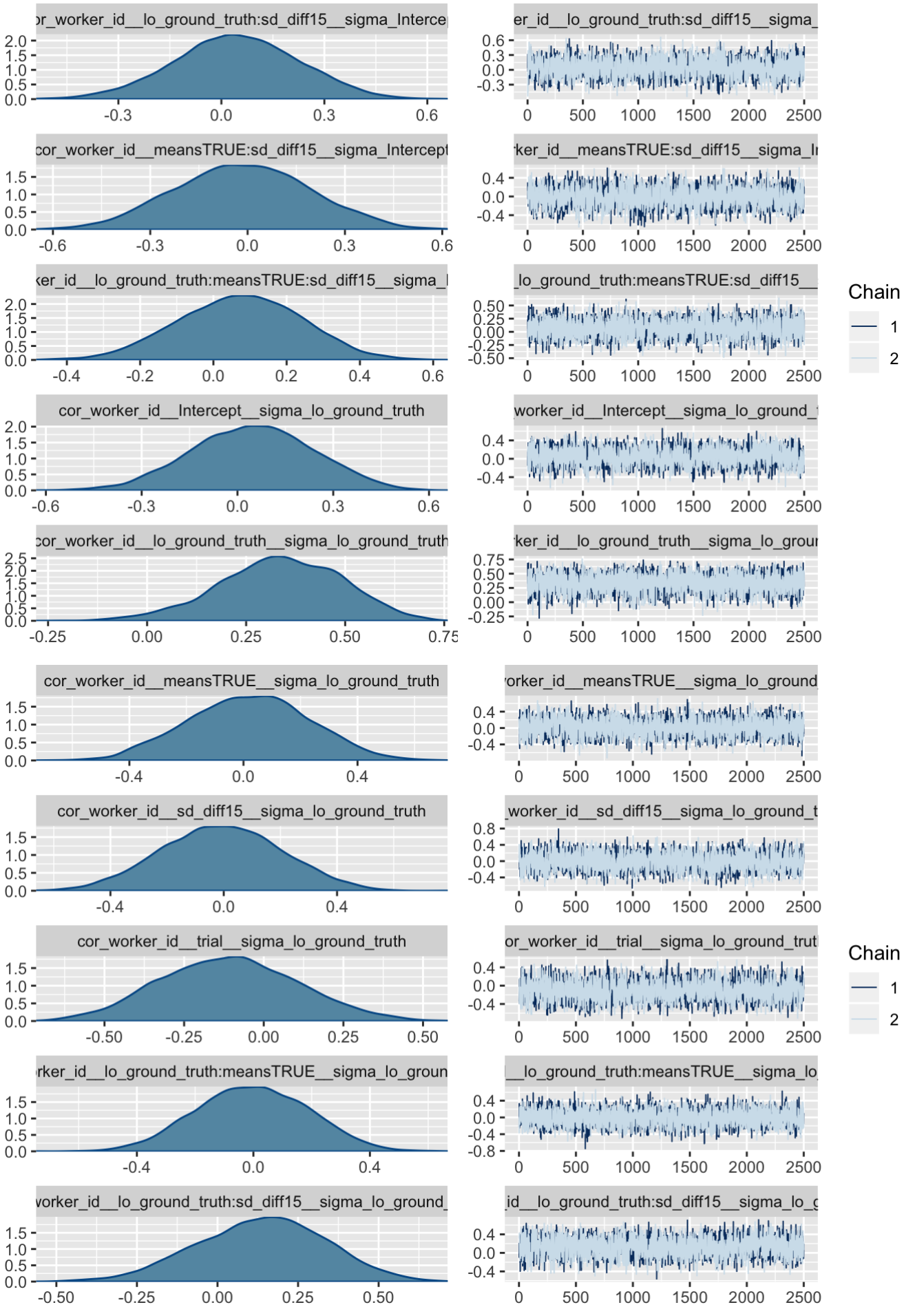

sd_worker_id__meansTRUE:sd_diff15
sd_worker_id__lo_ground_truth:meansTRUE:sd_diff15
sd_worker_id__sigma_Intercept
sd_worker_id__sigma_lo_ground_truth
sd_worker_id__sigma_meansTRUE
sd_worker_id__sigma_sd_diff15
sd_worker_id__sigma_trial
sd_worker_id__sigma_meansTRUE:sd_diff15
cor_worker_id__Intercept__lo_ground_truth
cor_worker_id__Intercept__meansTRUE
Chain
1
2

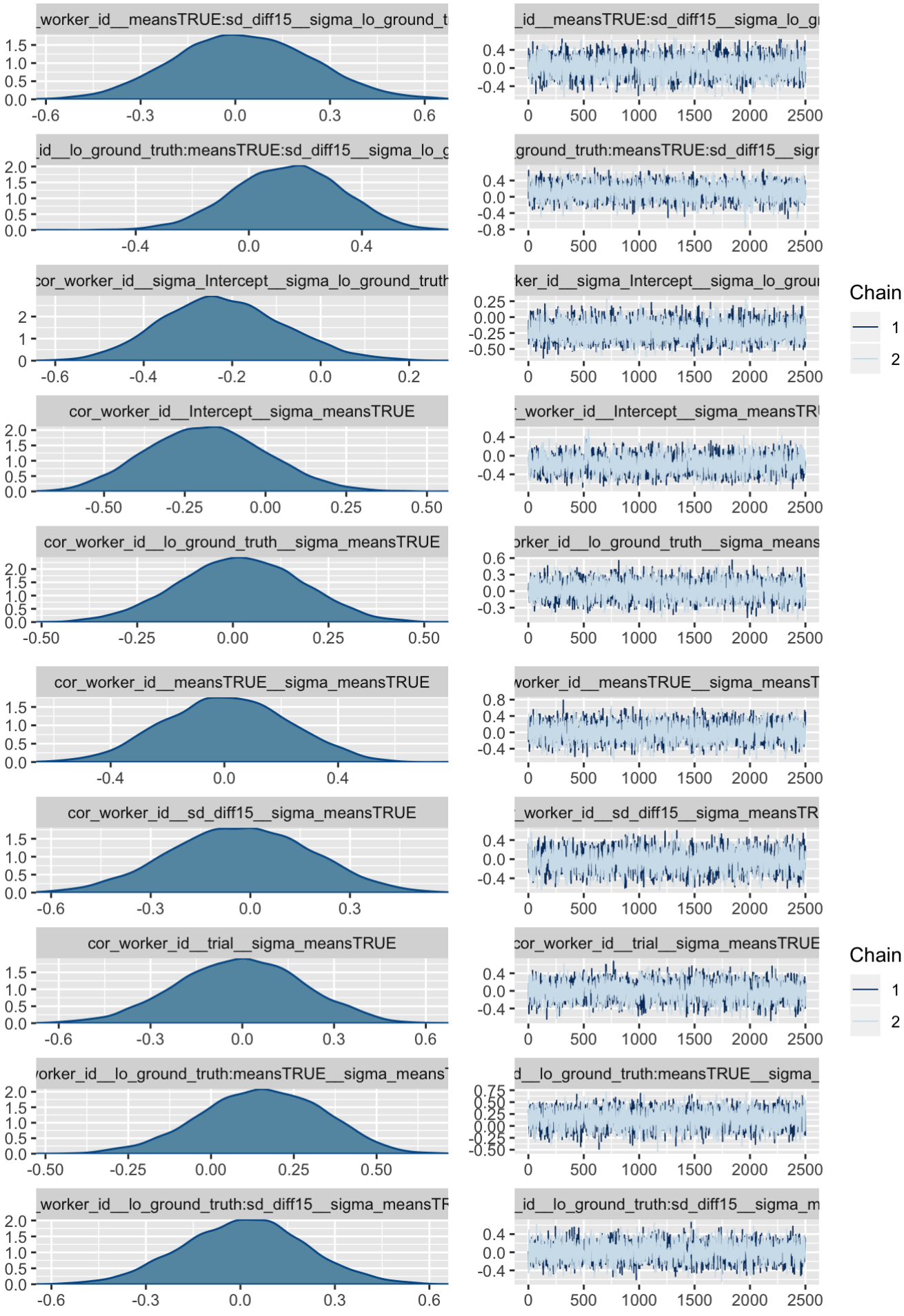

cor_worker_id__lo_ground_truth__meansTRUE
cor_worker_id__Intercept__sd_diff15
cor_worker_id__lo_ground_truth__sd_diff15
cor_worker_id__meansTRUE__sd_diff15
cor_worker_id__Intercept__trial
cor_worker_id__lo_ground_truth__trial
cor_worker_id__meansTRUE__trial
cor_worker_id__sd_diff15__trial
cor_worker_id__Intercept__lo_ground_truth:meansTRUE
_worker_id__lo_ground_truth__lo_ground_truth:meansTR
Chain
1
2

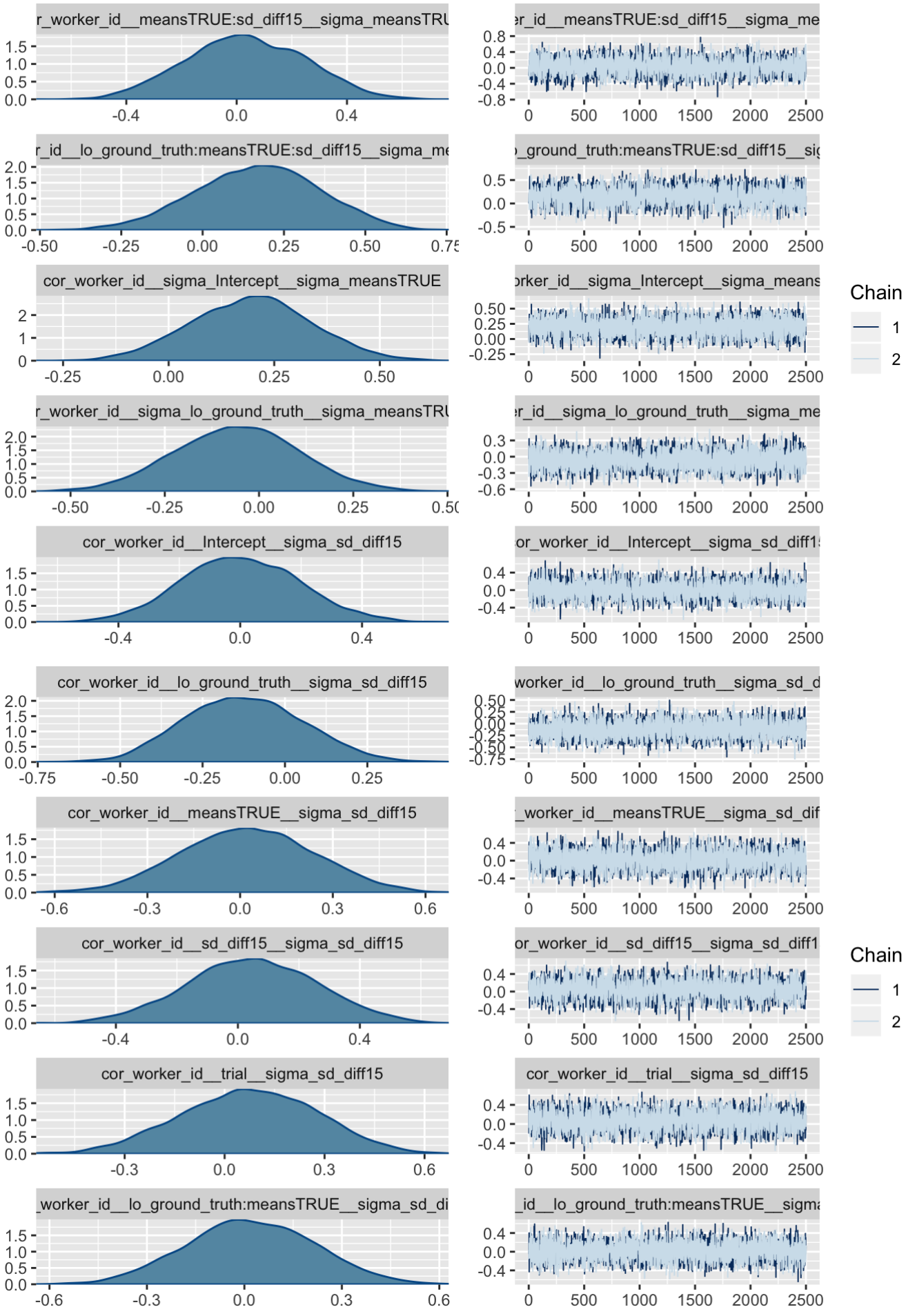

r_worker_id__meansTRUE__lo_ground_truth:meansTRU
er_id__meansTRUE__lo_ground_truth:mea
cor_worker_id__sd_diff15__lo_ground_truth:meansTRUE
rker_id__sd_diff15__lo_ground_truth:mean
cor_worker_id__trial__lo_ground_truth:meansTRUE
worker_id__trial__lo_ground_truth:meansTR
cor_worker_id__Intercept__lo_ground_truth:sd_diff15
worker_id__Intercept__lo_ground_truth:sd_
r_worker_id__lo_ground_truth__lo_ground_truth:sd_diff1
ker_id__lo_ground_truth__lo_ground_truth:s
cor_worker_id__meansTRUE__lo_ground_truth:sd_diff15
rker_id__meansTRUE__lo_ground_truth:sd
cor_worker_id__sd_diff15__lo_ground_truth:sd_diff15
vorker_id__sd_diff15__lo_ground_truth:sd_
cor_worker_id__trial__lo_ground_truth:sd_diff15
r_worker_id__trial__lo_ground_truth:sd_diff
ker_id__lo_ground_truth:meansTRUE__lo_ground_truth:s
_lo_ground_truth:meansTRUE__lo_ground_
cor_worker_id__Intercept__meansTRUE:sd_diff15
_worker_id__Intercept__meansTRUE:sd_di
Chain
1
2
Chain
1
2

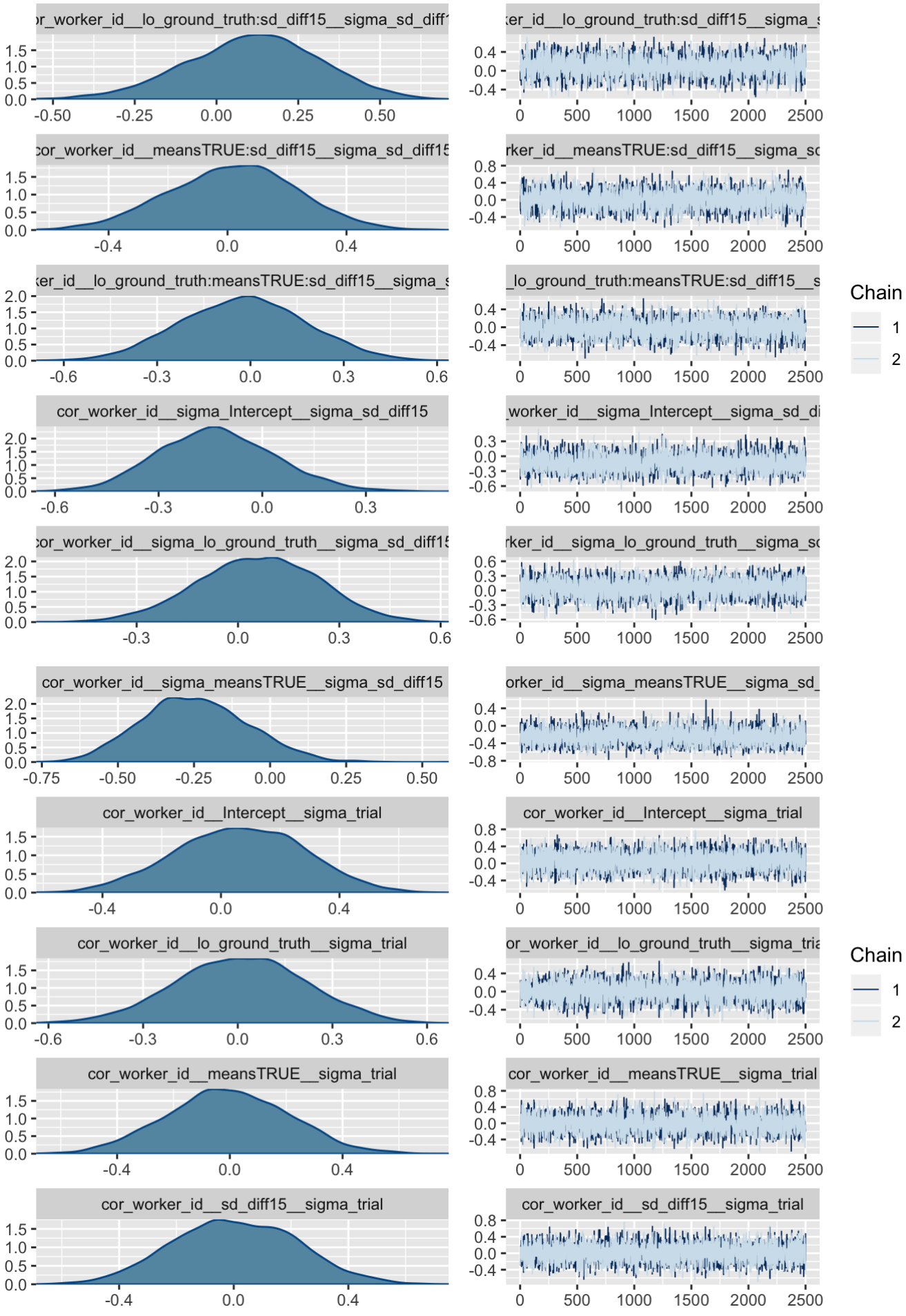

cor_worker_id__lo_ground_truth__meansTRUE:sd_diff15
cor_worker_id__meansTRUE__meansTRUE:sd_diff15
cor_worker_id__sd_diff15__meansTRUE:sd_diff15
cor_worker_id__trial__meansTRUE:sd_diff15
rker_id__lo_ground_truth:meansTRUE__meansTRUE:sd
worker_id__lo_ground_truth:sd_diff15__meansTRUE:sd_
worker_id__Intercept__lo_ground_truth:meansTRUE:sd_
ker_id__lo_ground_truth__lo_ground_truth:meansTRUE:s
rker_id__meansTRUE__lo_ground_truth:meansTRUE:sd
worker_id__sd_diff15__lo_ground_truth:meansTRUE:sd_
rker_id__lo_ground_truth__meansTRUE:sd
orker_id__meansTRUE__meansTRUE:sd_
worker_id__sd_diff15__meansTRUE:sd_di
or_worker_id__trial__meansTRUE:sd_diff1
__lo_ground_truth:meansTRUE__meansTR
id__lo_ground_truth:sd_diff15__meansTRU
id__Intercept__lo_ground_truth:meansTRU
_lo_ground_truth__lo_ground_truth:meansT
__meansTRUE__lo_ground_truth:meansTR
id__sd_diff15__lo_ground_truth:meansTRU
Chain
1
2

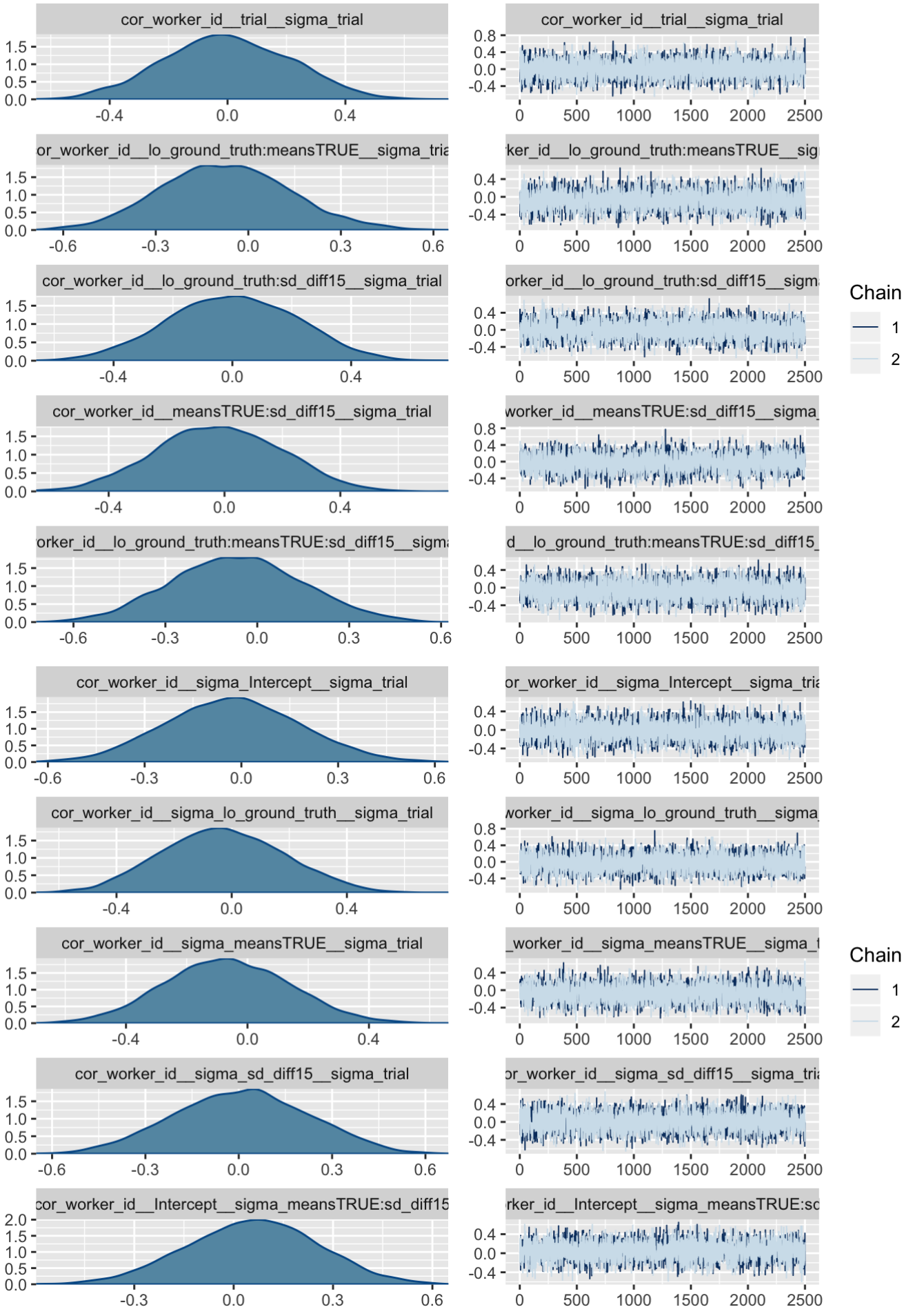

r_worker_id__trial__lo_ground_truth:meansTRUE:sd_diff
er_id__trial__lo_ground_truth:meansTRUE:
_lo_ground_truth:meansTRUE__lo_ground_truth:meansT
ound_truth:meansTRUE__lo_ground_truth:r
d__lo_ground_truth:sd_diff15__lo_ground_truth:meansTR
ground_truth:sd_diff15__lo_ground_truth:me
_id__meansTRUE:sd_diff15__lo_ground_truth:meansTRU
eansTRUE:sd_diff15__lo_ground_truth:mea
cor_worker_id__Intercept__sigma_Intercept
cor_worker_id__Intercept__sigma_Intercept
cor_worker_id__lo_ground_truth__sigma_Intercept
_worker_id__lo_ground_truth__sigma_Interc
cor_worker_id__meansTRUE__sigma_Intercept
r_worker_id__meansTRUE__sigma_Interce
cor_worker_id__sd_diff15__sigma_Intercept
cor_worker_id__sd_diff15__sigma_Intercept
cor_worker_id__trial__sigma_Intercept
cor_worker_id__trial__sigma_Intercept
_worker_id__lo_ground_truth:meansTRUE__sigma_Interc
r_id__lo_ground_truth:meansTRUE__sigma
Chain
1
2
Chain
1
2

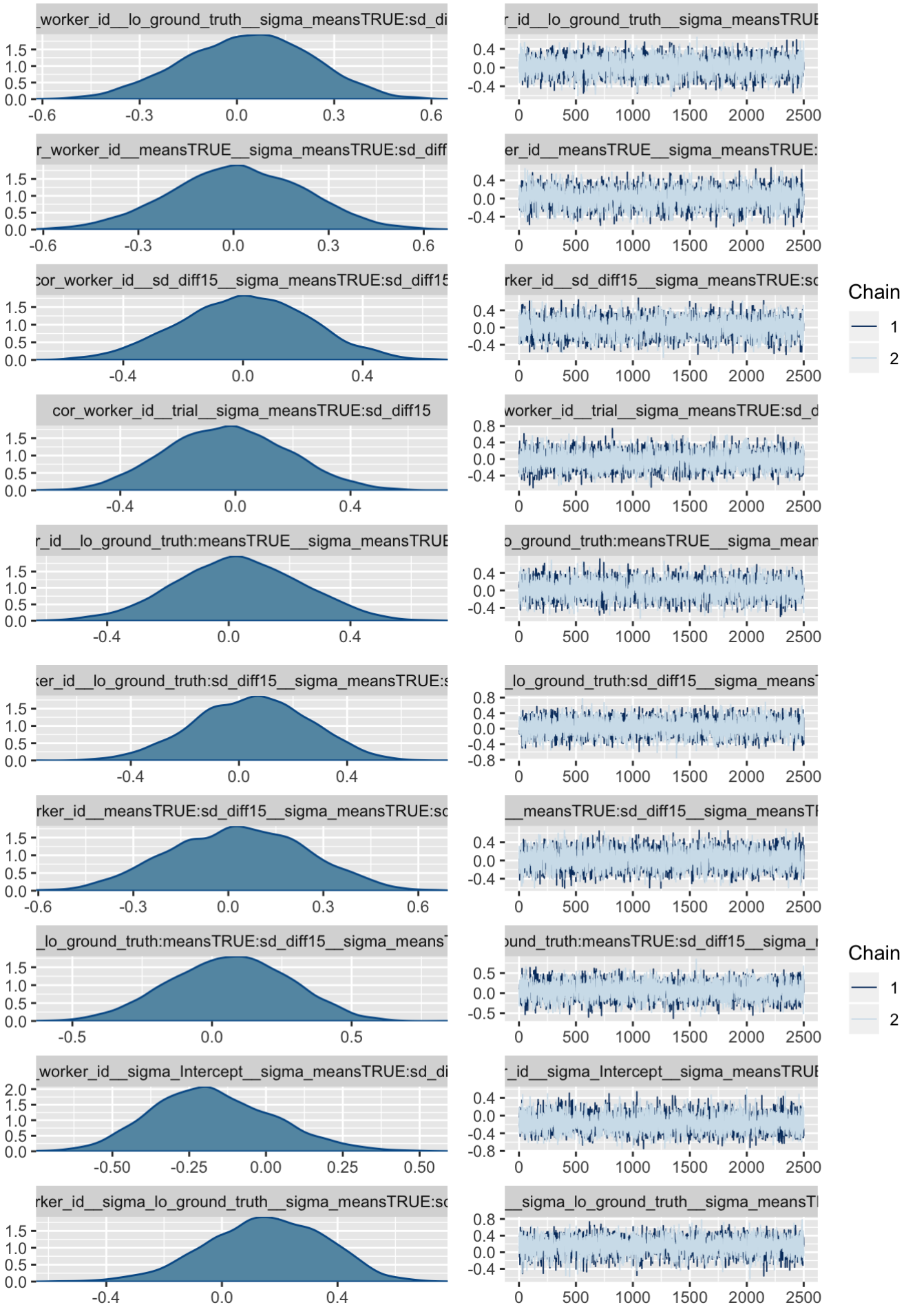

cor_worker_id__lo_ground_truth:sd_diff15__sigma_Intercept
cor_worker_id__meansTRUE:sd_diff15__sigma_Intercept
cor_worker_id__lo_ground_truth:meansTRUE:sd_diff15__sigma_Intercept
cor_worker_id__Intercept__sigma_lo_ground_truth
cor_worker_id__lo_ground_truth__sigma_lo_ground_truth
cor_worker_id__meansTRUE__sigma_lo_ground_truth
cor_worker_id__sd_diff15__sigma_lo_ground_truth
cor_worker_id__trial__sigma_lo_ground_truth
cor_worker_id__lo_ground_truth:meansTRUE__sigma_lo_ground_truth
cor_worker_id__lo_ground_truth:sd_diff15__sigma_lo_ground_truth
Chain
1
2

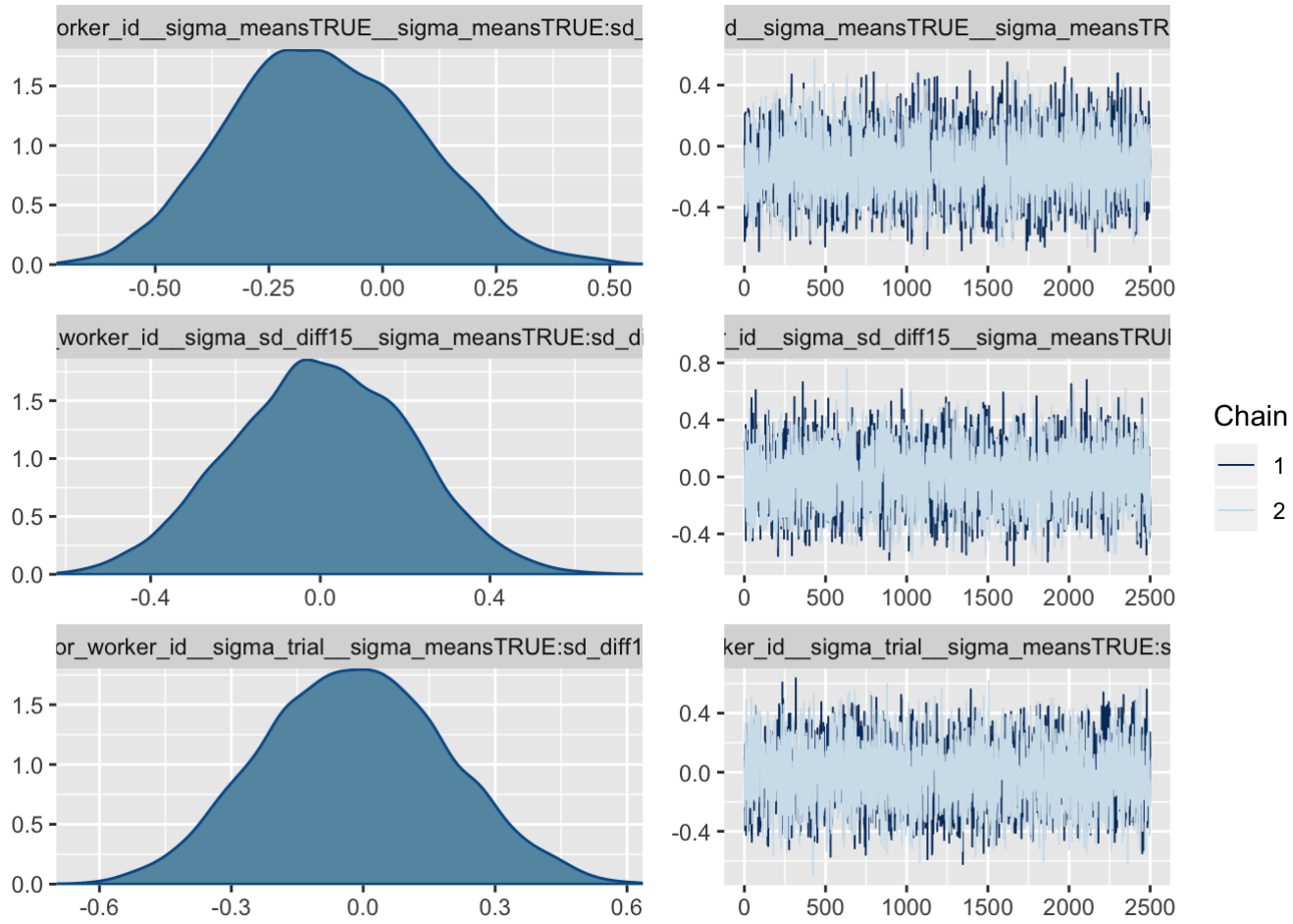

_worker_id__meansTRUE:sd_diff15__sigma_lo_ground_t
_id__meansTRUE:sd_diff15__sigma_lo_gr
_id__lo_ground_truth:meansTRUE:sd_diff15__sigma_lo_g
ground_truth:meansTRUE:sd_diff15__sigr
cor_worker_id__sigma_Intercept__sigma_lo_ground_truth
ker_id__sigma_Intercept__sigma_lo_grou
cor_worker_id__Intercept__sigma_meansTRUE
_worker_id__Intercept__sigma_meansTR
cor_worker_id__lo_ground_truth__sigma_meansTRUE
orker_id__lo_ground_truth__sigma_means
cor_worker_id__meansTRUE__sigma_meansTRUE
worker_id__meansTRUE__sigma_meansT
cor_worker_id__sd_diff15__sigma_meansTRUE
_worker_id__sd_diff15__sigma_meansTR
cor_worker_id__trial__sigma_meansTRUE
cor_worker_id__trial__sigma_meansTRUE
orker_id__lo_ground_truth:meansTRUE__sigma_means
__lo_ground_truth:meansTRUE__sigma_
_worker_id__lo_ground_truth:sd_diff15__sigma_meansTR
_id__lo_ground_truth:sd_diff15__sigma_m
Chain
1
2
Chain
1
2


r_worker_id__meansTRUE:sd_diff15__sigma_meansTRU
r_id__lo_ground_truth:meansTRUE:sd_diff15__sigma_me
cor_worker_id__sigma_Intercept__sigma_meansTRUE
r_worker_id__sigma_lo_ground_truth__sigma_meansTRU
cor_worker_id__Intercept__sigma_sd_diff15
cor_worker_id__lo_ground_truth__sigma_sd_diff15
cor_worker_id__meansTRUE__sigma_sd_diff15
cor_worker_id__sd_diff15__sigma_sd_diff15
cor_worker_id__trial__sigma_sd_diff15
worker_id__lo_ground_truth:meansTRUE__sigma_sd_di
Chain
1
2


cor_worker_id__lo_ground_truth:sd_diff15__sigma_sd_diff
cor_worker_id__meansTRUE:sd_diff15__sigma_sd_diff15
cor_worker_id__lo_ground_truth:meansTRUE:sd_diff15__sigma_s
cor_worker_id__sigma_Intercept__sigma_sd_diff15
cor_worker_id__sigma_lo_ground_truth__sigma_sd_diff15
cor_worker_id__sigma_meansTRUE__sigma_sd_diff15
cor_worker_id__Intercept__sigma_trial
cor_worker_id__lo_ground_truth__sigma_trial
cor_worker_id__meansTRUE__sigma_trial
cor_worker_id__sd_diff15__sigma_trial
Chain
1
2


cor_worker_id__trial__sigma_trial
cor_worker_id__lo_ground_truth:meansTRUE__sigma_trial
cor_worker_id__lo_ground_truth:sd_diff15__sigma_trial
cor_worker_id__meansTRUE:sd_diff15__sigma_trial
cor_worker_id__lo_ground_truth:meansTRUE:sd_diff15__sigma_trial
cor_worker_id__sigma_Intercept__sigma_trial
cor_worker_id__sigma_lo_ground_truth__sigma_trial
cor_worker_id__sigma_meansTRUE__sigma_trial
cor_worker_id__sigma_sd_diff15__sigma_trial
cor_worker_id__Intercept__sigma_meansTRUE:sd_diff15
Chain
1
2


_worker_id__lo_ground_truth__sigma_meansTRUE:sd_di
r_id__lo_ground_truth__sigma_meansTRUE
r_worker_id__meansTRUE__sigma_meansTRUE:sd_diff
er_id__meansTRUE__sigma_meansTRUE:
cor_worker_id__sd_diff15__sigma_meansTRUE:sd_diff15
rker_id__sd_diff15__sigma_meansTRUE:sd
cor_worker_id__trial__sigma_meansTRUE:sd_diff15
worker_id__trial__sigma_meansTRUE:sd_d
r_id__lo_ground_truth:meansTRUE__sigma_meansTRUE
o_ground_truth:meansTRUE__sigma_mean
ker_id__lo_ground_truth:sd_diff15__sigma_meansTRUE:s
_lo_ground_truth:sd_diff15__sigma_means
rker_id__meansTRUE:sd_diff15__sigma_meansTRUE:sd
__meansTRUE:sd_diff15__sigma_meansTR
_lo_ground_truth:meansTRUE:sd_diff15__sigma_means
ound_truth:meansTRUE:sd_diff15__sigma_
_worker_id__sigma_Intercept__sigma_meansTRUE:sd_di
_id__sigma_Intercept__sigma_meansTRUE
rker_id__sigma_lo_ground_truth__sigma_meansTRUE:sd
__sigma_lo_ground_truth__sigma_meansTR
Chain
1
2
Chain
1
2



Model Comparison

Each time we add a random effect, the number of parameters multiplies, especially since all random effects share a covariance matrix. We want to make sure these parameters are contributing to the predictive validity of the model more than they risk overfitting. We'll evaluate this by using WAIC to compare models. Whichever model has the smallest value of WAIC is the one that has the best predictive validity for the fewest parameters.

```
waic(
  m.wrkr.means.sd.vis.llo_p_sup,
  m.wrkr.means.sd.vis.llo_p_sup.r.means.sd,
  m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.sigma.gt,
  m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.sigma.gt.means.sd,
  m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.trial.sigma.gt.means.sd,
  m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.trial.sigma.gt.means.sd.trial)
```

```
##
WAIC
## m.wrkr.means.sd.vis.llo_p_sup
840.79
## m.wrkr.means.sd.vis.llo_p_sup.r.means.sd
686.32
## m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.sigma.gt
605.00
## m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.sigma.gt.means.sd
502.64
## m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.trial.sigma.gt.means.sd
498.00
## m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.trial.sigma.gt.means.sd.trial
489.70
## m.wrkr.means.sd.vis.llo_p_sup - m.wrkr.means.sd.vis.llo_p_sup.r.means.sd
154.47
## m.wrkr.means.sd.vis.llo_p_sup - m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.sigma.gt
235.79
## m.wrkr.means.sd.vis.llo_p_sup - m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.sigma.gt.mea
ns.sd                                                 338.14
## m.wrkr.means.sd.vis.llo_p_sup - m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.trial.sigma.
gt.means.sd                                           342.78
## m.wrkr.means.sd.vis.llo_p_sup - m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.trial.sigma.
gt.means.sd.trial                                     351.09
## m.wrkr.means.sd.vis.llo_p_sup.r.means.sd - m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.s
igma.gt                                                81.32
## m.wrkr.means.sd.vis.llo_p_sup.r.means.sd - m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.s
igma.gt.means.sd                                      183.68
## m.wrkr.means.sd.vis.llo_p_sup.r.means.sd - m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.t
rial.sigma.gt.means.sd                                188.32
## m.wrkr.means.sd.vis.llo_p_sup.r.means.sd - m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.t
rial.sigma.gt.means.sd.trial                          196.62
## m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.sigma.gt - m.wrkr.means.sd.vis.llo_p_sup.r.m
eans.sd.sigma.gt.means.sd                             102.36
## m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.sigma.gt - m.wrkr.means.sd.vis.llo_p_sup.r.m
eans.sd.trial.sigma.gt.means.sd                       107.00
## m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.sigma.gt - m.wrkr.means.sd.vis.llo_p_sup.r.m
eans.sd.trial.sigma.gt.means.sd.trial                 115.30
## m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.sigma.gt.means.sd - m.wrkr.means.sd.vis.llo_
p_sup.r.means.sd.trial.sigma.gt.means.sd               4.64
## m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.sigma.gt.means.sd - m.wrkr.means.sd.vis.llo_
p_sup.r.means.sd.trial.sigma.gt.means.sd.trial        12.95
## m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.trial.sigma.gt.means.sd - m.wrkr.means.sd.vi
s.llo_p_sup.r.means.sd.trial.sigma.gt.means.sd.trial   8.31
##
SE
## m.wrkr.means.sd.vis.llo_p_sup
81.55
## m.wrkr.means.sd.vis.llo_p_sup.r.means.sd
83.32
## m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.sigma.gt
80.77
## m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.sigma.gt.means.sd
```

```
77.64
## m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.trial.sigma.gt.means.sd
77.53
## m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.trial.sigma.gt.means.sd.trial
77.50
## m.wrkr.means.sd.vis.llo_p_sup - m.wrkr.means.sd.vis.llo_p_sup.r.means.sd
25.03
## m.wrkr.means.sd.vis.llo_p_sup - m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.sigma.gt
35.92
## m.wrkr.means.sd.vis.llo_p_sup - m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.sigma.gt.mea
ns.sd                                          42.39
## m.wrkr.means.sd.vis.llo_p_sup - m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.trial.sigma.
gt.means.sd                                    42.87
## m.wrkr.means.sd.vis.llo_p_sup - m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.trial.sigma.
gt.means.sd.trial                              43.71
## m.wrkr.means.sd.vis.llo_p_sup.r.means.sd - m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.s
igma.gt                                        28.00
## m.wrkr.means.sd.vis.llo_p_sup.r.means.sd - m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.s
igma.gt.means.sd                               35.47
## m.wrkr.means.sd.vis.llo_p_sup.r.means.sd - m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.t
rial.sigma.gt.means.sd                         35.95
## m.wrkr.means.sd.vis.llo_p_sup.r.means.sd - m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.t
rial.sigma.gt.means.sd.trial                   36.96
## m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.sigma.gt - m.wrkr.means.sd.vis.llo_p_sup.r.m
eans.sd.sigma.gt.means.sd                      22.12
## m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.sigma.gt - m.wrkr.means.sd.vis.llo_p_sup.r.m
eans.sd.trial.sigma.gt.means.sd                22.50
## m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.sigma.gt - m.wrkr.means.sd.vis.llo_p_sup.r.m
eans.sd.trial.sigma.gt.means.sd.trial          23.87
## m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.sigma.gt.means.sd - m.wrkr.means.sd.vis.llo_
p_sup.r.means.sd.trial.sigma.gt.means.sd        2.85
## m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.sigma.gt.means.sd - m.wrkr.means.sd.vis.llo_
p_sup.r.means.sd.trial.sigma.gt.means.sd.trial  5.14
## m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.trial.sigma.gt.means.sd - m.wrkr.means.sd.vi
s.llo_p_sup.r.means.sd.trial.sigma.gt.means.sd.trial  3.92
```

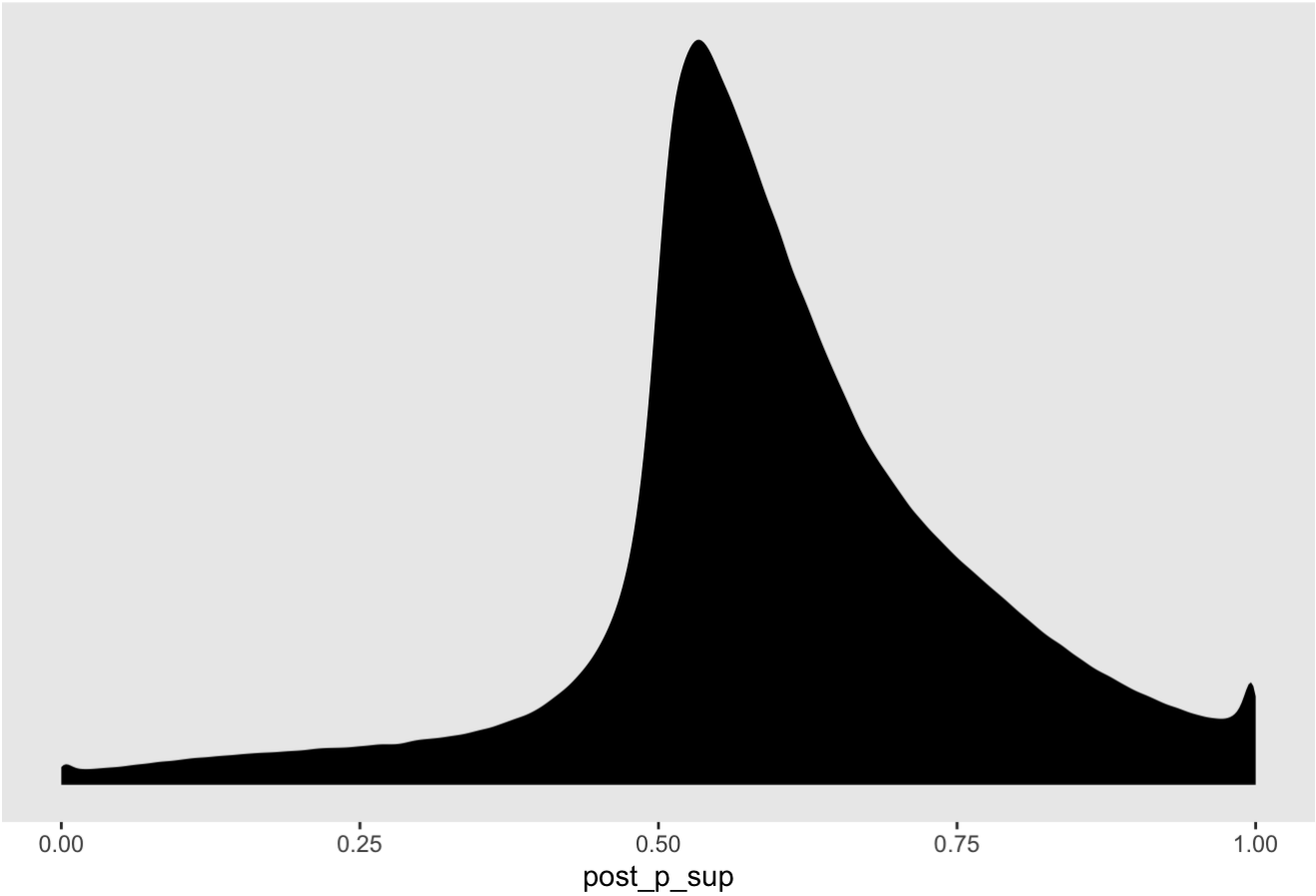
The maximal model has the best WAIC value.

Predictive Checks

Let’s check our posterior predictive distribution.

```
# posterior predictive check
model_df %>%
  select(lo_ground_truth, worker_id, means, sd_diff, condition, trial) %>%
  add_predicted_draws(m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.trial.sigma.gt.means.sd.trial, prediction = "lo_p_sup", seed = 1234) %>%
  mutate(
    # transform to probability units
    post_p_sup = plogis(lo_p_sup)
  ) %>%
  ggplot(aes(x = post_p_sup)) +
  geom_density(fill = "black", size = 0) +
  scale_y_continuous(NULL, breaks = NULL) +
  labs(subtitle = "Posterior predictive distribution for probability of superiority") +
  theme(panel.grid = element_blank())
```

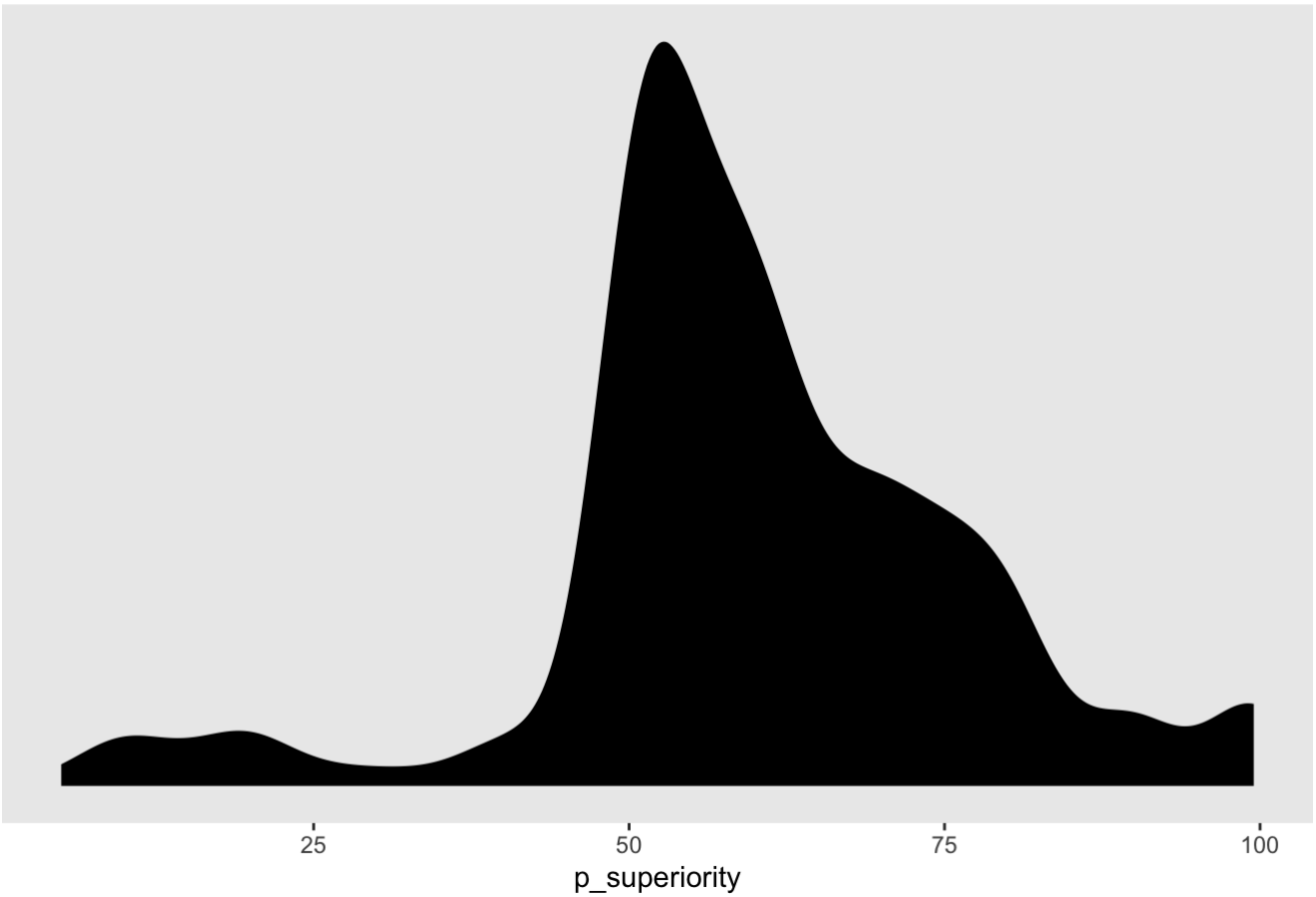
Posterior predictive distribution for probability of superiority



How do these predictions compare to the observed data?

```
# data density
model_df %>%
  ggplot(aes(x = p_superiority)) +
  geom_density(fill = "black", size = 0) +
  scale_y_continuous(NULL, breaks = NULL) +
  labs(subtitle = "Data distribution for probability of superiority") +
  theme(panel.grid = element_blank())
```

Data distribution for probability of superiority



Running a leave one out posterior predictive check, we can see that overall this model has decent predictive validity.

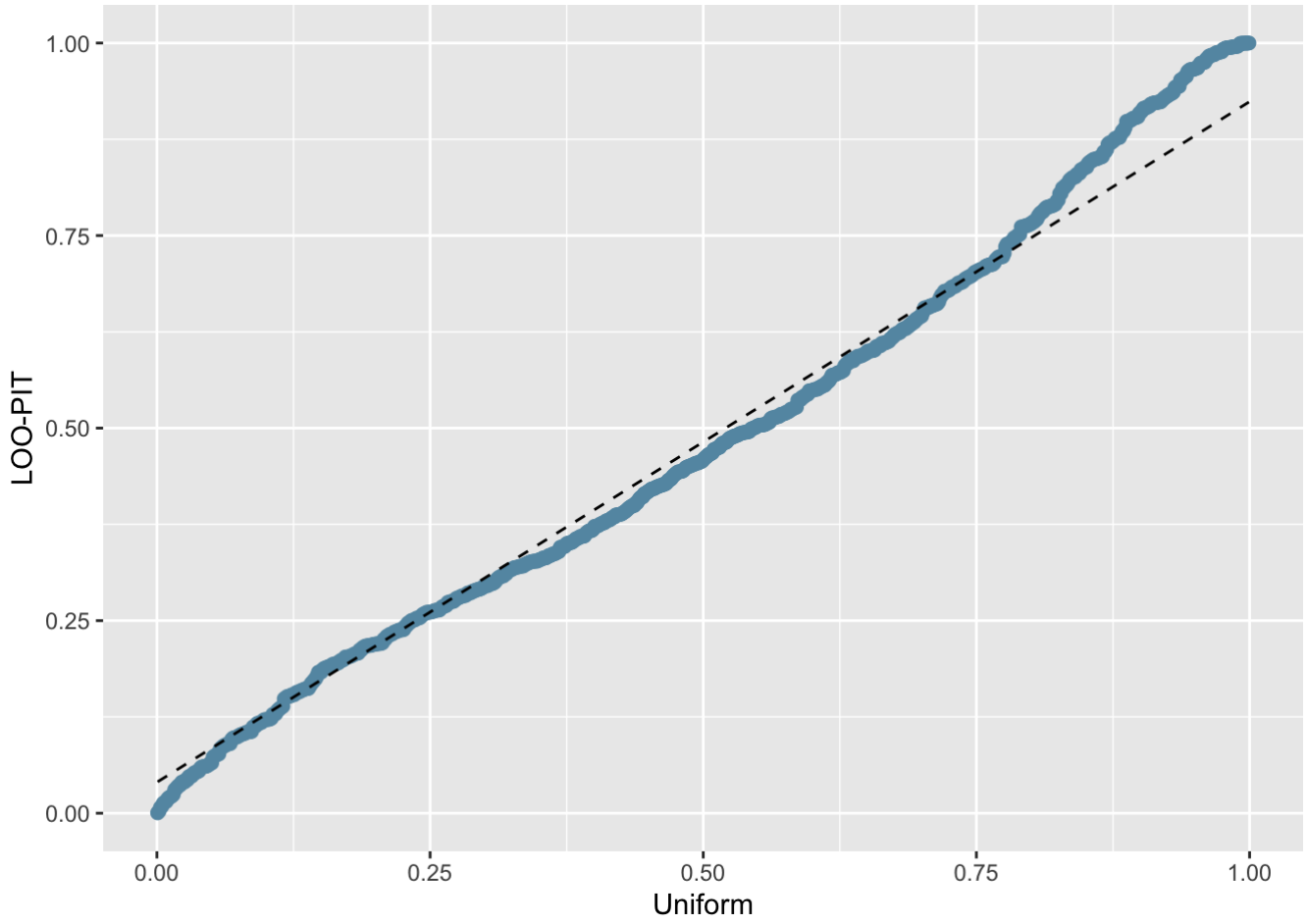
```
# set up data for LOO posterior predictive check
y <- model_df$lo_p_sup
yrep <- posterior_predict(m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.trial.sigma.gt.means.
sd.trial)

# run LOO to get weights
loo <- loo(m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.trial.sigma.gt.means.sd.trial, save_
psis = TRUE, cores = 2)
```

```
## Warning: Found 35 observations with a pareto_k > 0.7 in model
## 'm.wrkr.means.sd.vis.llo_p_sup.r.means.sd.trial.sigma.gt.means.sd.trial'.
## With this many problematic observations, it may be more appropriate to use
## 'kfold' with argument 'K = 10' to perform 10-fold cross-validation rather
## than LOO.
```

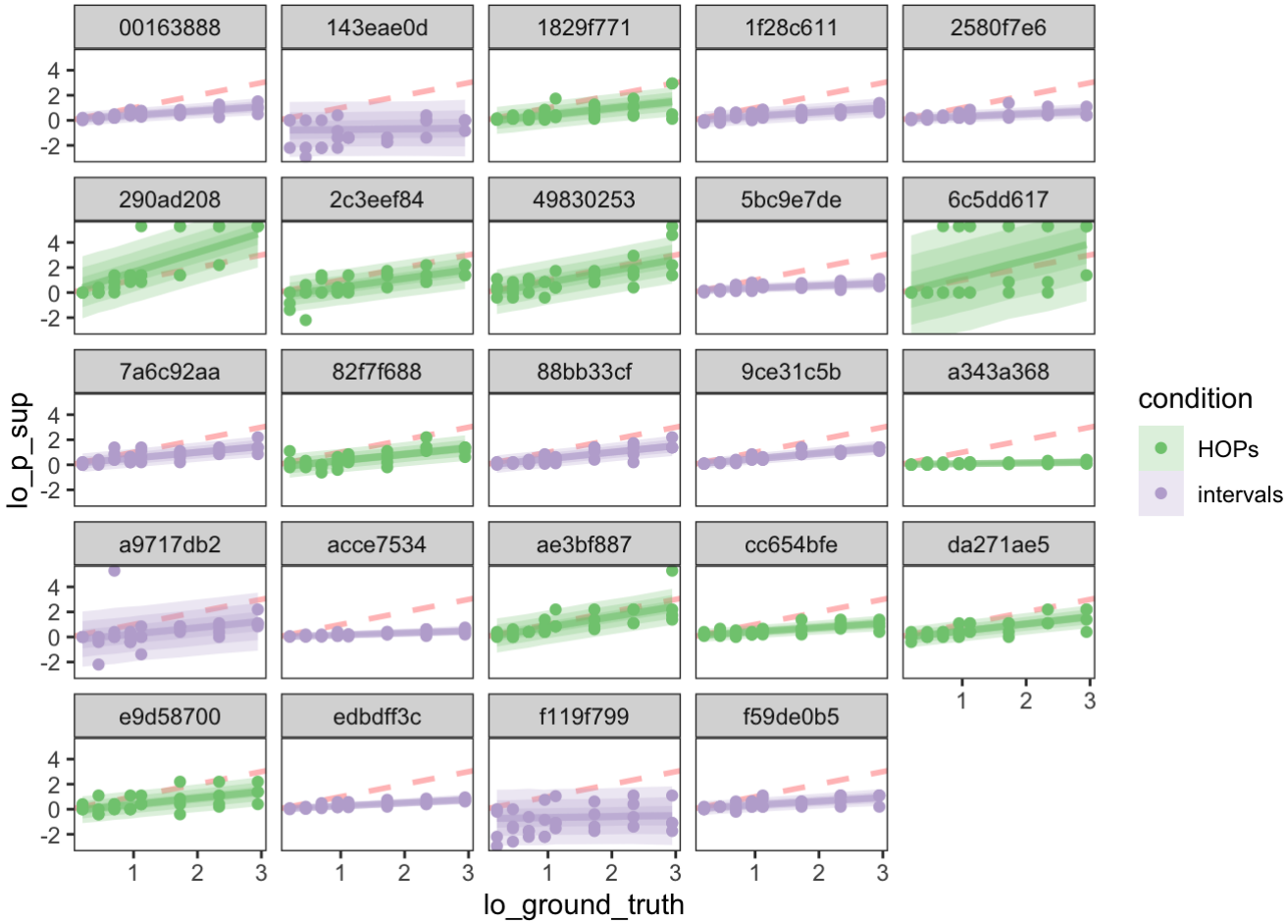
```
psis <- loo$psis_object
lw <- weights(psis)
```

```
ppc_loo_pit_qq(y, yrep, lw = lw)
```



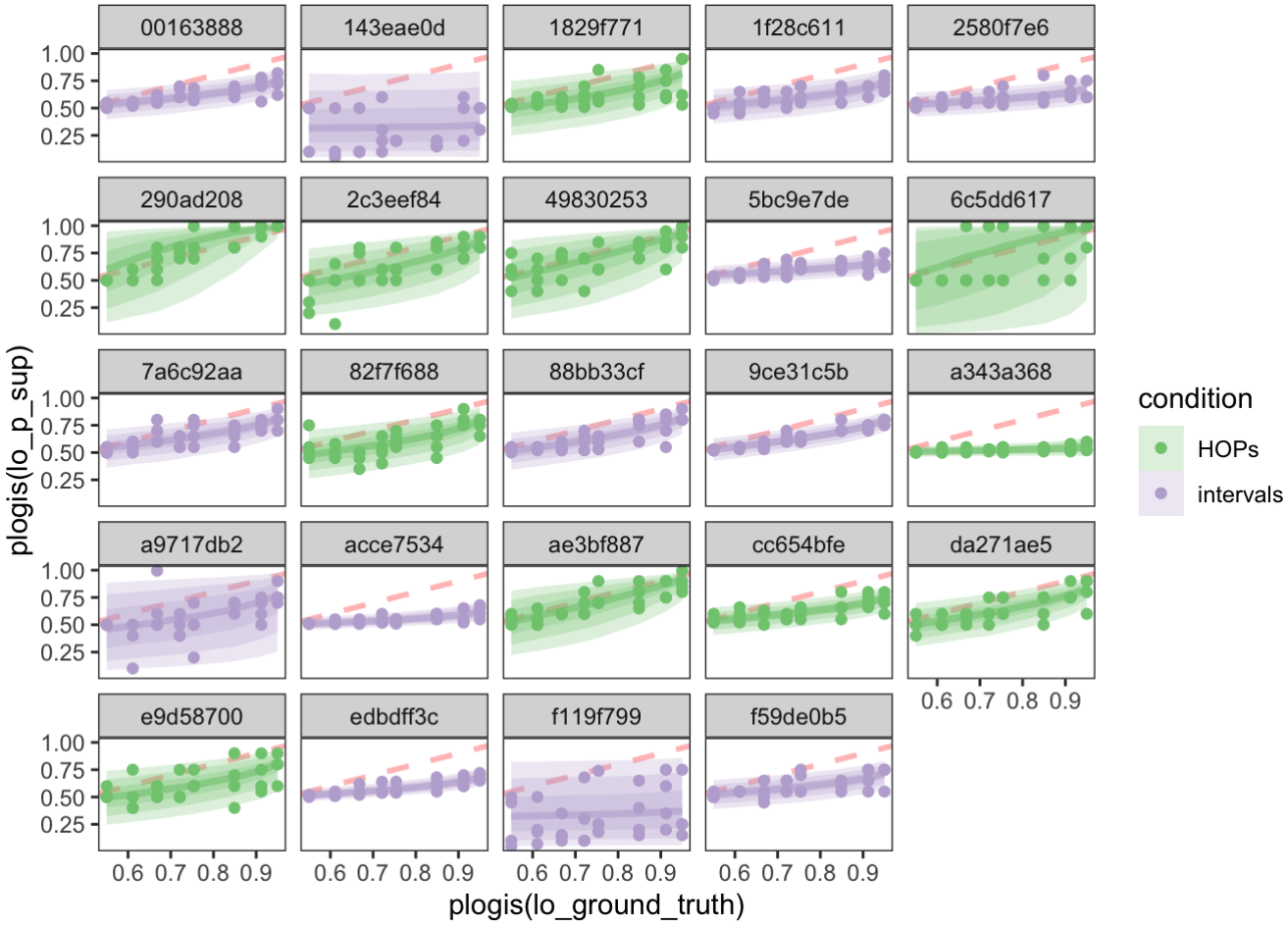
Let's take a look at predictions per worker and visualization condition to get a more granular sense of our model fit.

```
model_df %>%
  group_by(lo_ground_truth, worker_id, means, sd_diff, condition) %>%
  add_predicted_draws(m.wrkr.means.sd.vis.llo_p_sup) %>%
  ggplot(aes(x = lo_ground_truth, y = lo_p_sup, color = condition, fill = condition)) +
  geom_abline(intercept = 0, slope = 1, size = 1, alpha = .3, color = "red", linetype =
"dashed") + # ground truth
  stat_lineribbon(aes(y = .prediction), .width = c(.95, .80, .50), alpha = .25) +
  geom_point(data = model_df) +
  scale_fill_brewer(type = "qual", palette = 1) +
  scale_color_brewer(type = "qual", palette = 1) +
  coord_cartesian(xlim = quantile(model_df$lo_ground_truth, c(0, 1)),
                  ylim = quantile(model_df$lo_p_sup, c(0, 1))) +
  theme_bw() +
  theme(panel.grid = element_blank()) +
  facet_wrap(~ worker_id)
```



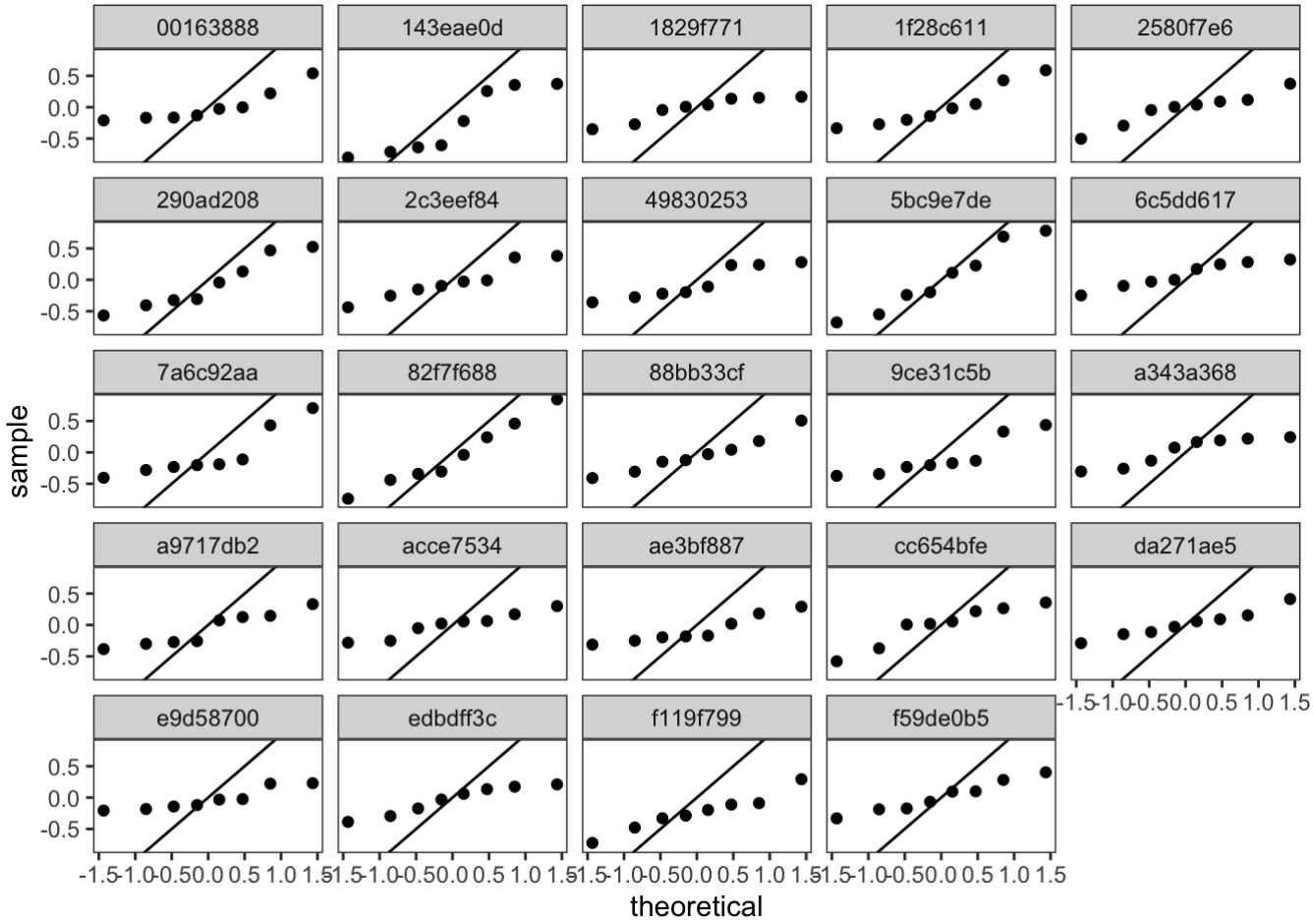
What does this look like in probability units?

```
model_df %>%
  group_by(lo_ground_truth, worker_id, means, sd_diff, condition) %>%
  add_predicted_draws(m.wrkr.means.sd.vis.llo_p_sup) %>%
  ggplot(aes(x = plogis(lo_ground_truth), y = plogis(lo_p_sup), color = condition, fill
= condition)) +
  geom_abline(intercept = 0, slope = 1, size = 1, alpha = .3, color = "red", linetype =
"dashed") + # ground truth
  stat_lineribbon(aes(y = plogis(.prediction)), .width = c(.95, .80, .50), alpha = .25)
 +
  geom_point(data = model_df) +
  scale_fill_brewer(type = "qual", palette = 1) +
  scale_color_brewer(type = "qual", palette = 1) +
  coord_cartesian(xlim = quantile(plogis(model_df$lo_ground_truth), c(0, 1)),
                  ylim = quantile(plogis(model_df$lo_p_sup), c(0, 1))) +
  theme_bw() +
  theme(panel.grid = element_blank()) +
  facet_wrap(~ worker_id)
```



To examine more closely whether our model has predictive validity at the level of each worker, we'll look at QQ plots for residuals at the worker level.

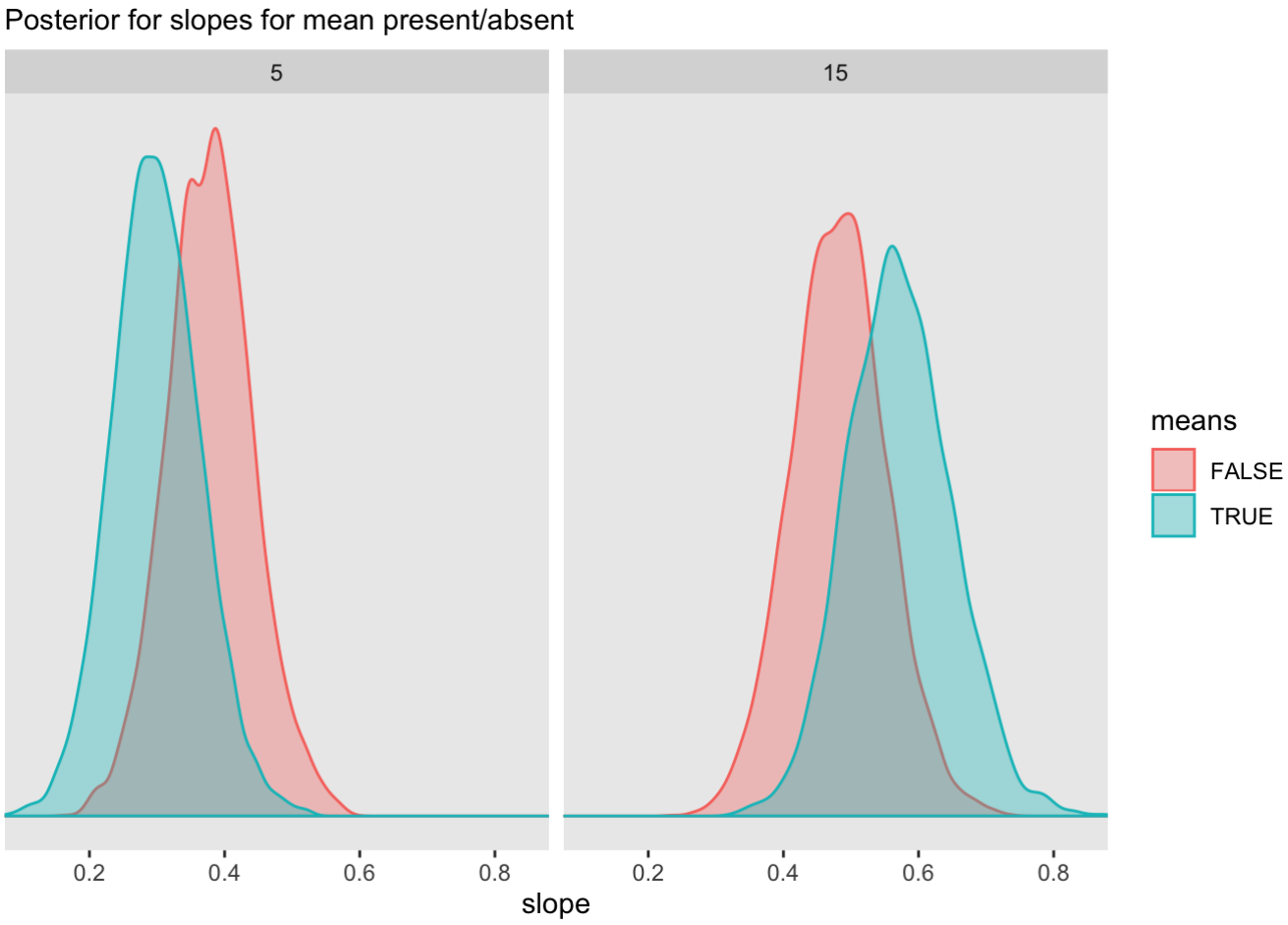
```
model_df %>%
  add_predicted_draws(m.wrkr.means.sd.vis.llo_p_sup) %>%
  group_by(lo_ground_truth, worker_id) %>%
  summarise(
    p_residual = mean(.prediction < lo_p_sup), # what proportion of predicted judgments
 are less than the observed response?
    z_residual = qnorm(p_residual)             # what are the z-scores of these cumulati
ve probabilities?
  ) %>%
  ggplot(aes(sample = z_residual)) +
  geom_qq() +
  geom_abline() +
  theme_bw() +
  theme(panel.grid = element_blank()) +
  facet_wrap(~ worker_id)
```



These still look pretty terrible.

What does the posterior for the slope of the LLO model look like when means are present vs absent at different levels of uncertainty, ignoring other manipulations?

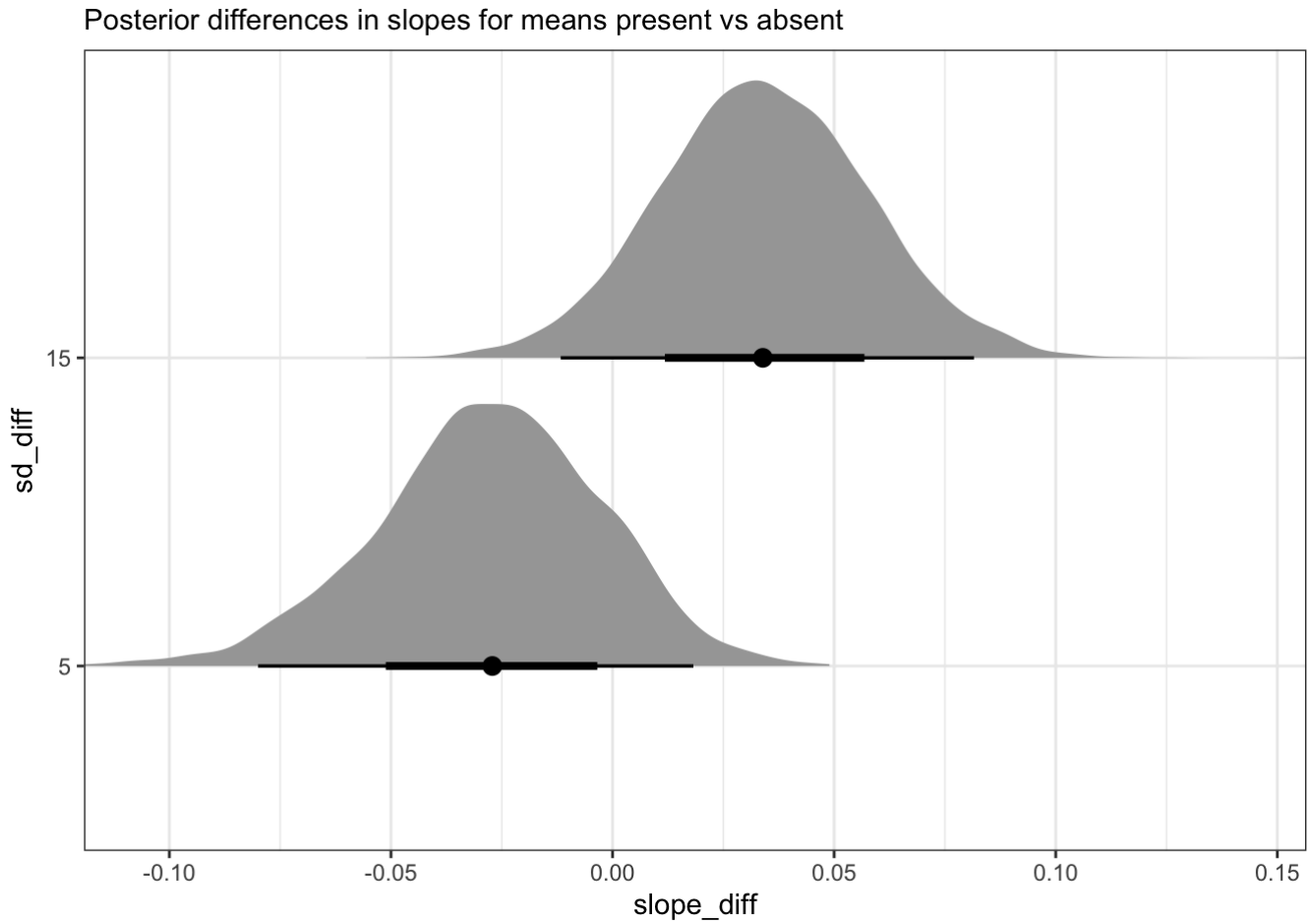
```
model_df %>%
  group_by(means, sd_diff, condition, trial) %>%
  data_grid(lo_ground_truth = c(0, 1)) %>%          # get fitted draws (in log odds units) only for ground truth of 0 and 1
  add_fitted_draws(m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.trial.sigma.gt.means.sd.trial, re_formula = NA) %>%
  compare_levels(.value, by = lo_ground_truth) %>%  # calculate the difference between fits at 1 and 0 (i.e., slope)
  rename(slope = .value) %>%
  group_by(means, sd_diff, .draw) %>%               # group by predictors to keep
  summarise(slope = weighted.mean(slope)) %>%       # marginalize out visualization condition by taking a weighted average
  ggplot(aes(x = slope, group = means, color = means, fill = means)) +
  geom_density(alpha = 0.35) +
  scale_x_continuous(expression(slope), expand = c(0, 0)) +
  scale_y_continuous(NULL, breaks = NULL) +
  labs(subtitle = "Posterior for slopes for mean present/absent") +
  theme(panel.grid = element_blank()) +
  facet_grid(. ~ sd_diff)
```



This effect suggests that adding means has a debiasing effect on average when visualized uncertainty is high and a biasing effect when uncertainty is low (marginalizing across visualization conditions). Our estimate is more precise with the more complex model.

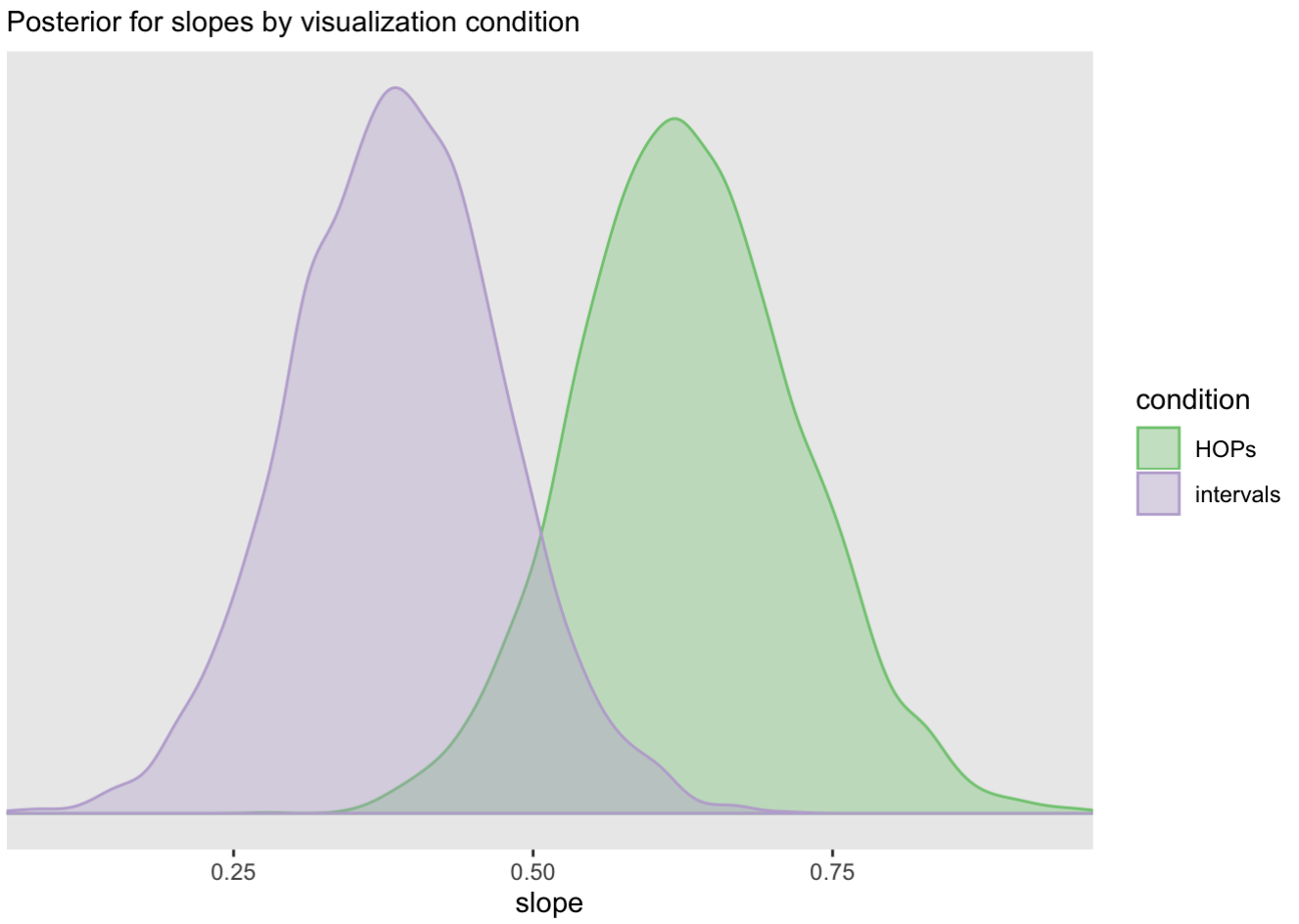
Let's look at this difference in a forest plot style display.

```
model_df %>%
  group_by(means, sd_diff, condition) %>%
  data_grid(lo_ground_truth = c(0, 1)) %>%                    # get fitted draws (in log
odds units) only for ground truth of 0 and 1
  add_fitted_draws(m.wrkr.means.sd.vis.llo_p_sup, re_formula = NA) %>%
  compare_levels(.value, by = lo_ground_truth) %>%            # calculate the difference
between fits at 1 and 0 (i.e., slope)
  compare_levels(.value, by = means) %>%                      # look at differences in s
lopes between means present vs absent
  rename(slope_diff = .value) %>%
  group_by(sd_diff, .draw) %>%                                # group by predictors to k
eep
  summarise(slope_diff = weighted.mean(slope_diff)) %>%       # marginalize out means pr
esent/absent by taking a weighted average
  ggplot(aes(x = slope_diff, y = sd_diff)) +
  stat_halfeyeh() +
  scale_x_continuous(expression(slope_diff), expand = c(0, 0)) +
  labs(subtitle = "Posterior differences in slopes for means present vs absent") +
  theme_bw()
```



What does the posterior for the slope in each visualization condition look like, marginalizing across other factors?

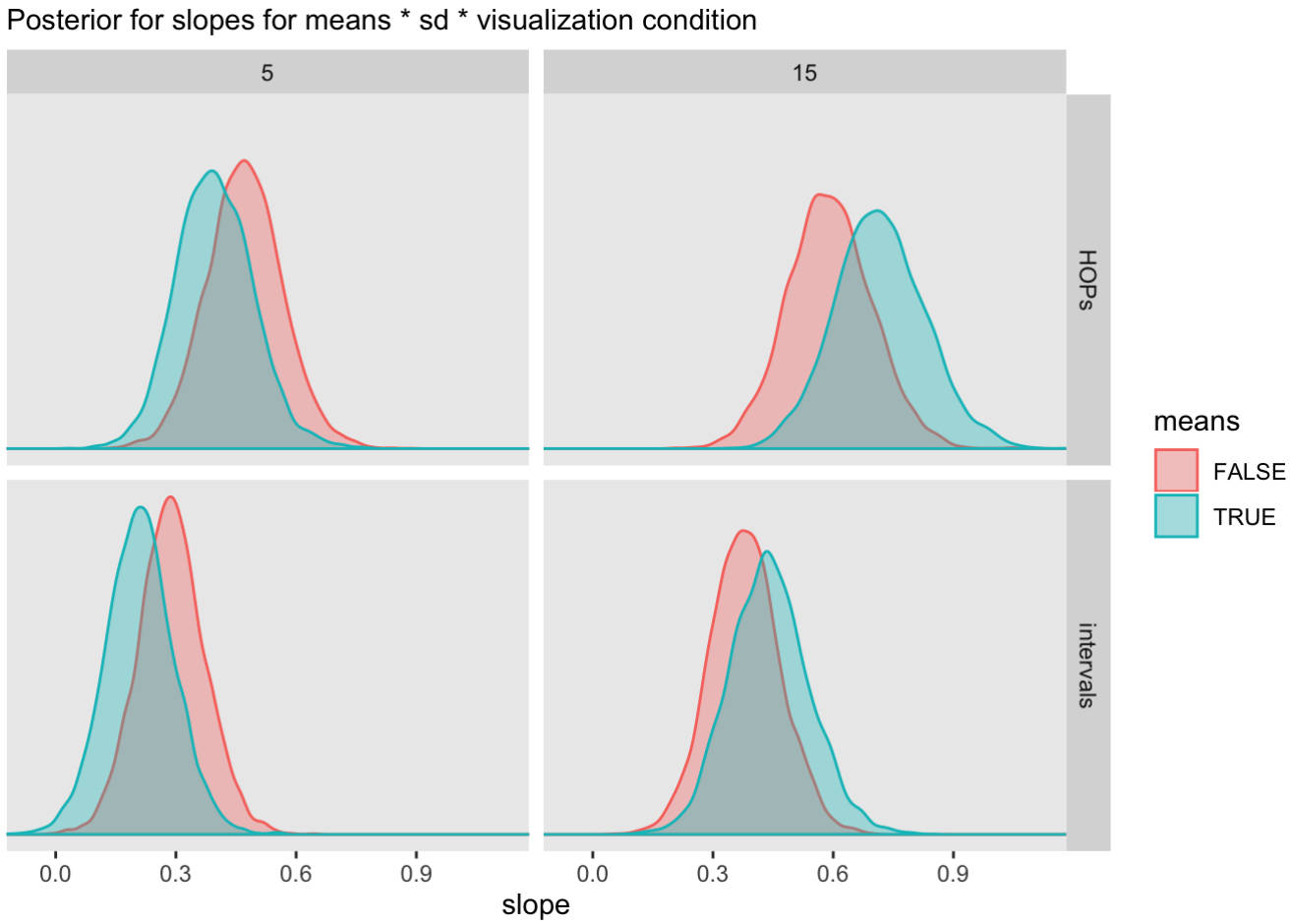
```
model_df %>%
  group_by(means, sd_diff, condition) %>%
  data_grid(lo_ground_truth = c(0, 1)) %>%          # get fitted draws (in log odds unit
s) only for ground truth of 0 and 1
  add_fitted_draws(m.wrkr.means.sd.vis.llo_p_sup, re_formula = NA) %>%
  compare_levels(.value, by = lo_ground_truth) %>%  # calculate the difference between f
its at 1 and 0 (i.e., slope)
  rename(slope = .value) %>%
  group_by(condition, .draw) %>%                    # group by predictors to keep
  summarise(slope = weighted.mean(slope)) %>%       # marginalize out means present/abse
nt by taking a weighted average
  ggplot(aes(x = slope, group = condition, color = condition, fill = condition)) +
  geom_density(alpha = 0.35) +
  scale_fill_brewer(type = "qual", palette = 1) +
  scale_color_brewer(type = "qual", palette = 1) +
  scale_x_continuous(expression(slope), expand = c(0, 0)) +
  scale_y_continuous(NULL, breaks = NULL) +
  labs(subtitle = "Posterior for slopes by visualization condition") +
  theme(panel.grid = element_blank())
```



Recall that a slope of 1 on the logit scale reflects no bias. This suggests that users are biased toward responses of 50% on the probability scale in both conditions, but especially with intervals. HOPs seem to have a substantial debiasing effect on effect size judgments when we marginalize across other manipulations.

What if we break these marginal effects down into simple effects for the interaction of the presence/absence of the mean, level of visualized uncertainty, and visualization condition?

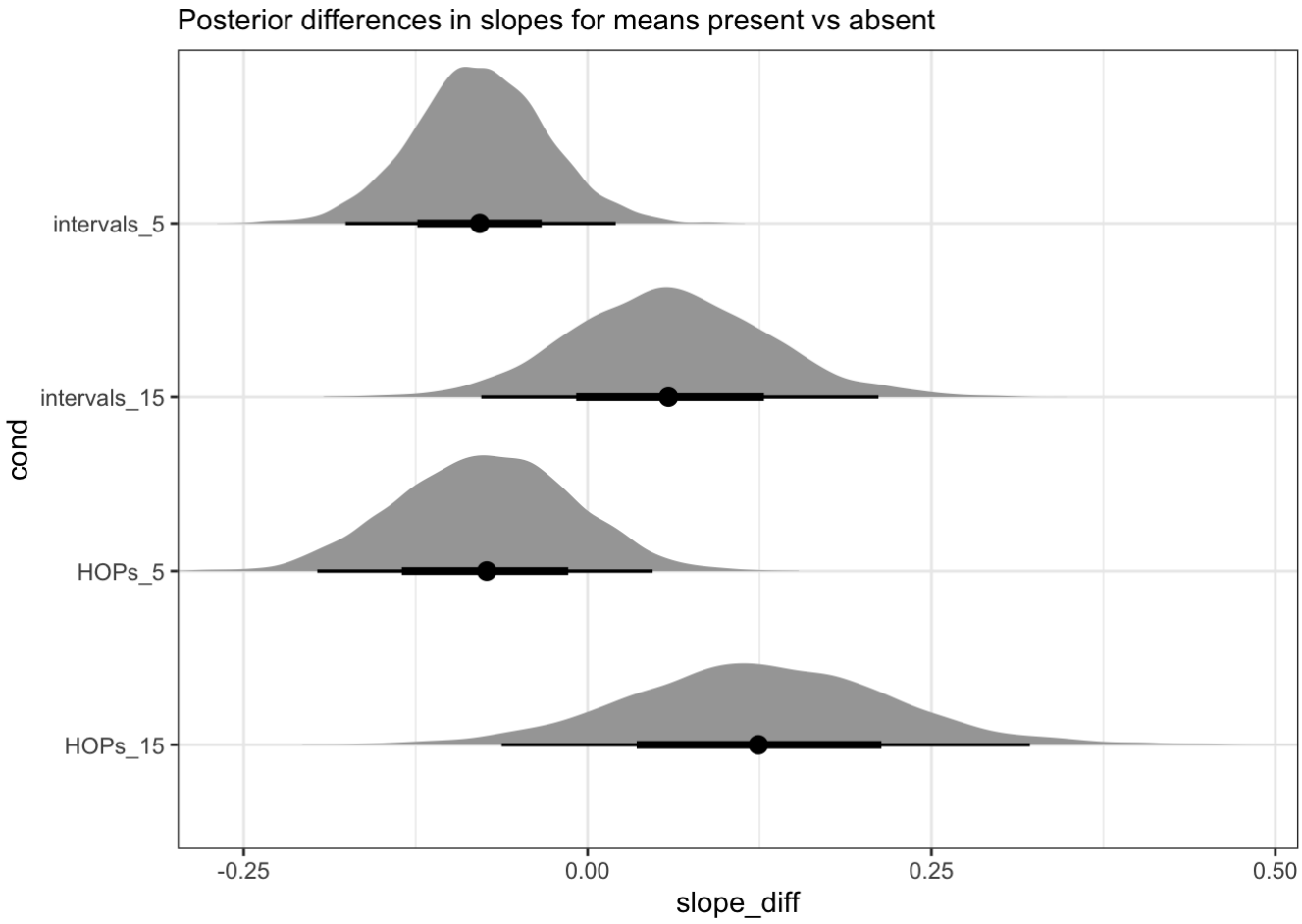
```
model_df %>%
  group_by(means, sd_diff, condition, trial) %>%
  data_grid(lo_ground_truth = c(0, 1)) %>%          # get fitted draws (in log odds units) only for ground truth of 0 and 1
  add_fitted_draws(m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.trial.sigma.gt.means.sd.trial, re_formula = NA) %>%
  compare_levels(.value, by = lo_ground_truth) %>%  # calculate the difference between fits at 1 and 0 (i.e., slope)
  rename(slope = .value) %>%
  group_by(means, sd_diff, condition, .draw) %>%    # group by predictors to keep
  summarise(slope = weighted.mean(slope)) %>%       # marginalize out means present/absent by taking a weighted average
  ggplot(aes(x = slope, group = means, color = means, fill = means)) +
  geom_density(alpha = 0.35) +
  scale_x_continuous(expression(slope), expand = c(0, 0)) +
  scale_y_continuous(NULL, breaks = NULL) +
  labs(subtitle = "Posterior for slopes for means * sd * visualization condition") +
  theme(panel.grid = element_blank()) +
  facet_grid(condition ~ sd_diff)
```



Again, this is what we expected to see. However, it is not clear form this chart if the effect is reliable.

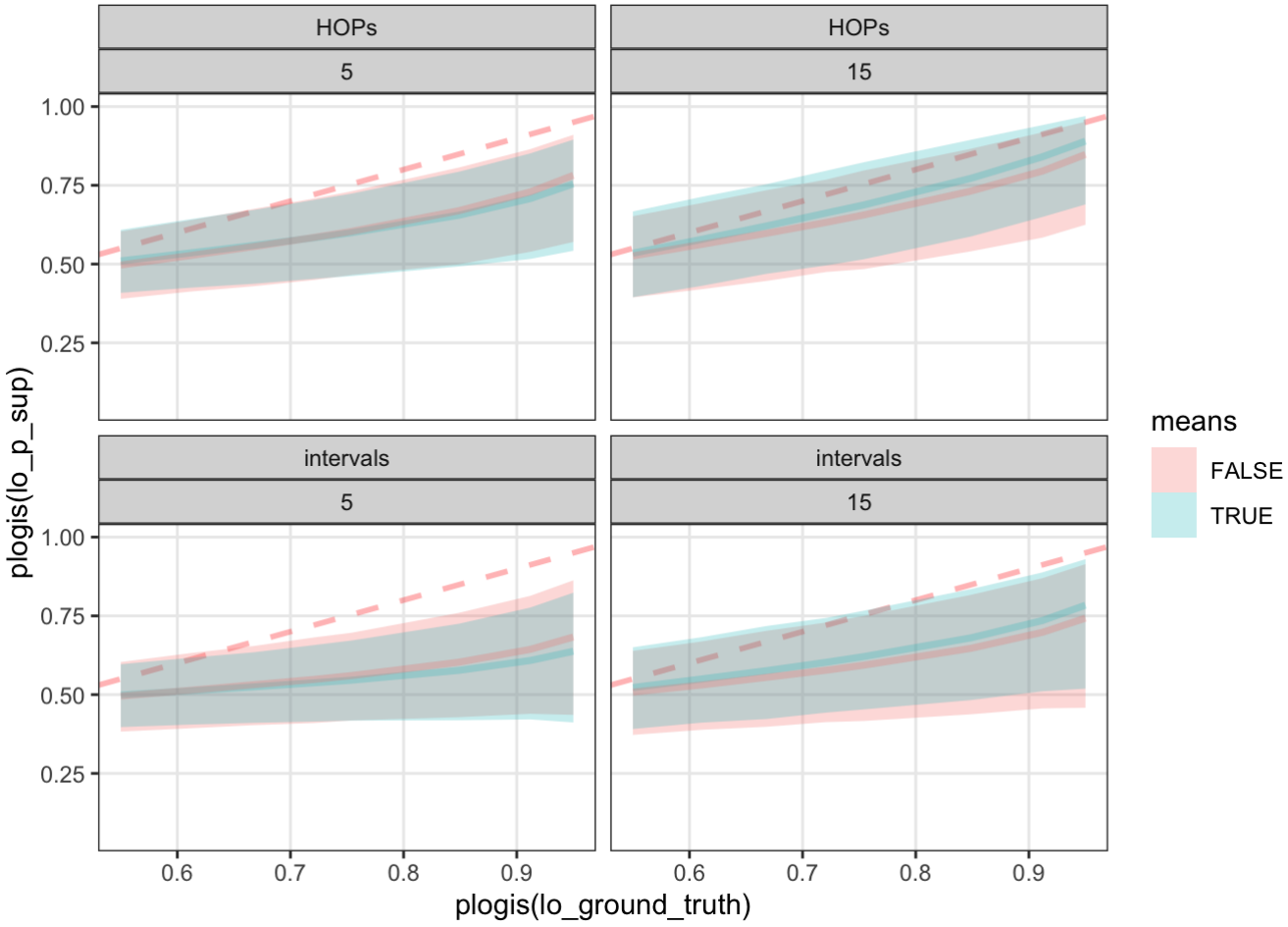
Let's look at the differences in a forest plot style display.

```
model_df %>%
  group_by(means, sd_diff, condition, trial) %>%
  data_grid(lo_ground_truth = c(0, 1)) %>%          # get fitted draws (in log odds units) only for ground truth of 0 and 1
  add_fitted_draws(m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.trial.sigma.gt.means.sd.trial, re_formula = NA) %>%
  compare_levels(.value, by = lo_ground_truth) %>%  # calculate the difference between fits at 1 and 0 (i.e., slope)
  rename(slope = .value) %>%
  group_by(means, sd_diff, condition, .draw) %>%     # group by predictors to keep
  summarise(slope = weighted.mean(slope)) %>%       # marginalize by taking a weighted average
  compare_levels(slope, by = means) %>%             # look at differences in slopes between means present vs absent
  rename(slope_diff = slope) %>%
  unite(cond, condition, sd_diff, sep = "_", remove = FALSE) %>%
  ggplot(aes(x = slope_diff, y = cond)) +
  stat_halfeyeh() +
  scale_x_continuous(expression(slope_diff), expand = c(0, 0)) +
  labs(subtitle = "Posterior differences in slopes for means present vs absent") +
  theme_bw()
```



What is the predicted pattern for responses for the average worker in each cell of this interaction?

```
model_df %>%
  group_by(lo_ground_truth, means, sd_diff, condition, trial) %>%
  add_predicted_draws(m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.trial.sigma.gt.means.sd.trial, re_formula = NA) %>%
  ggplot(aes(x = plogis(lo_ground_truth), y = plogis(lo_p_sup), color = means, fill = means)) +
  geom_abline(intercept = 0, slope = 1, size = 1, alpha = .3, color = "red", linetype = "dashed") + # ground truth
  stat_lineribbon(aes(y = plogis(.prediction)), .width = c(.95), alpha = .25) +
  coord_cartesian(xlim = quantile(plogis(model_df$lo_ground_truth), c(0, 1)),
                  ylim = quantile(plogis(model_df$lo_p_sup), c(0, 1))) +
  theme_bw() +
  theme(panel.grid.minor = element_blank()) +
  facet_wrap(condition ~ sd_diff)
```



Add Predictors for Block Order

Let's add block order to our previous model, just to check if the effect of the mean on judgments depends on block order. We'll model this as a fixed effects interaction between block order and the presence absence of means.

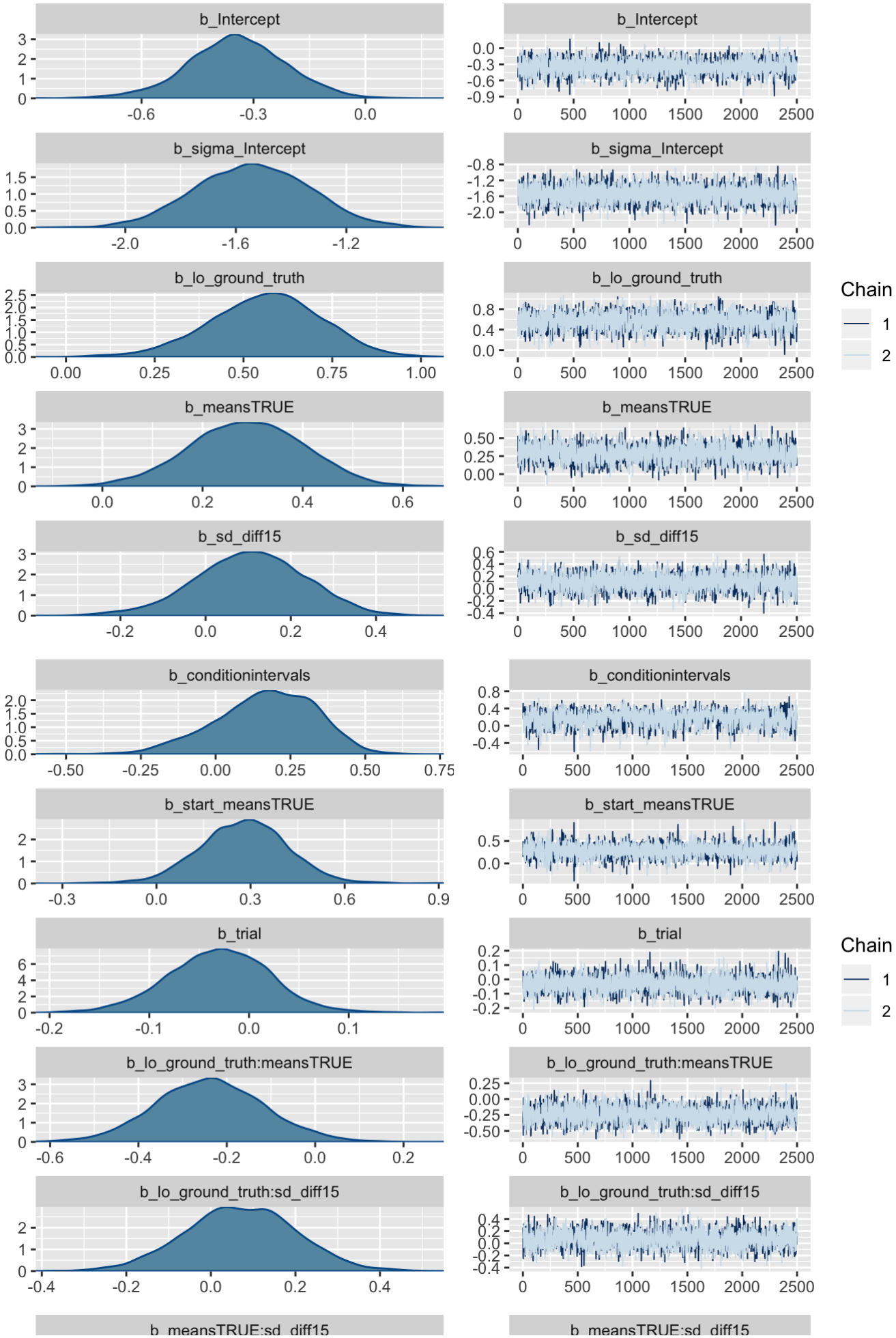
We use the same priors as we did for the previous model. Now, let's fit the model to our data.

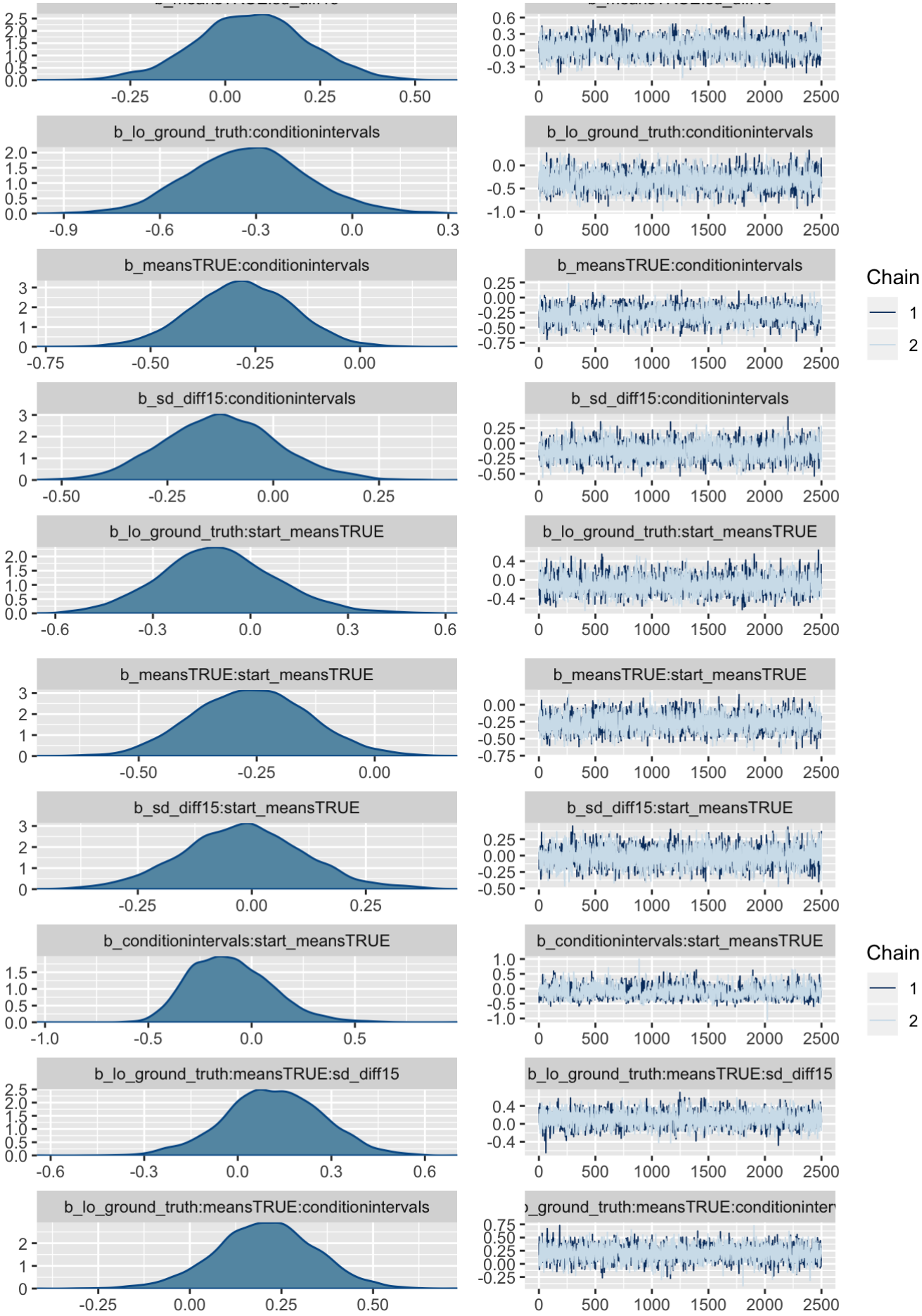
```
# hierarchical LLO model
m.max <- brm(
  data = model_df, family = "gaussian",
  formula = bf(lo_p_sup ~  (1 + lo_ground_truth*means*sd_diff + trial|sharecor|worker_i
d) + lo_ground_truth*means*sd_diff*condition*start_means + trial,
               sigma ~ (1 + lo_ground_truth + means*sd_diff + trial|sharecor|worker_id)
 + lo_ground_truth + means*sd_diff*start_means + trial),
  prior = c(prior(normal(1, 0.5), class = b),
            prior(normal(1.3, 1), class = Intercept),
            prior(normal(0, 0.15), class = sd, group = worker_id),
            prior(normal(0, 0.3), class = b, dpar = sigma),
            prior(normal(0, 0.15), class = sd, dpar = sigma),
            prior(lkj(4), class = cor)),
  iter = 3000, warmup = 500, chains = 2, cores = 2,
  control = list(adapt_delta = 0.99, max_treedepth = 12),
  file = "model-fits/llo_mdl-max")
```

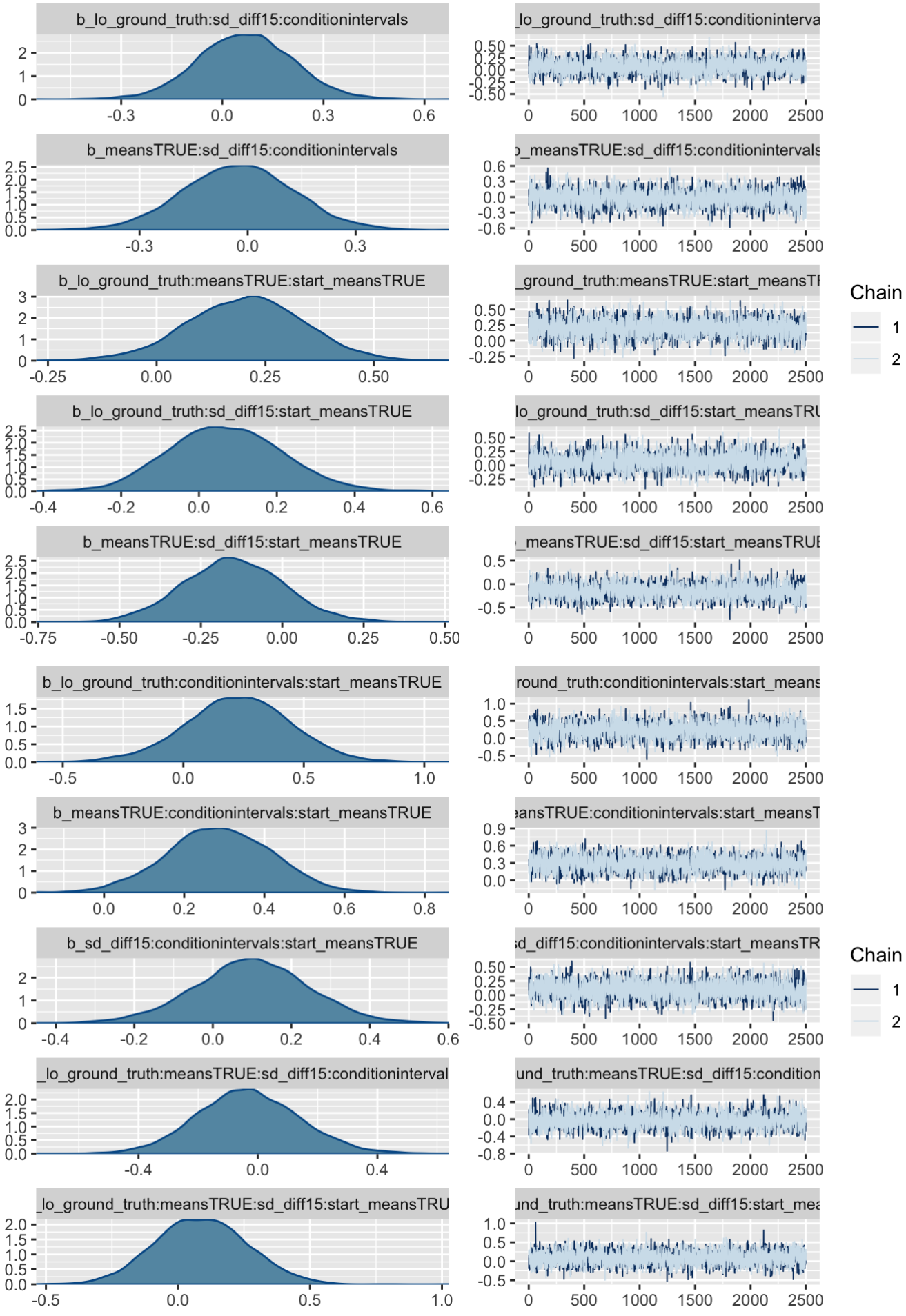
Check diagnostics:

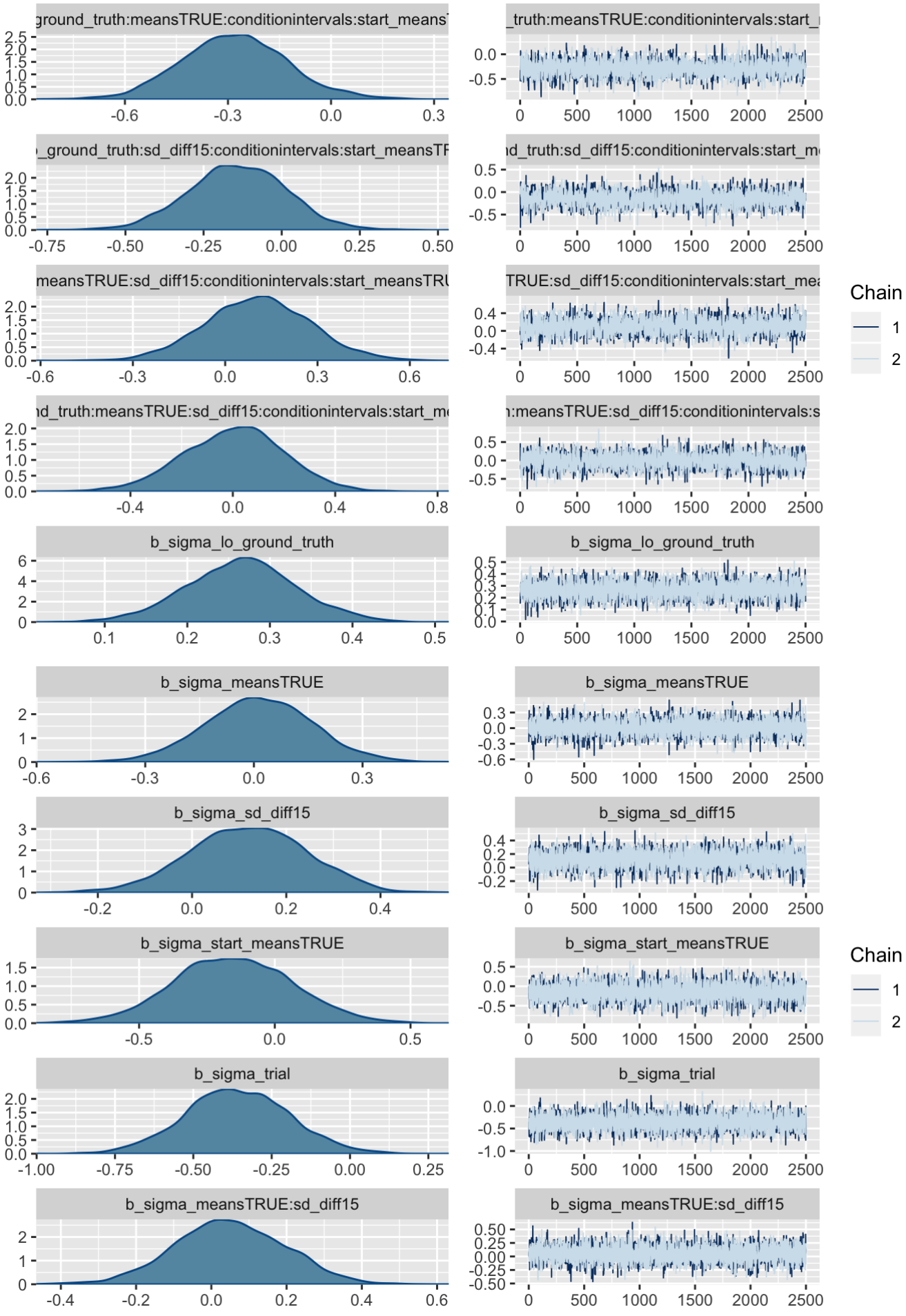
- Trace plots

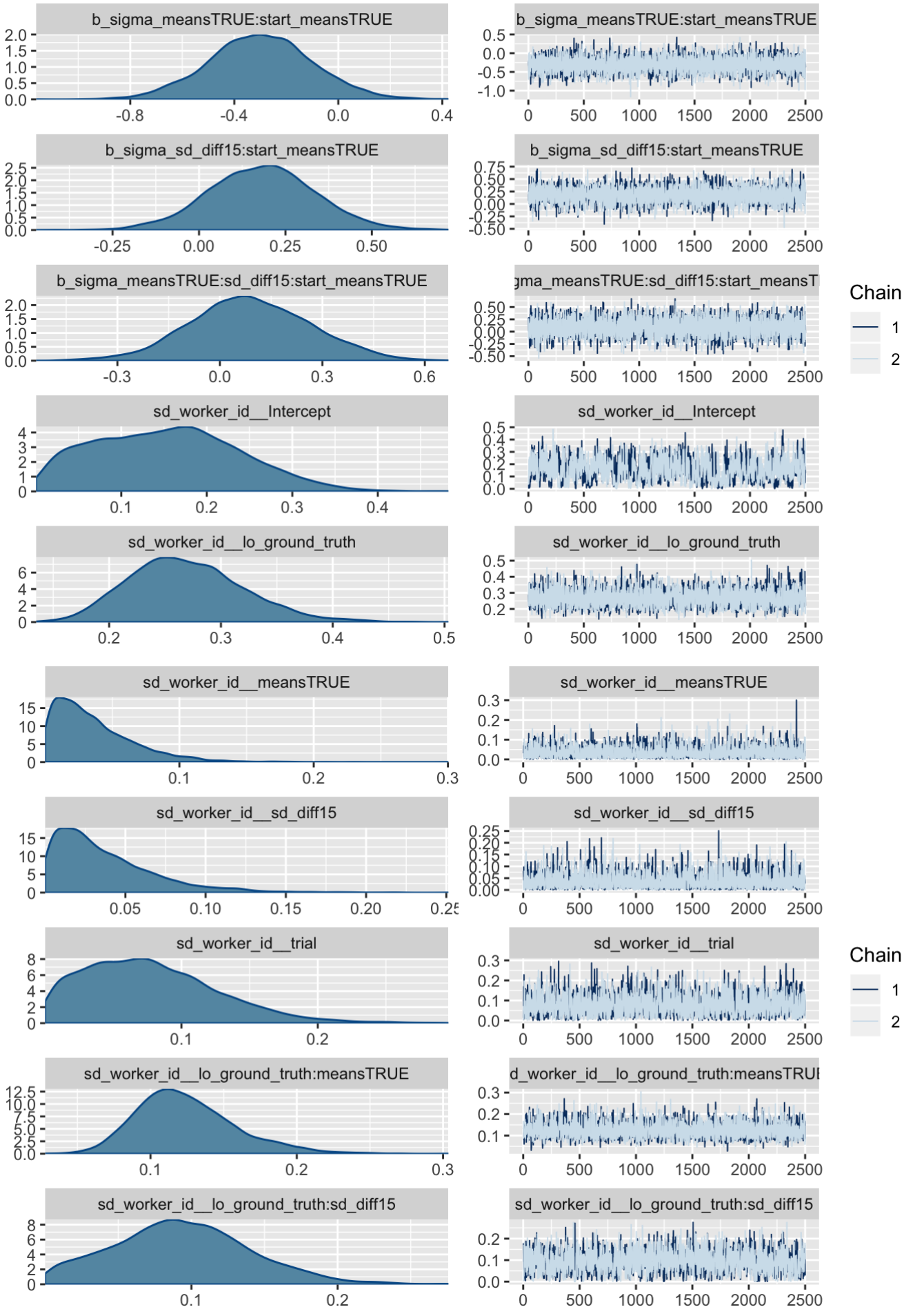
```
# trace plots
plot(m.max)
```

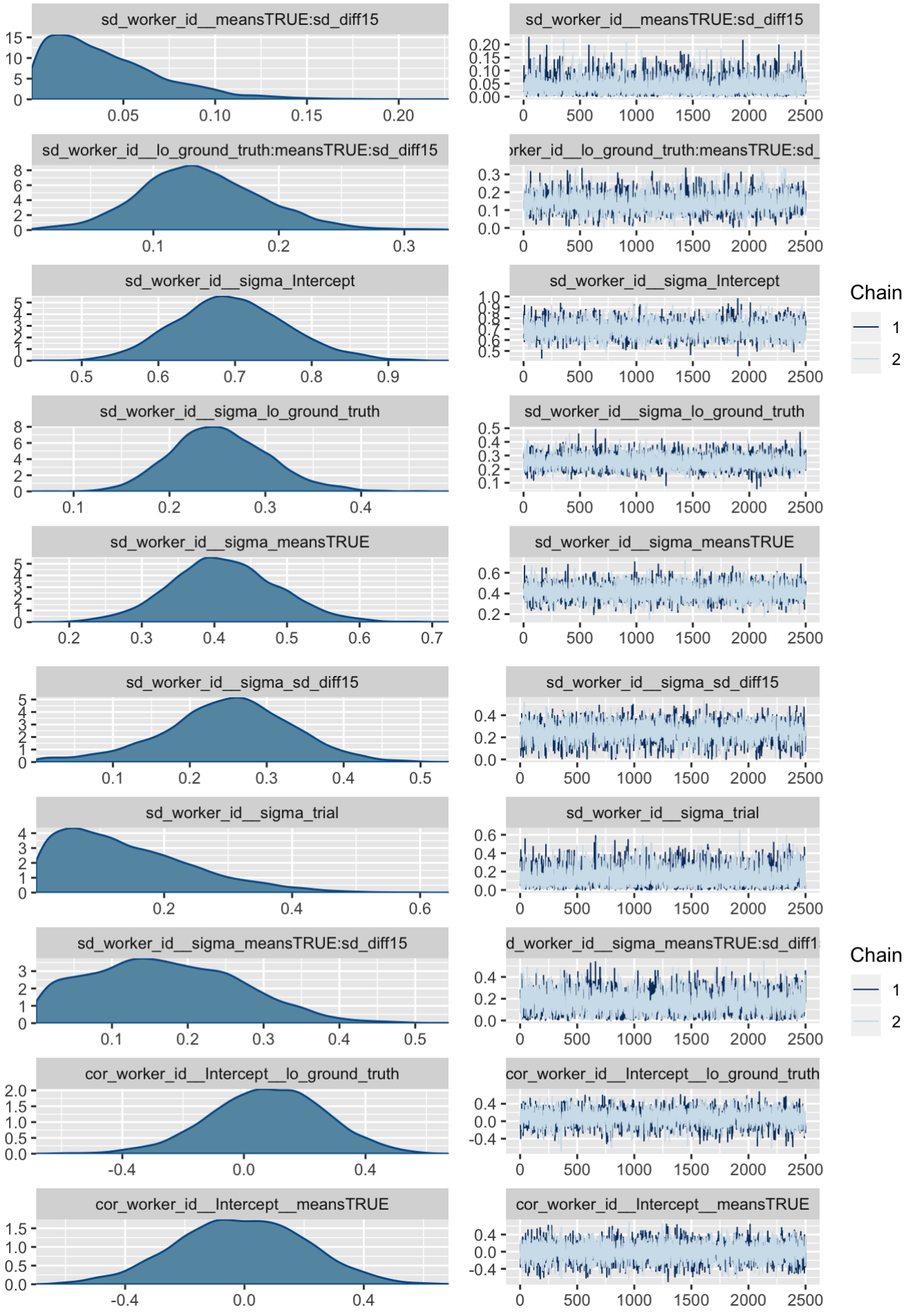

b_Intercept
b_sigma_Intercept
b_lo_ground_truth
b_meansTRUE
b_sd_diff15
b_conditionintervals
b_start_meansTRUE
b_trial
b_lo_ground_truth:meansTRUE
b_lo_ground_truth:sd_diff15
b_meansTRUE:sd_diff15
Chain
1
2

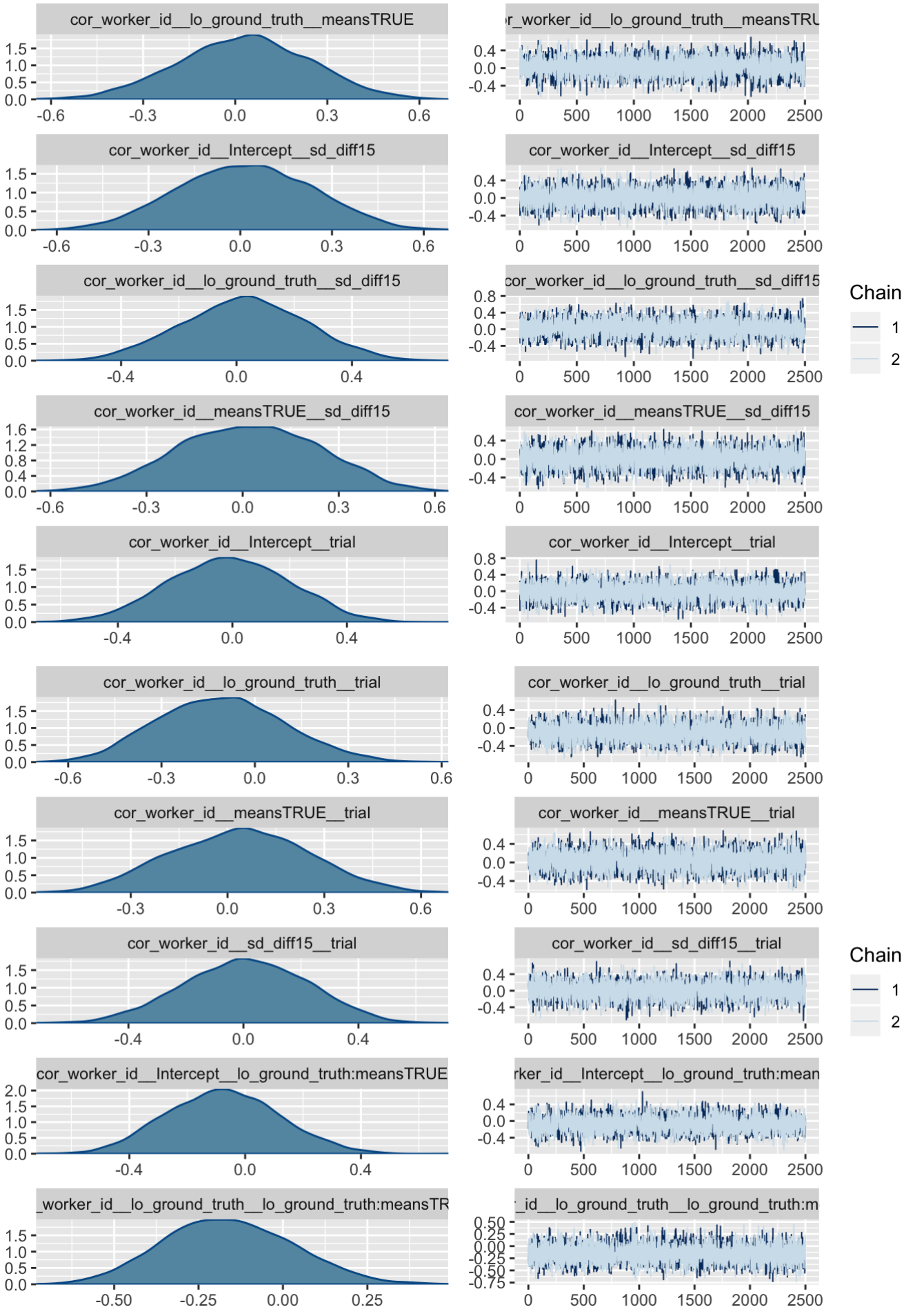

b_lo_ground_truth:conditionintervals
b_meansTRUE:conditionintervals
b_sd_diff15:conditionintervals
b_lo_ground_truth:start_meansTRUE
b_meansTRUE:start_meansTRUE
b_sd_diff15:start_meansTRUE
b_conditionintervals:start_meansTRUE
b_lo_ground_truth:meansTRUE:sd_diff15
b_lo_ground_truth:meansTRUE:conditionintervals
Chain
1
2

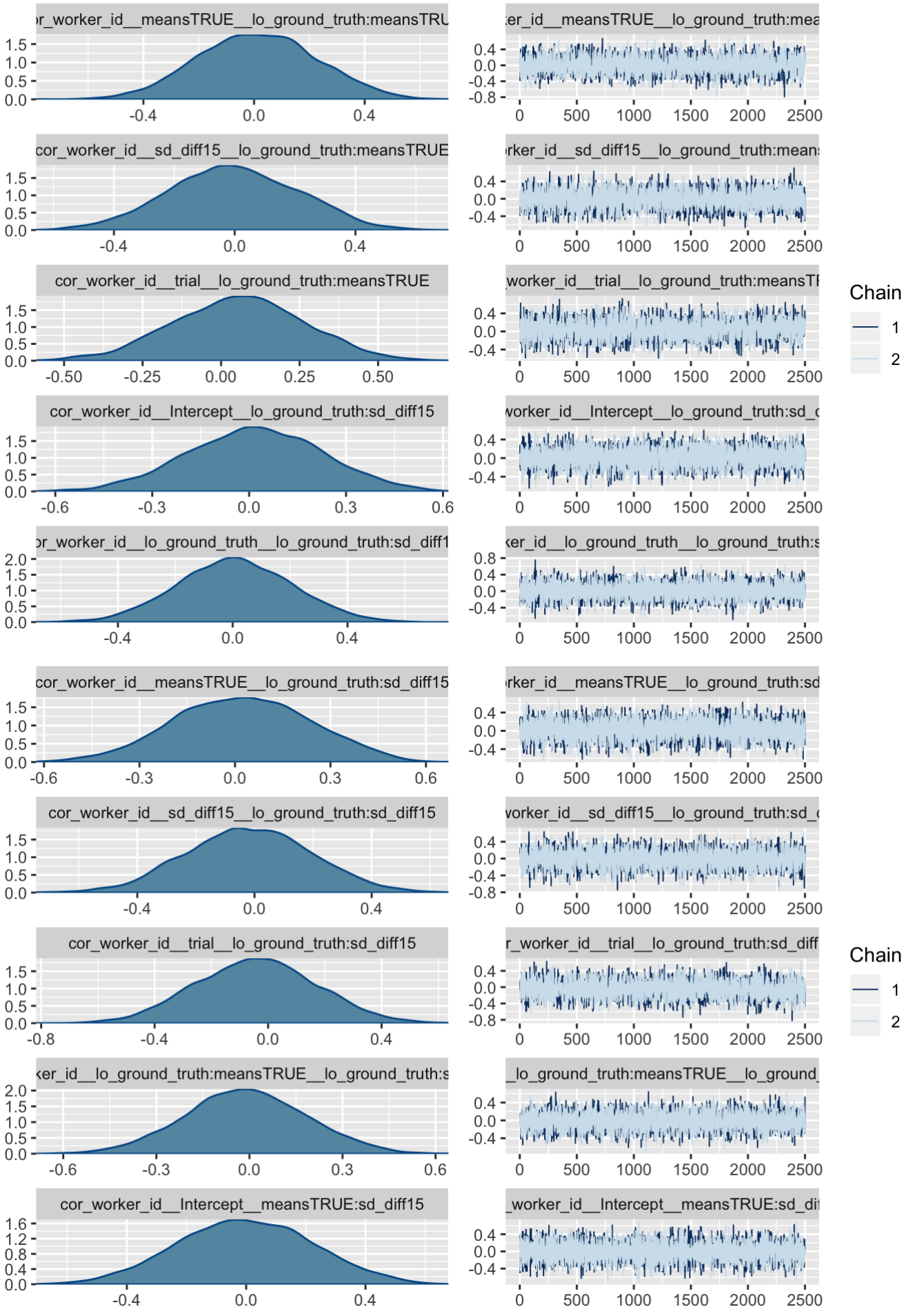

b_lo_ground_truth:sd_diff15:conditionintervals
b_meansTRUE:sd_diff15:conditionintervals
b_lo_ground_truth:meansTRUE:start_meansTRUE
b_lo_ground_truth:sd_diff15:start_meansTRUE
b_meansTRUE:sd_diff15:start_meansTRUE
b_lo_ground_truth:conditionintervals:start_meansTRUE
b_meansTRUE:conditionintervals:start_meansTRUE
b_sd_diff15:conditionintervals:start_meansTRUE
_lo_ground_truth:meansTRUE:sd_diff15:conditioninterval
_lo_ground_truth:meansTRUE:sd_diff15:start_meansTRU
Chain
1
2

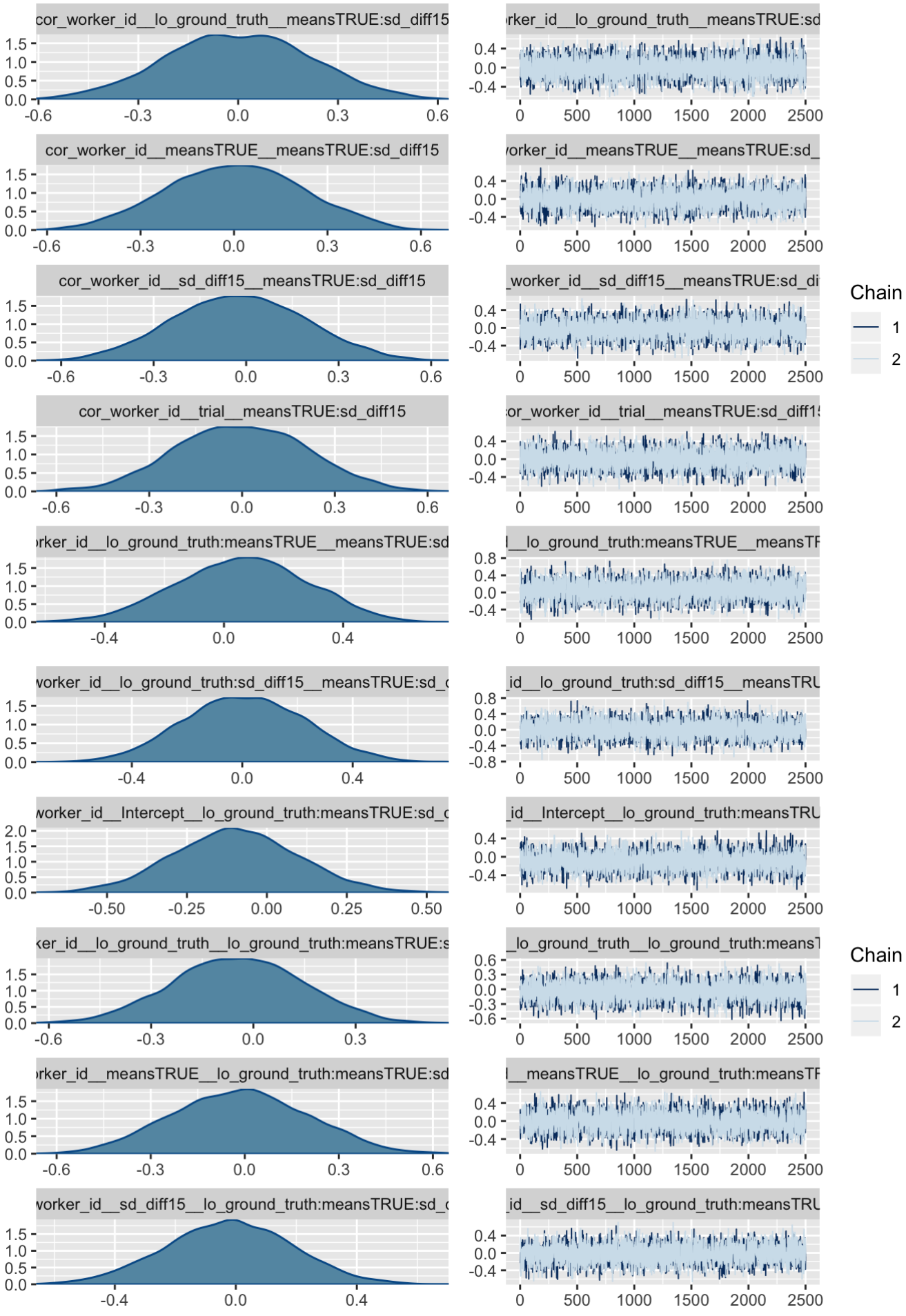

ground_truth:meansTRUE:conditionintervals:start_means
truth:meansTRUE:conditionintervals:start
_ground_truth:sd_diff15:conditionintervals:start_meansTR
d_truth:sd_diff15:conditionintervals:start_m
meansTRUE:sd_diff15:conditionintervals:start_meansTRU
TRUE:sd_diff15:conditionintervals:start_mea
d_truth:meansTRUE:sd_diff15:conditionintervals:start_m
:meansTRUE:sd_diff15:conditionintervals:s
b_sigma_lo_ground_truth
b_sigma_meansTRUE
b_sigma_sd_diff15
b_sigma_start_meansTRUE
b_sigma_trial
b_sigma_meansTRUE:sd_diff15
Chain
1
2

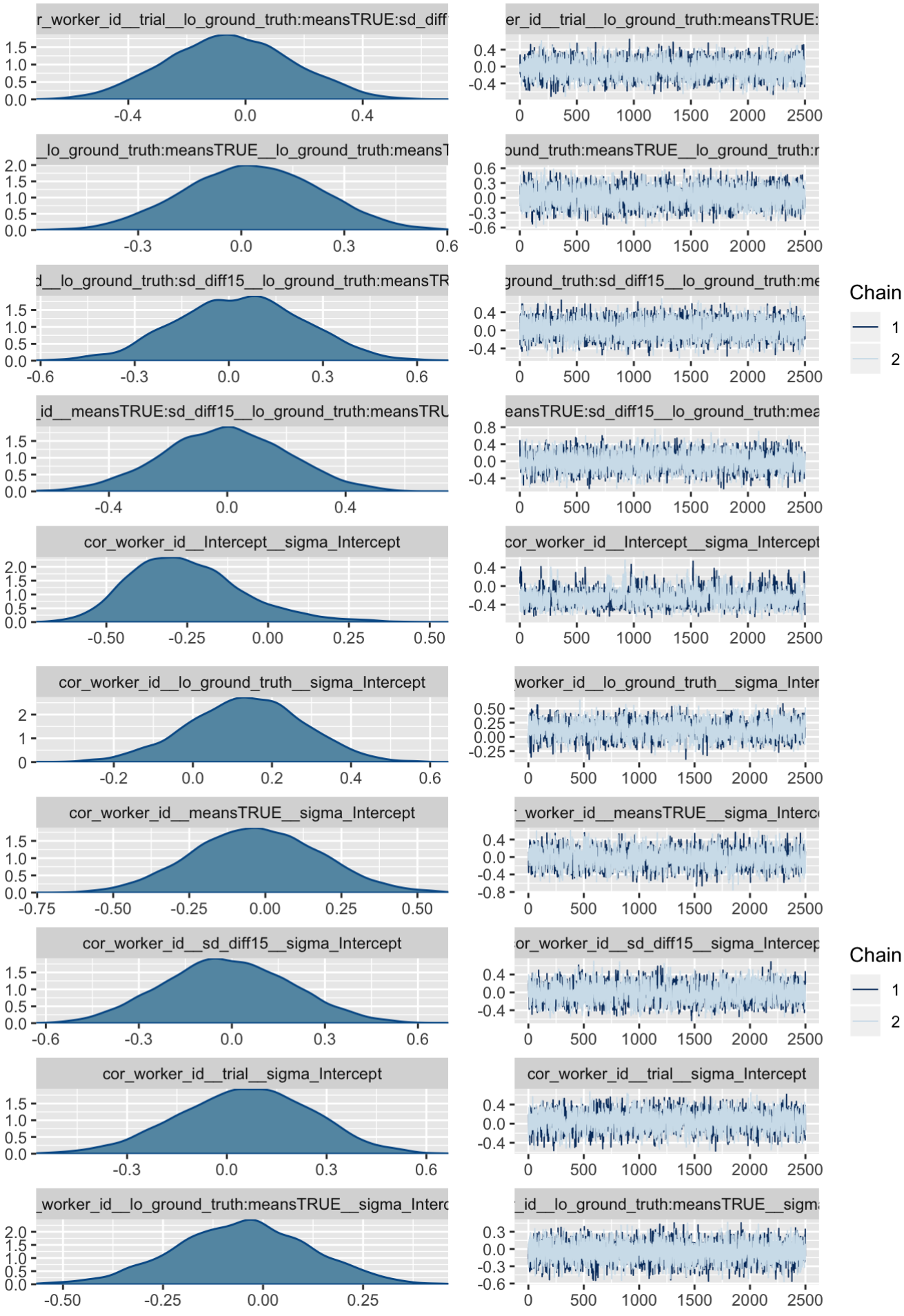

b_sigma_meansTRUE:start_meansTRUE
b_sigma_meansTRUE:start_meansTRUE
b_sigma_sd_diff15:start_meansTRUE
b_sigma_sd_diff15:start_meansTRUE
b_sigma_meansTRUE:sd_diff15:start_meansTRUE
gma_meansTRUE:sd_diff15:start_meansT
sd_worker_id__Intercept
sd_worker_id__Intercept
sd_worker_id__lo_ground_truth
sd_worker_id__lo_ground_truth
sd_worker_id__meansTRUE
sd_worker_id__meansTRUE
sd_worker_id__sd_diff15
sd_worker_id__sd_diff15
sd_worker_id__trial
sd_worker_id__trial
sd_worker_id__lo_ground_truth:meansTRUE
d_worker_id__lo_ground_truth:meansTRUI
sd_worker_id__lo_ground_truth:sd_diff15
sd_worker_id__lo_ground_truth:sd_diff15
Chain
1
2
Chain
1
2

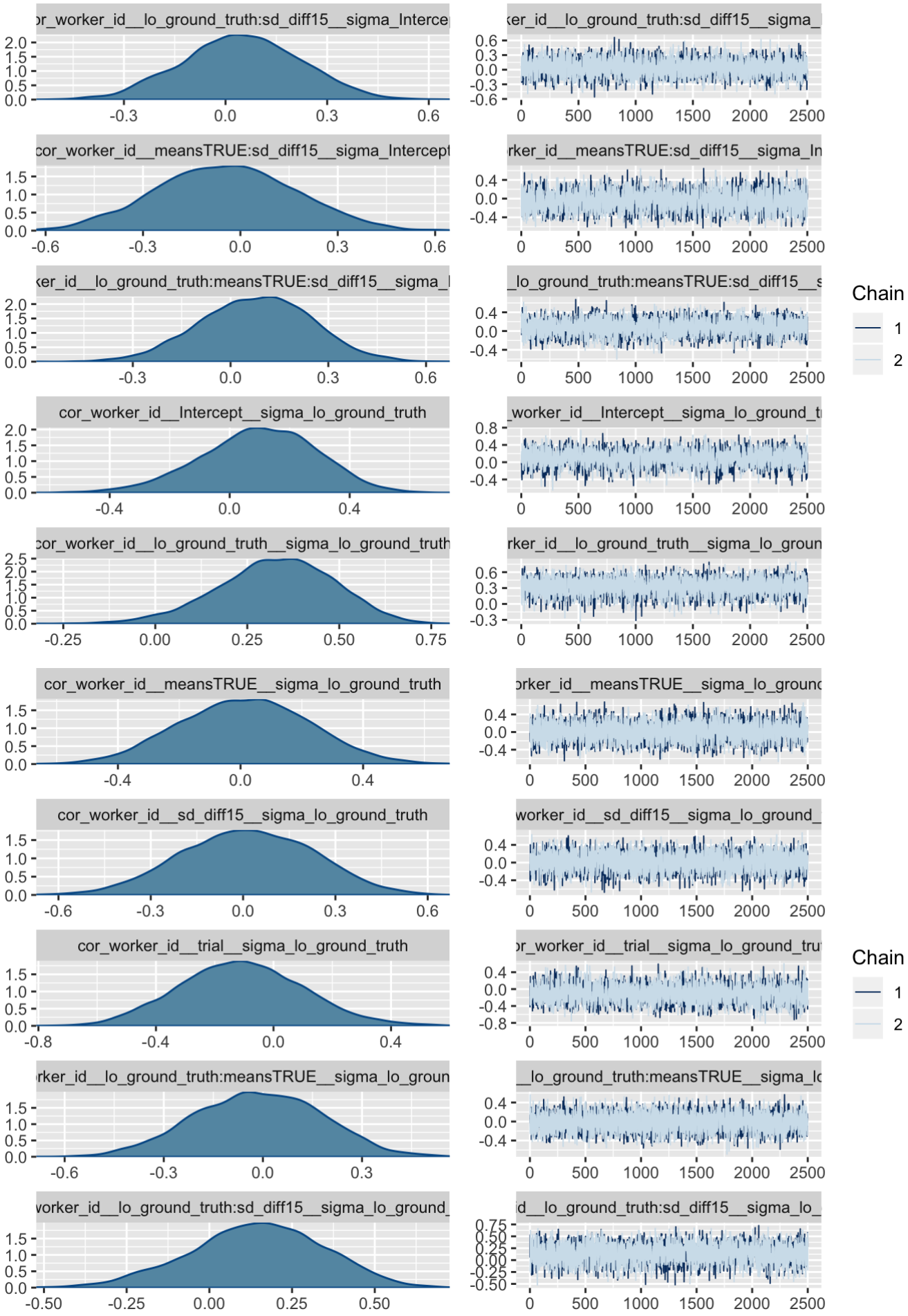

sd_worker_id__meansTRUE:sd_diff15
sd_worker_id__lo_ground_truth:meansTRUE:sd_diff15
sd_worker_id__sigma_Intercept
sd_worker_id__sigma_lo_ground_truth
sd_worker_id__sigma_meansTRUE
sd_worker_id__sigma_sd_diff15
sd_worker_id__sigma_trial
sd_worker_id__sigma_meansTRUE:sd_diff15
cor_worker_id__Intercept__lo_ground_truth
cor_worker_id__Intercept__meansTRUE
Chain
1
2

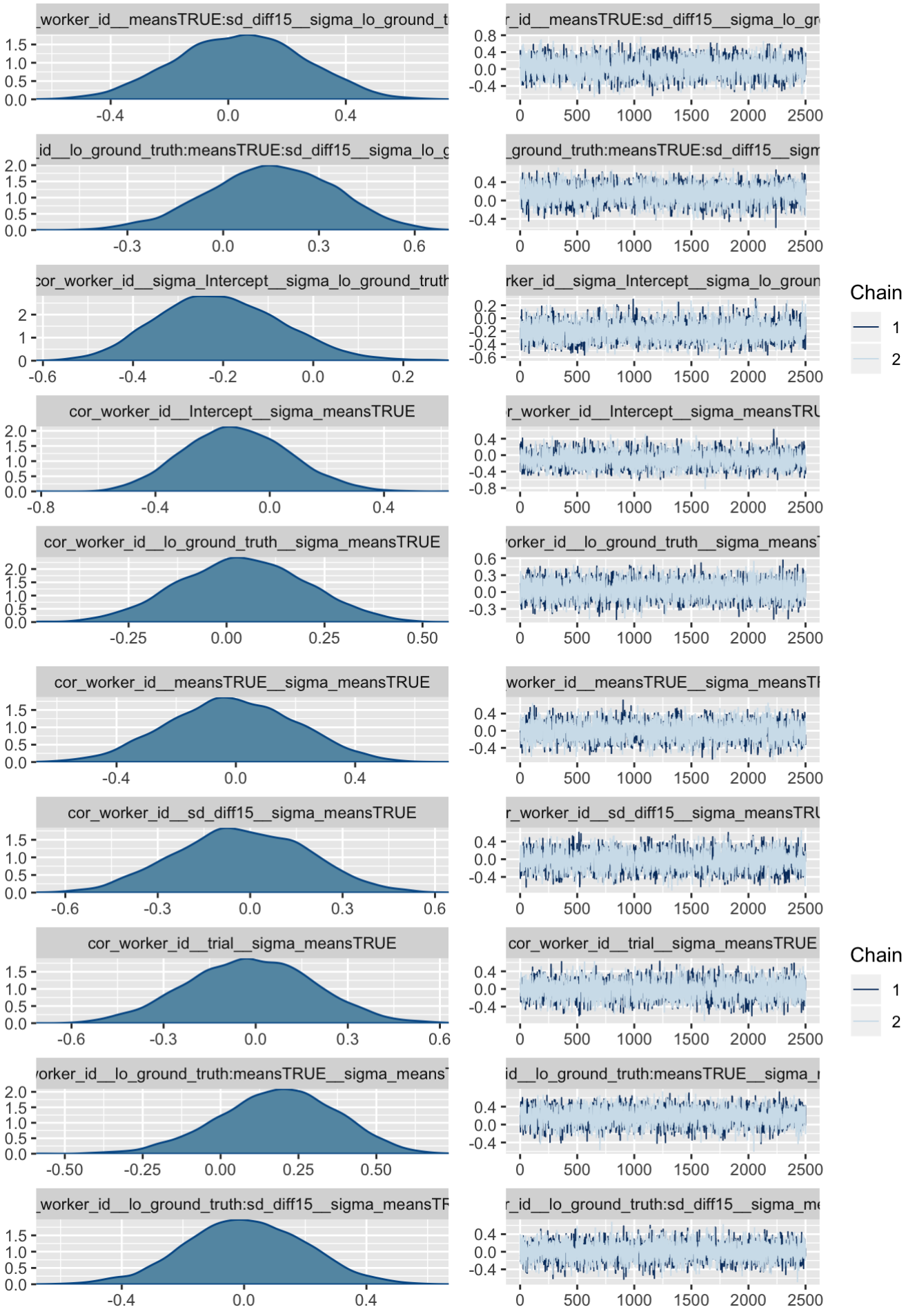

cor_worker_id__lo_ground_truth__meansTRUE
cor_worker_id__Intercept__sd_diff15
cor_worker_id__lo_ground_truth__sd_diff15
cor_worker_id__meansTRUE__sd_diff15
cor_worker_id__Intercept__trial
cor_worker_id__lo_ground_truth__trial
cor_worker_id__meansTRUE__trial
cor_worker_id__sd_diff15__trial
cor_worker_id__Intercept__lo_ground_truth:meansTRUE
_worker_id__lo_ground_truth__lo_ground_truth:meansTR
Chain
1
2

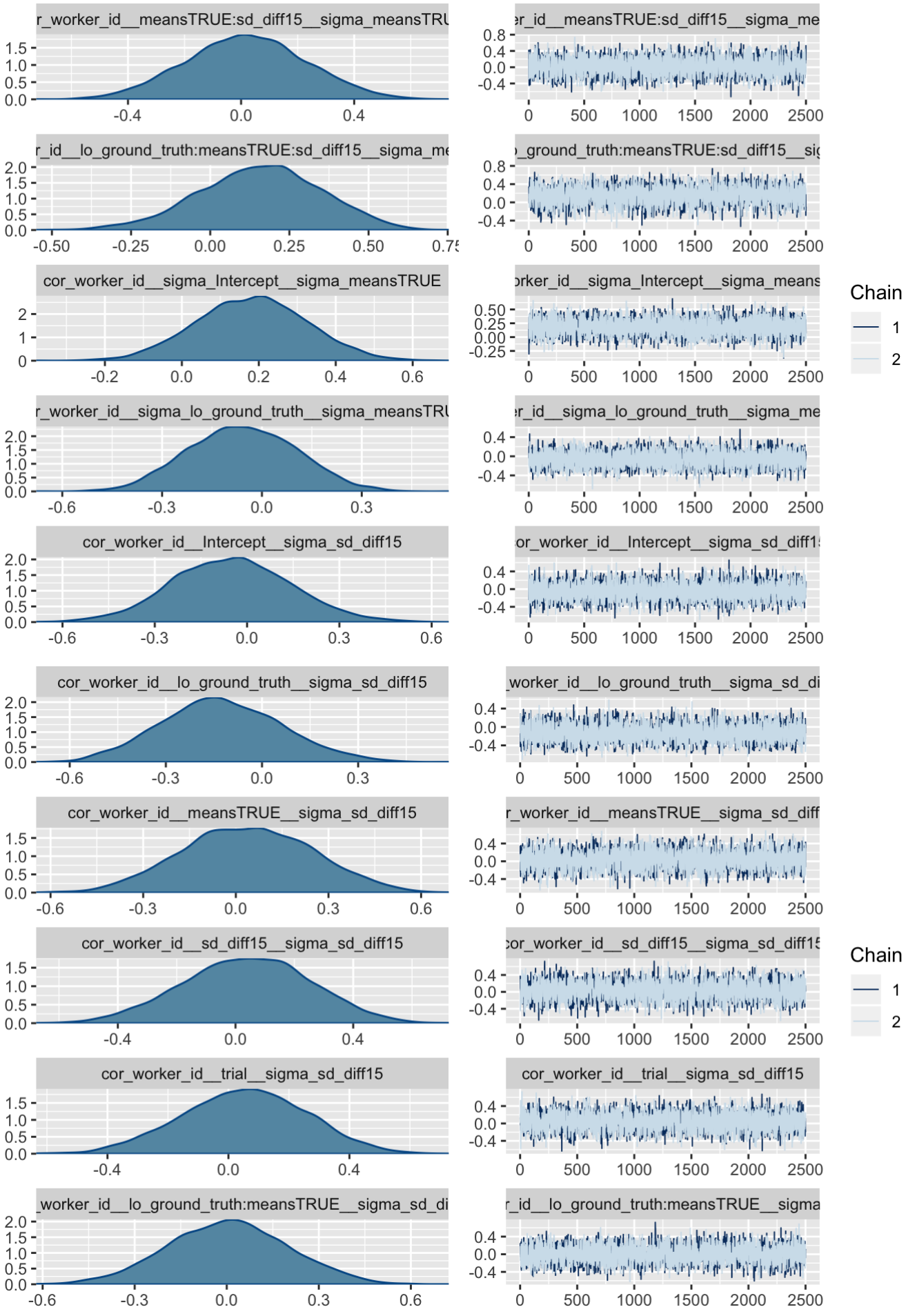

cor_worker_id__meansTRUE__lo_ground_truth:meansTRUE
cor_worker_id__sd_diff15__lo_ground_truth:meansTRUE
cor_worker_id__trial__lo_ground_truth:meansTRUE
cor_worker_id__Intercept__lo_ground_truth:sd_diff15
cor_worker_id__lo_ground_truth__lo_ground_truth:sd_diff1
cor_worker_id__meansTRUE__lo_ground_truth:sd_diff15
cor_worker_id__sd_diff15__lo_ground_truth:sd_diff15
cor_worker_id__trial__lo_ground_truth:sd_diff15
cor_worker_id__Intercept__meansTRUE:sd_diff15
Chain
1
2

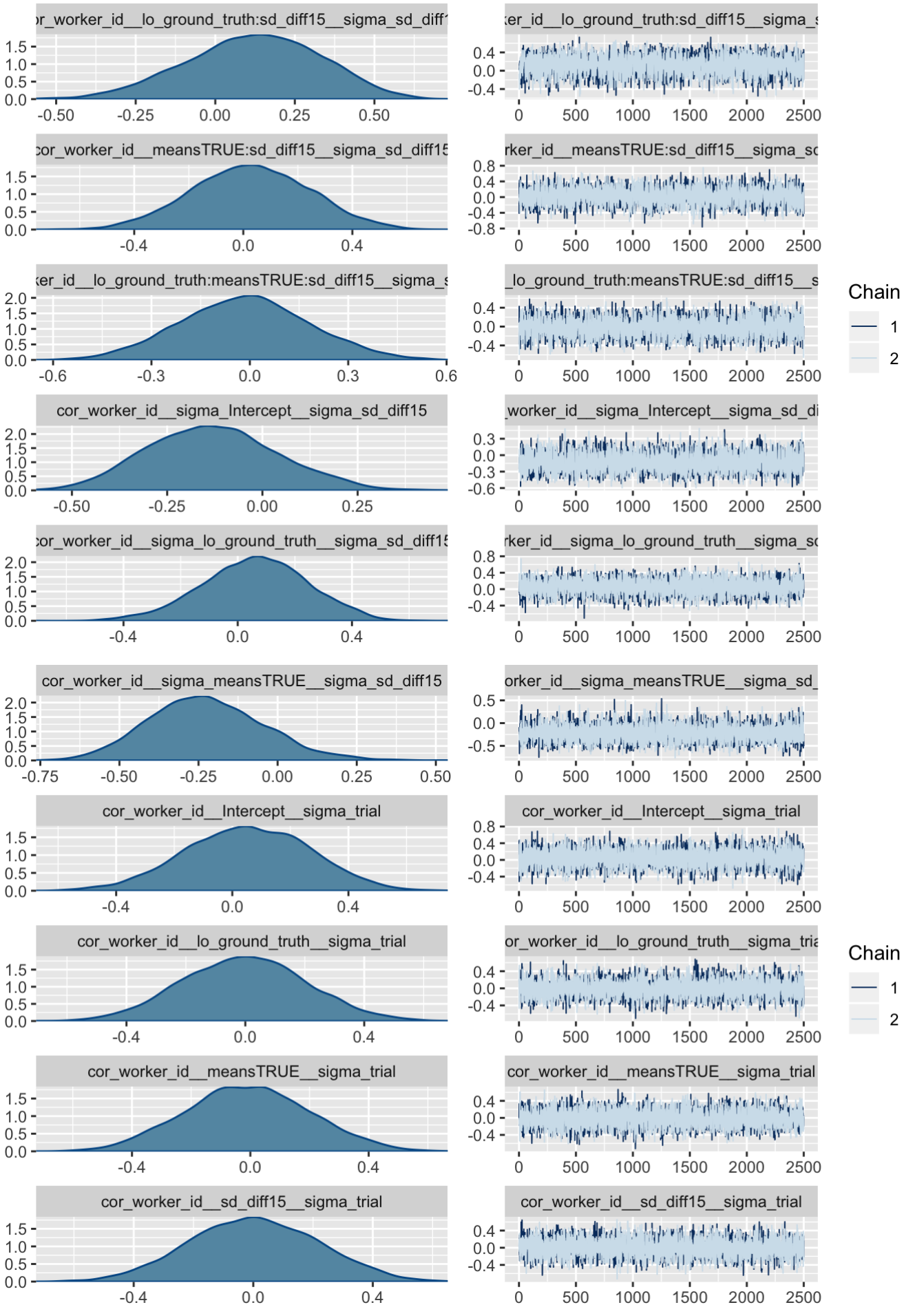

cor_worker_id__lo_ground_truth__meansTRUE:sd_diff15
cor_worker_id__meansTRUE__meansTRUE:sd_diff15
cor_worker_id__sd_diff15__meansTRUE:sd_diff15
cor_worker_id__trial__meansTRUE:sd_diff15
rker_id__lo_ground_truth:meansTRUE__meansTRUE:sd
worker_id__lo_ground_truth:sd_diff15__meansTRUE:sd_
worker_id__Intercept__lo_ground_truth:meansTRUE:sd_
ker_id__lo_ground_truth__lo_ground_truth:meansTRUE:s
rker_id__meansTRUE__lo_ground_truth:meansTRUE:sd
worker_id__sd_diff15__lo_ground_truth:meansTRUE:sd_
Chain
1
2

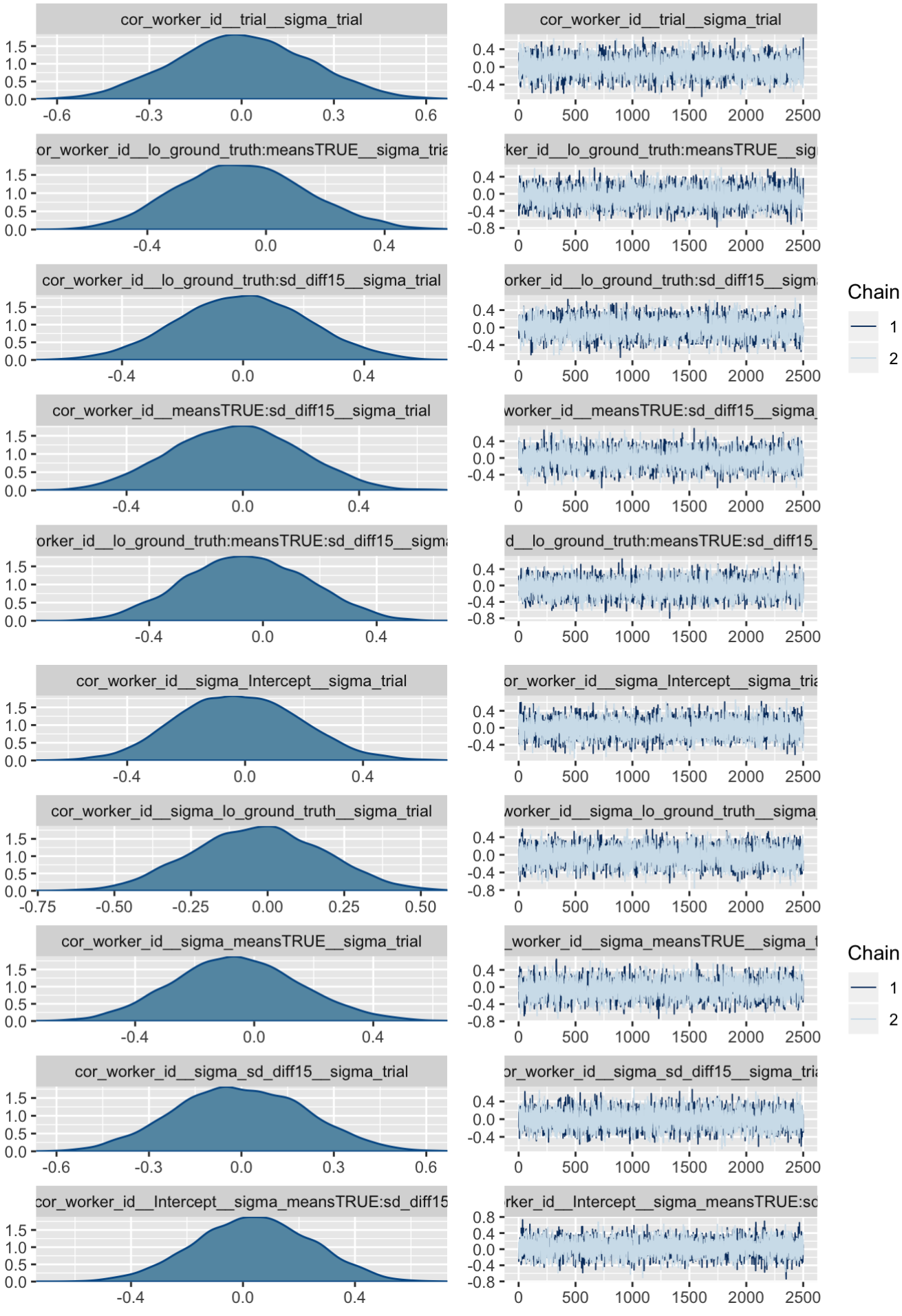

r_worker_id__trial__lo_ground_truth:meansTRUE:sd_diff
er_id__trial__lo_ground_truth:meansTRUE:
_lo_ground_truth:meansTRUE__lo_ground_truth:meansT
ound_truth:meansTRUE__lo_ground_truth:r
d__lo_ground_truth:sd_diff15__lo_ground_truth:meansTR
ground_truth:sd_diff15__lo_ground_truth:me
_id__meansTRUE:sd_diff15__lo_ground_truth:meansTRU
eansTRUE:sd_diff15__lo_ground_truth:mea
cor_worker_id__Intercept__sigma_Intercept
cor_worker_id__lo_ground_truth__sigma_Intercept
worker_id__lo_ground_truth__sigma_Inter
cor_worker_id__meansTRUE__sigma_Intercept
_worker_id__meansTRUE__sigma_Interc
cor_worker_id__sd_diff15__sigma_Intercept
or_worker_id__sd_diff15__sigma_Intercep
cor_worker_id__trial__sigma_Intercept
_worker_id__lo_ground_truth:meansTRUE__sigma_Interc
_id__lo_ground_truth:meansTRUE__sigma
Chain
1
2

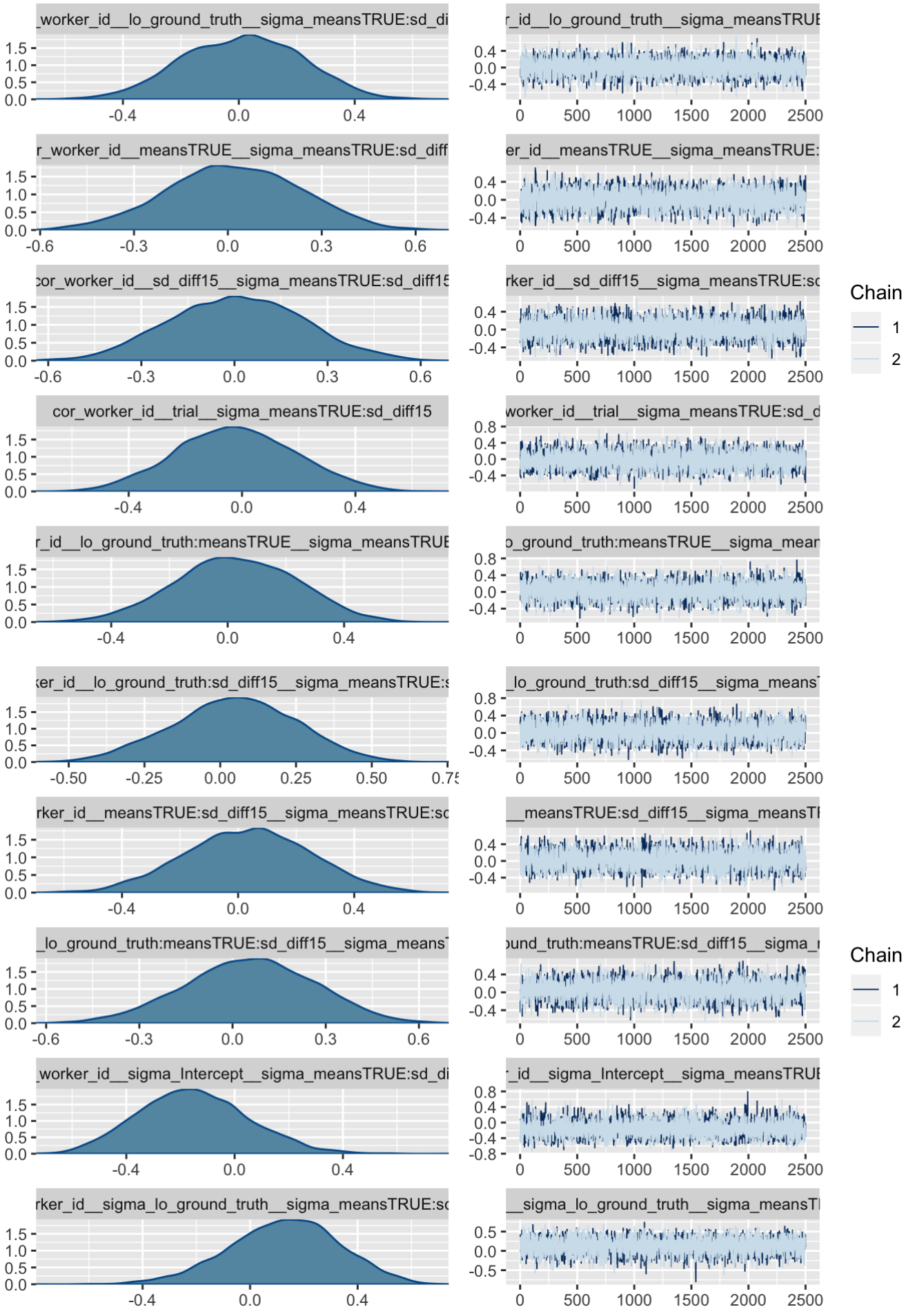

cor_worker_id__lo_ground_truth:sd_diff15__sigma_Intercept
cor_worker_id__meansTRUE:sd_diff15__sigma_Intercept
cor_worker_id__lo_ground_truth:meansTRUE:sd_diff15__sigma_Intercept
cor_worker_id__Intercept__sigma_lo_ground_truth
cor_worker_id__lo_ground_truth__sigma_lo_ground_truth
cor_worker_id__meansTRUE__sigma_lo_ground_truth
cor_worker_id__sd_diff15__sigma_lo_ground_truth
cor_worker_id__trial__sigma_lo_ground_truth
cor_worker_id__lo_ground_truth:meansTRUE__sigma_lo_ground_truth
worker_id__lo_ground_truth:sd_diff15__sigma_lo_ground_truth
Chain
1
2

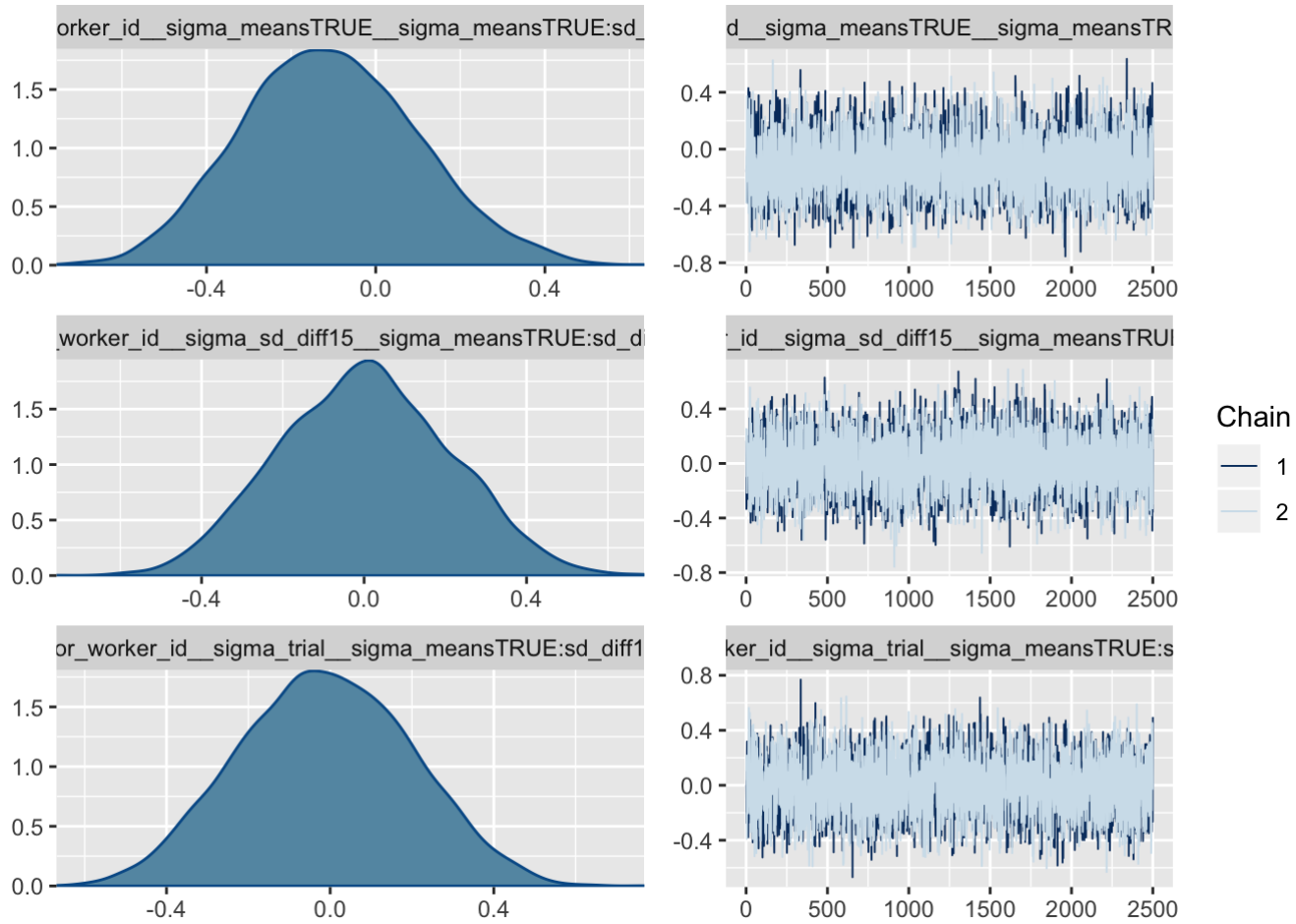

_worker_id__meansTRUE:sd_diff15__sigma_lo_ground_t
r_id__meansTRUE:sd_diff15__sigma_lo_gr
id__lo_ground_truth:meansTRUE:sd_diff15__sigma_lo_g
_ground_truth:meansTRUE:sd_diff15__sigm
cor_worker_id__sigma_Intercept__sigma_lo_ground_truth
rker_id__sigma_Intercept__sigma_lo_groun
cor_worker_id__Intercept__sigma_meansTRUE
r_worker_id__Intercept__sigma_meansTRU
cor_worker_id__lo_ground_truth__sigma_meansTRUE
orker_id__lo_ground_truth__sigma_means
cor_worker_id__meansTRUE__sigma_meansTRUE
worker_id__meansTRUE__sigma_meansTR
cor_worker_id__sd_diff15__sigma_meansTRUE
r_worker_id__sd_diff15__sigma_meansTRU
cor_worker_id__trial__sigma_meansTRUE
cor_worker_id__trial__sigma_meansTRUE
orker_id__lo_ground_truth:meansTRUE__sigma_means
id__lo_ground_truth:meansTRUE__sigma_
_worker_id__lo_ground_truth:sd_diff15__sigma_meansTR
r_id__lo_ground_truth:sd_diff15__sigma_me
Chain
1
2
Chain
1
2


r_worker_id__meansTRUE:sd_diff15__sigma_meansTRU
r_id__lo_ground_truth:meansTRUE:sd_diff15__sigma_me
cor_worker_id__sigma_Intercept__sigma_meansTRUE
r_worker_id__sigma_lo_ground_truth__sigma_meansTRU
cor_worker_id__Intercept__sigma_sd_diff15
cor_worker_id__lo_ground_truth__sigma_sd_diff15
cor_worker_id__meansTRUE__sigma_sd_diff15
cor_worker_id__sd_diff15__sigma_sd_diff15
cor_worker_id__trial__sigma_sd_diff15
_worker_id__lo_ground_truth:meansTRUE__sigma_sd_di
Chain
1
2


cor_worker_id__lo_ground_truth:sd_diff15__sigma_sd_diff
cor_worker_id__meansTRUE:sd_diff15__sigma_sd_diff15
cor_worker_id__lo_ground_truth:meansTRUE:sd_diff15__sigma_s
cor_worker_id__sigma_Intercept__sigma_sd_diff15
cor_worker_id__sigma_lo_ground_truth__sigma_sd_diff15
cor_worker_id__sigma_meansTRUE__sigma_sd_diff15
cor_worker_id__Intercept__sigma_trial
cor_worker_id__lo_ground_truth__sigma_trial
cor_worker_id__meansTRUE__sigma_trial
cor_worker_id__sd_diff15__sigma_trial
Chain
1
2


cor_worker_id__trial__sigma_trial
cor_worker_id__lo_ground_truth:meansTRUE__sigma_tria
cor_worker_id__lo_ground_truth:sd_diff15__sigma_trial
cor_worker_id__meansTRUE:sd_diff15__sigma_trial
orker_id__lo_ground_truth:meansTRUE:sd_diff15__sigma
cor_worker_id__sigma_Intercept__sigma_trial
cor_worker_id__sigma_lo_ground_truth__sigma_trial
cor_worker_id__sigma_meansTRUE__sigma_trial
cor_worker_id__sigma_sd_diff15__sigma_trial
cor_worker_id__Intercept__sigma_meansTRUE:sd_diff15
Chain
1
2


_worker_id__lo_ground_truth__sigma_meansTRUE:sd_di
r_id__lo_ground_truth__sigma_meansTRUE
r_worker_id__meansTRUE__sigma_meansTRUE:sd_diff
er_id__meansTRUE__sigma_meansTRUE:
cor_worker_id__sd_diff15__sigma_meansTRUE:sd_diff15
rker_id__sd_diff15__sigma_meansTRUE:sc
cor_worker_id__trial__sigma_meansTRUE:sd_diff15
worker_id__trial__sigma_meansTRUE:sd_d
r_id__lo_ground_truth:meansTRUE__sigma_meansTRUE
o_ground_truth:meansTRUE__sigma_mean
ker_id__lo_ground_truth:sd_diff15__sigma_meansTRUE:s
_lo_ground_truth:sd_diff15__sigma_means
rker_id__meansTRUE:sd_diff15__sigma_meansTRUE:sc
__meansTRUE:sd_diff15__sigma_meansTR
_lo_ground_truth:meansTRUE:sd_diff15__sigma_means
ound_truth:meansTRUE:sd_diff15__sigma_
_worker_id__sigma_Intercept__sigma_meansTRUE:sd_di
_id__sigma_Intercept__sigma_meansTRUE
rker_id__sigma_lo_ground_truth__sigma_meansTRUE:sc
__sigma_lo_ground_truth__sigma_meansTR
Chain
1
2
Chain
1
2



Model Comparison

Let's see how this maximal model compares with our previous model.

```
waic(m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.trial.sigma.gt.means.sd.trial, m.max)
```

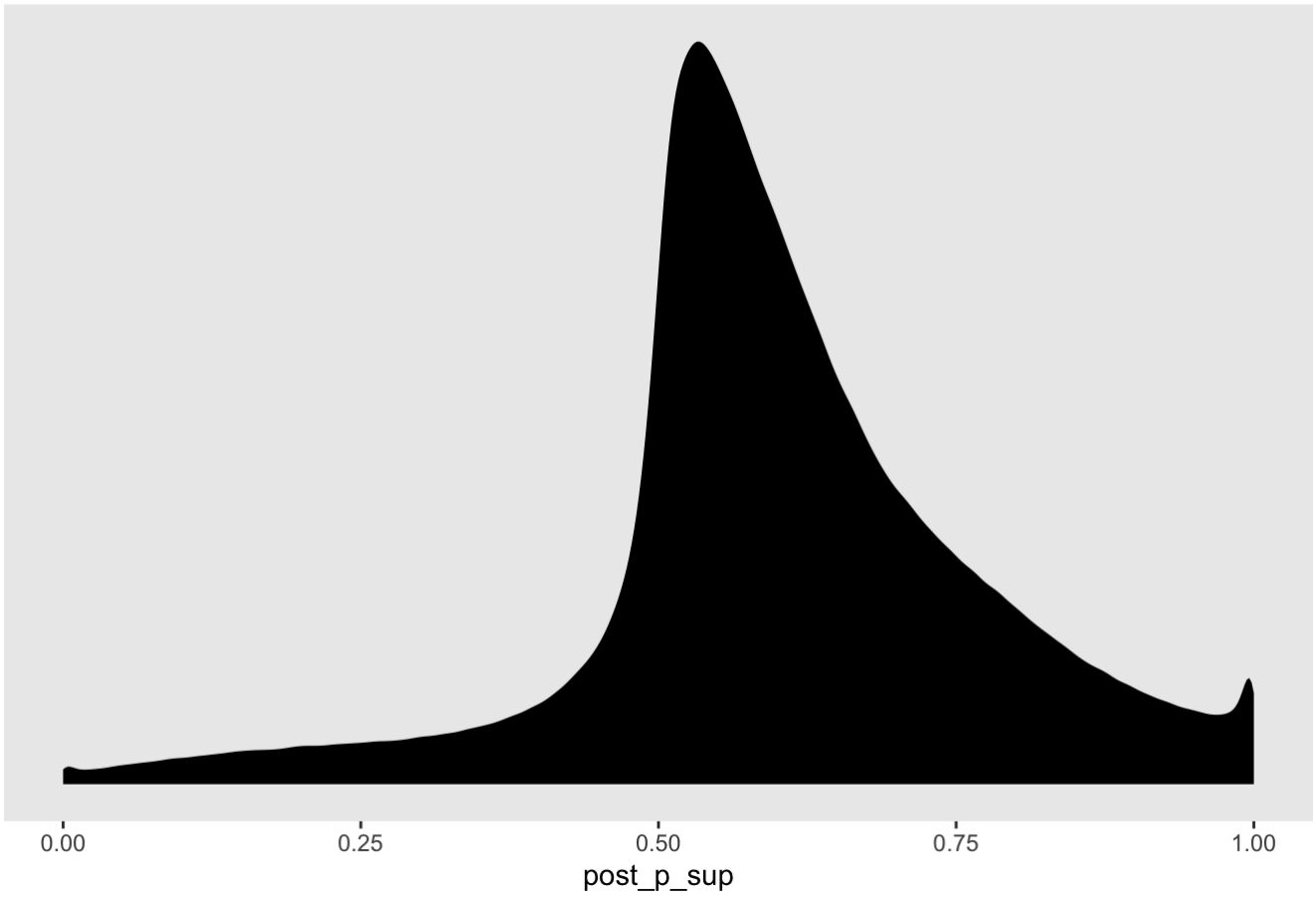
```
##                                                                          WAIC
## m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.trial.sigma.gt.means.sd.trial 489.70
## m.max                                                                  480.65
## m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.trial.sigma.gt.means.sd.trial - m.max   9.05
##                                                                           SE
## m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.trial.sigma.gt.means.sd.trial 77.50
## m.max                                                                  77.16
## m.wrkr.means.sd.vis.llo_p_sup.r.means.sd.trial.sigma.gt.means.sd.trial - m.max  7.11
```

It looks like adding predictors for block order helps marginally.

Predictive Checks

Let's check our posterior predictive distribution.

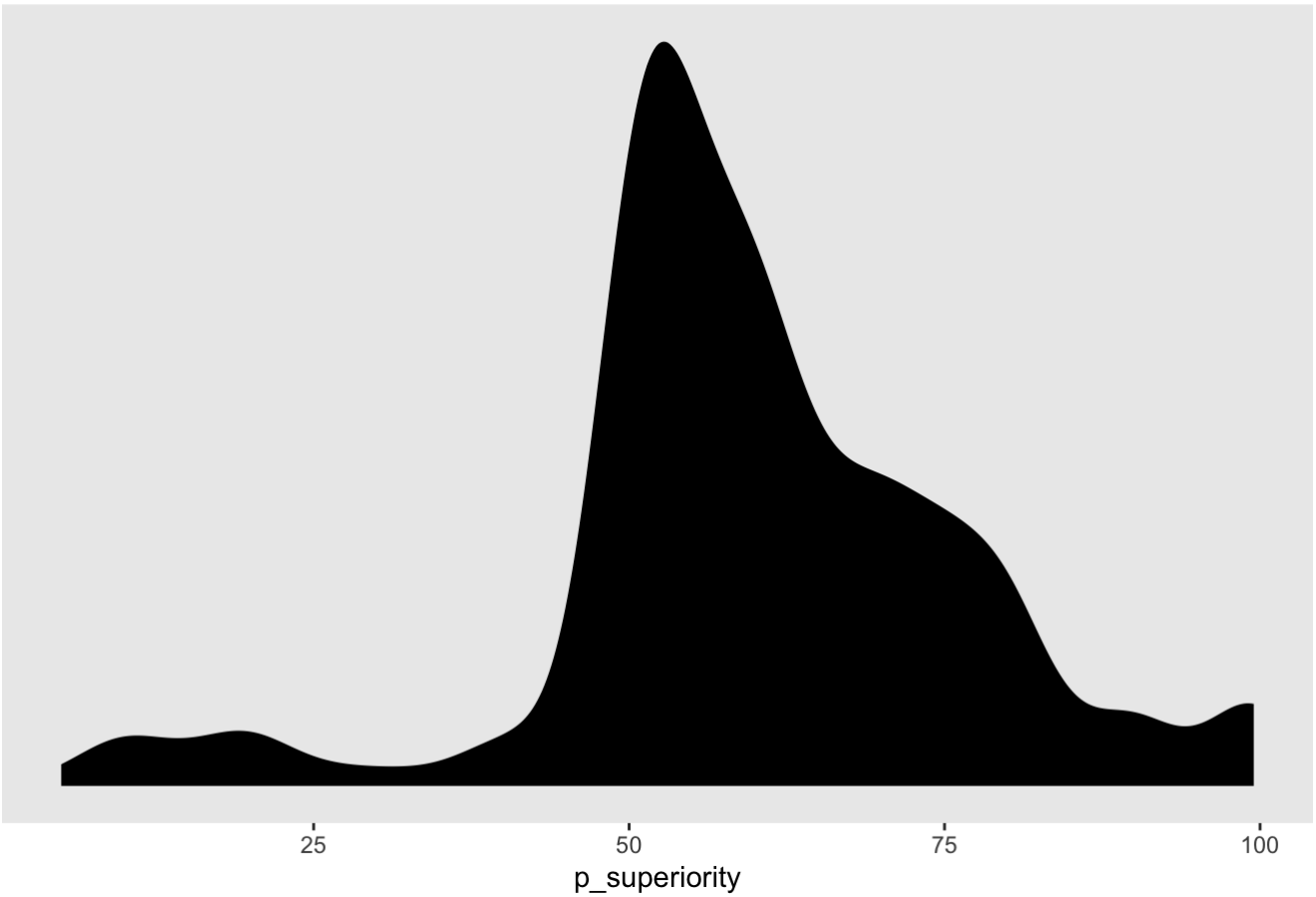
```
# posterior predictive check
model_df %>%
  select(lo_ground_truth, worker_id, means, sd_diff, condition, trial, start_means) %>%
  add_predicted_draws(m.max, prediction = "lo_p_sup", seed = 1234) %>%
  mutate(
    # transform to probability units
    post_p_sup = plogis(lo_p_sup)
  ) %>%
  ggplot(aes(x = post_p_sup)) +
  geom_density(fill = "black", size = 0) +
  scale_y_continuous(NULL, breaks = NULL) +
  labs(subtitle = "Posterior predictive distribution for probability of superiority") +
  theme(panel.grid = element_blank())
```



How do these predictions compare to the observed data?

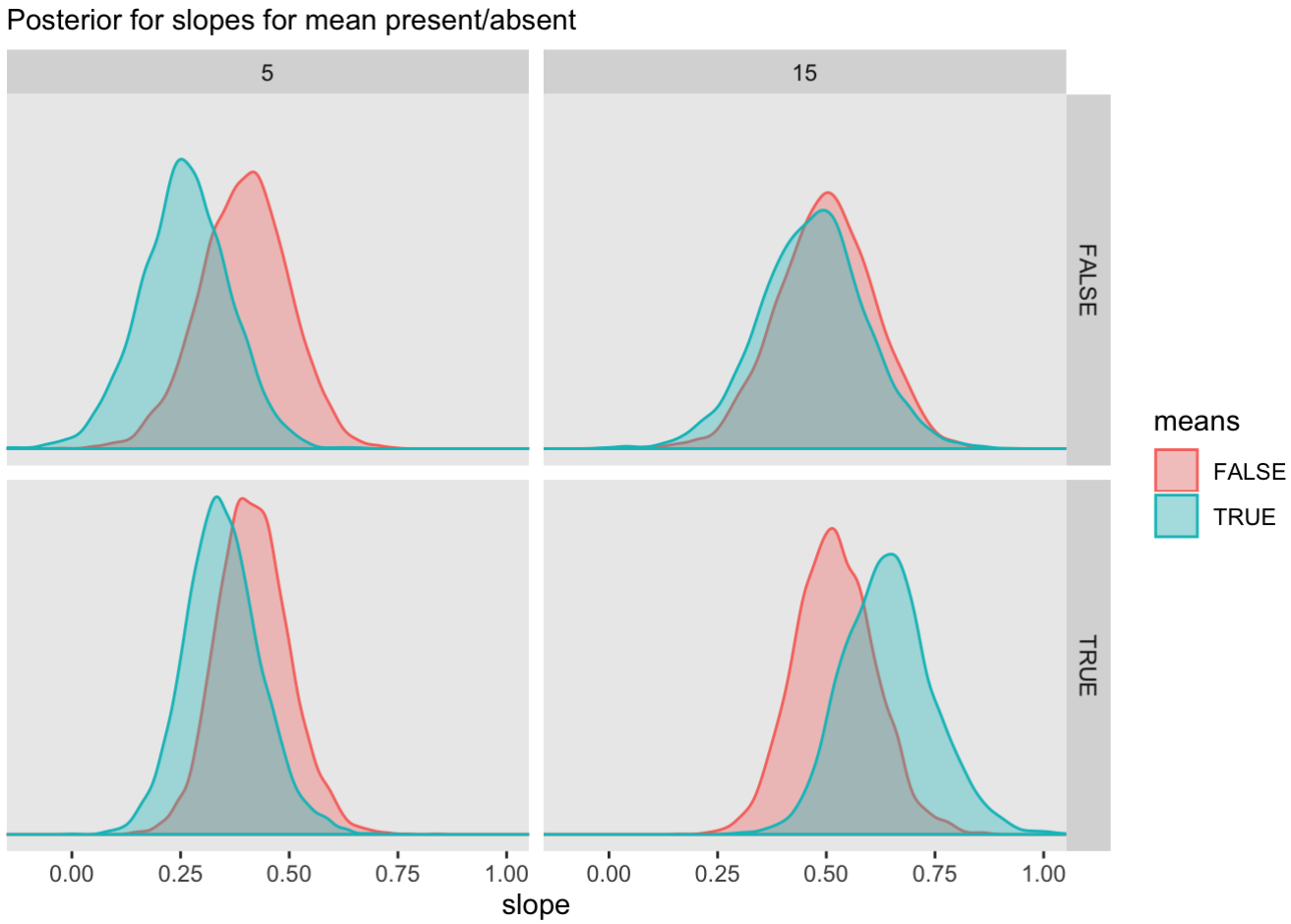
```
# data density
model_df %>%
  ggplot(aes(x = p_superiority)) +
  geom_density(fill = "black", size = 0) +
  scale_y_continuous(NULL, breaks = NULL) +
  labs(subtitle = "Data distribution for probability of superiority") +
  theme(panel.grid = element_blank())
```

Data distribution for probability of superiority



What does the posterior for the slope look like when means are present vs absent? We'll split this based on uncertainty shown and block order (marginalizing across visualization conditions) to see if there is a difference in the effect of extrinsic means per block.

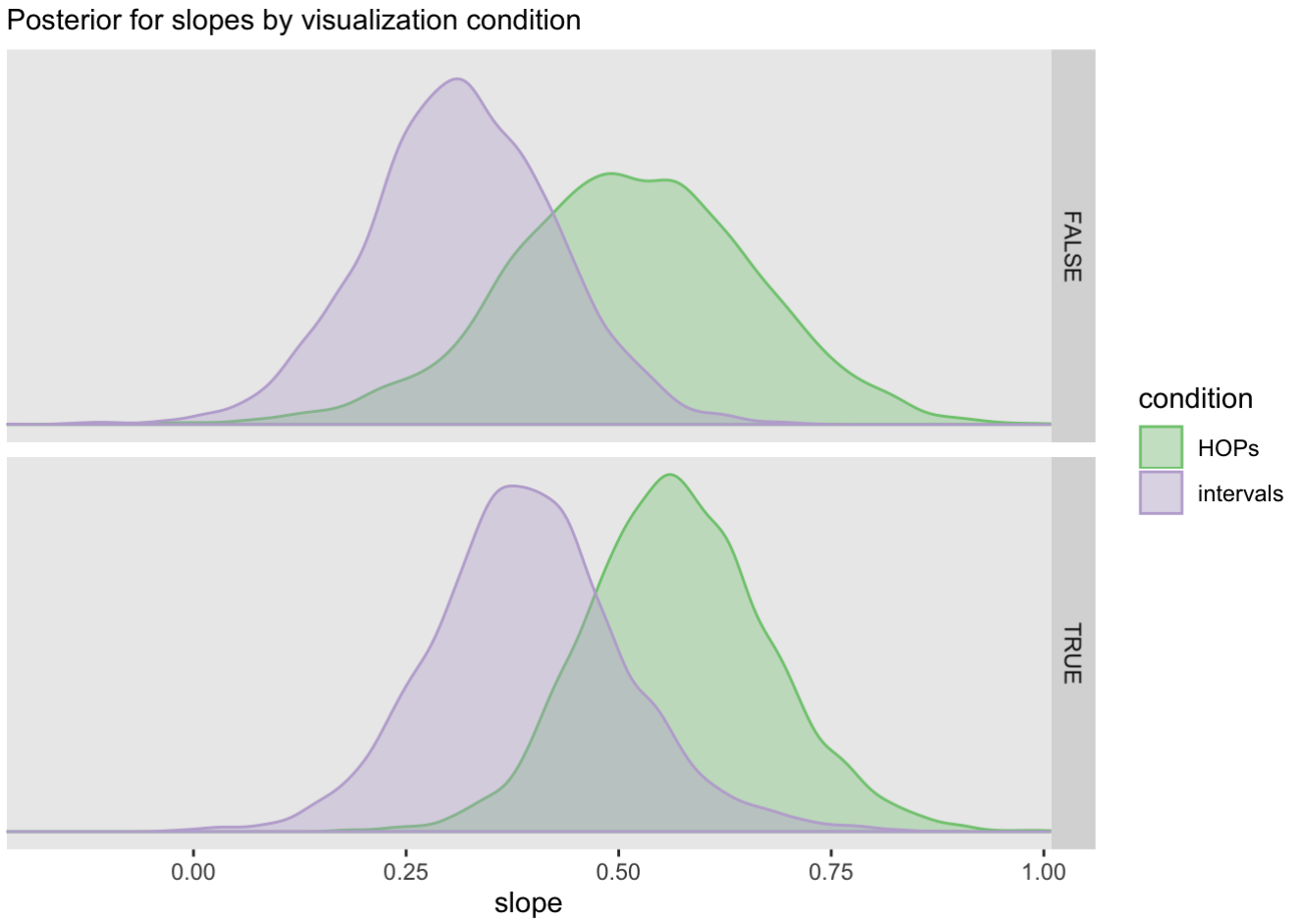
```
model_df %>%
  group_by(means, sd_diff, condition, trial, start_means) %>%
  data_grid(lo_ground_truth = c(0, 1)) %>%          # get fitted draws (in log odds unit
s) only for ground truth of 0 and 1
  add_fitted_draws(m.max, re_formula = NA) %>%
  compare_levels(.value, by = lo_ground_truth) %>%  # calculate the difference between f
its at 1 and 0 (i.e., slope)
  rename(slope = .value) %>%
  group_by(means, sd_diff, start_means, .draw) %>%  # group by predictors to keep
  summarise(slope = weighted.mean(slope)) %>%       # marginalize out visualization cond
ition by taking a weighted average
  ggplot(aes(x = slope, group = means, color = means, fill = means)) +
  geom_density(alpha = 0.35) +
  scale_x_continuous(expression(slope), expand = c(0, 0)) +
  scale_y_continuous(NULL, breaks = NULL) +
  labs(subtitle = "Posterior for slopes for mean present/absent") +
  theme(panel.grid = element_blank()) +
  facet_grid(start_means ~ sd_diff)
```



This effect suggests that adding means only helps in when they are used in the first block of trials.

What does the posterior for the slope in each visualization condition look like, marginalizing across the presence/absence of the mean? Again, we'll facet by block order.

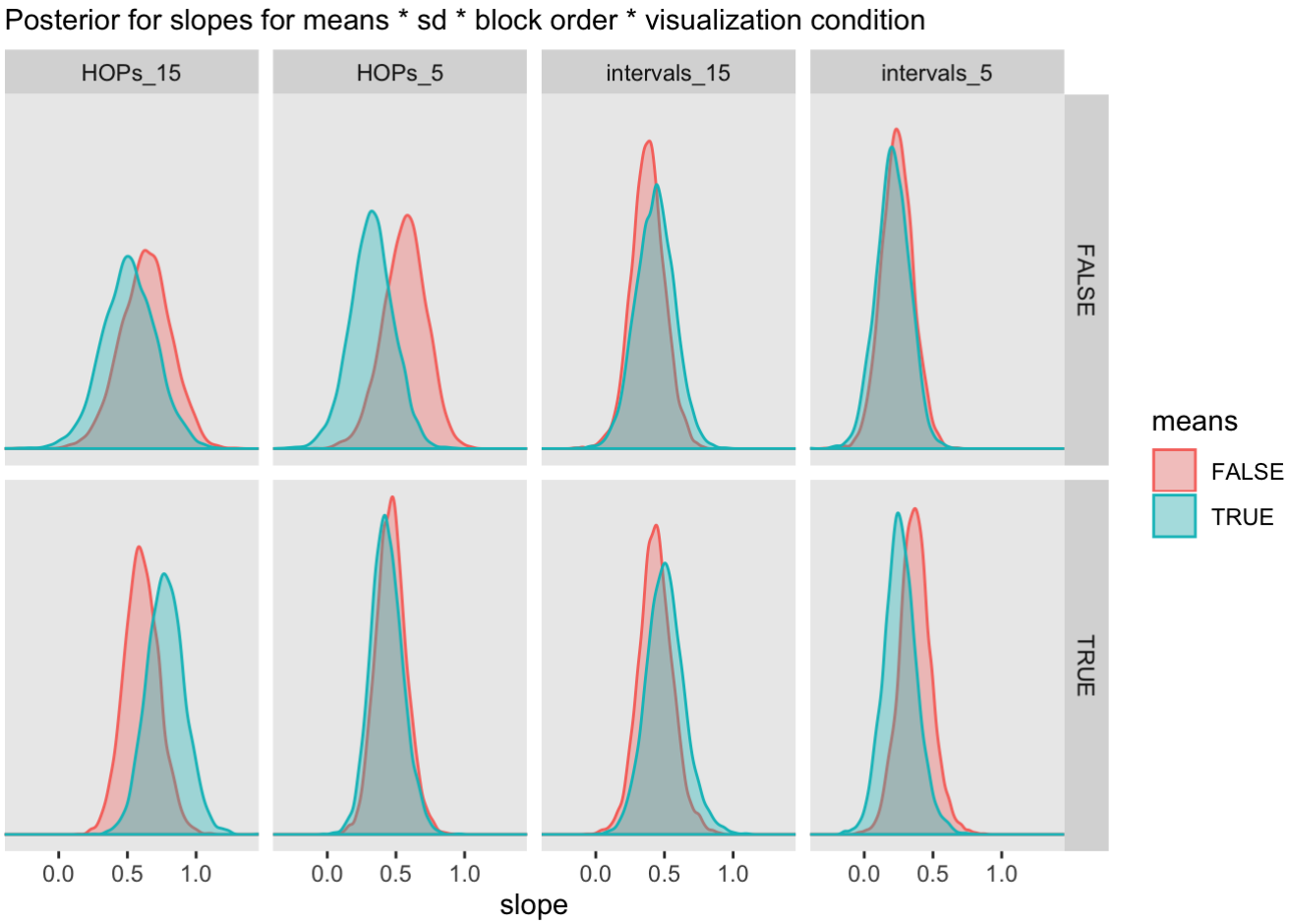
```
model_df %>%
  group_by(means, sd_diff, condition, trial, start_means) %>%
  data_grid(lo_ground_truth = c(0, 1)) %>%          # get fitted draws (in log odds units) only for ground truth of 0 and 1
  add_fitted_draws(m.max, re_formula = NA) %>%
  compare_levels(.value, by = lo_ground_truth) %>%  # calculate the difference between fits at 1 and 0 (i.e., slope)
  rename(slope = .value) %>%
  group_by(condition, start_means, .draw) %>%       # group by predictors to keep
  summarise(slope = weighted.mean(slope)) %>%       # marginalize out means present/absent by taking a weighted average
  ggplot(aes(x = slope, group = condition, color = condition, fill = condition)) +
  geom_density(alpha = 0.35) +
  scale_fill_brewer(type = "qual", palette = 1) +
  scale_color_brewer(type = "qual", palette = 1) +
  scale_x_continuous(expression(slope), expand = c(0, 0)) +
  scale_y_continuous(NULL, breaks = NULL) +
  labs(subtitle = "Posterior for slopes by visualization condition") +
  theme(panel.grid = element_blank()) +
  facet_grid(start_means ~ .)
```



The difference between visualization conditions is stable regardless of block order.

What if we break these marginal effects down into simple effects for the interaction of the presence/absence of the mean, uncertainty shown, block order, and visualization condition?

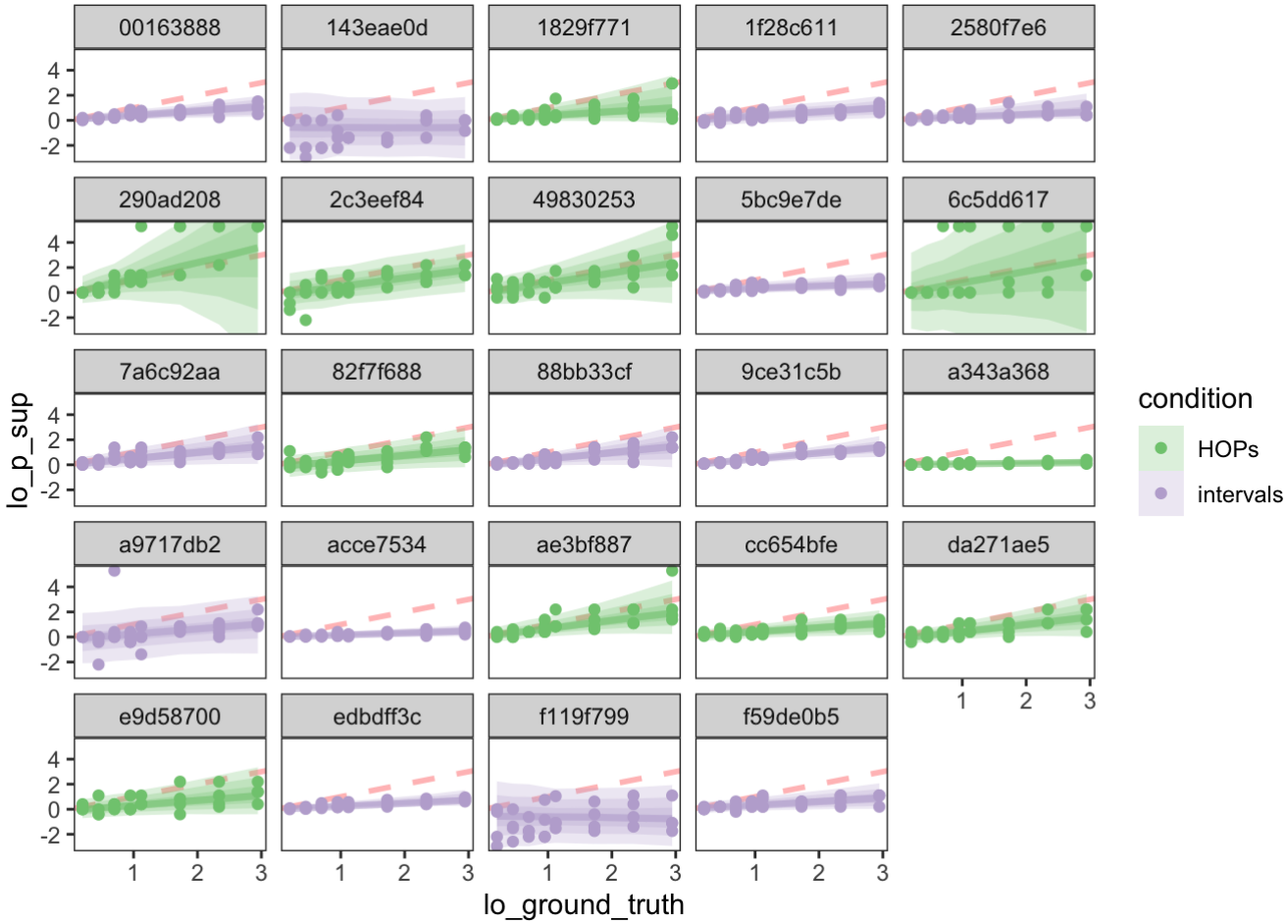
```
model_df %>%
  group_by(means, sd_diff, condition, trial, start_means) %>%
  data_grid(lo_ground_truth = c(0, 1)) %>%                      # get fitted draws (in l
og odds units) only for ground truth of 0 and 1
  add_fitted_draws(m.max, re_formula = NA) %>%
  compare_levels(.value, by = lo_ground_truth) %>%              # calculate the differen
ce between fits at 1 and 0 (i.e., slope)
  rename(slope = .value) %>%
  group_by(means, sd_diff, condition, start_means, .draw) %>%   # group by predictors to
keep
  summarise(slope = weighted.mean(slope)) %>%                   # marginalize out means
 present/absent by taking a weighted average
  unite(vis_cond, condition, sd_diff, remove = FALSE) %>%
  ggplot(aes(x = slope, group = means, color = means, fill = means)) +
  geom_density(alpha = 0.35) +
  scale_x_continuous(expression(slope), expand = c(0, 0)) +
  scale_y_continuous(NULL, breaks = NULL) +
  labs(subtitle = "Posterior for slopes for means * sd * block order * visualization con
dition") +
  theme(panel.grid = element_blank()) +
  facet_grid(start_means ~ vis_cond)
```



The effect of the mean seems to be the most reliable in the intervals condition, especially when the mean is added in the second block.

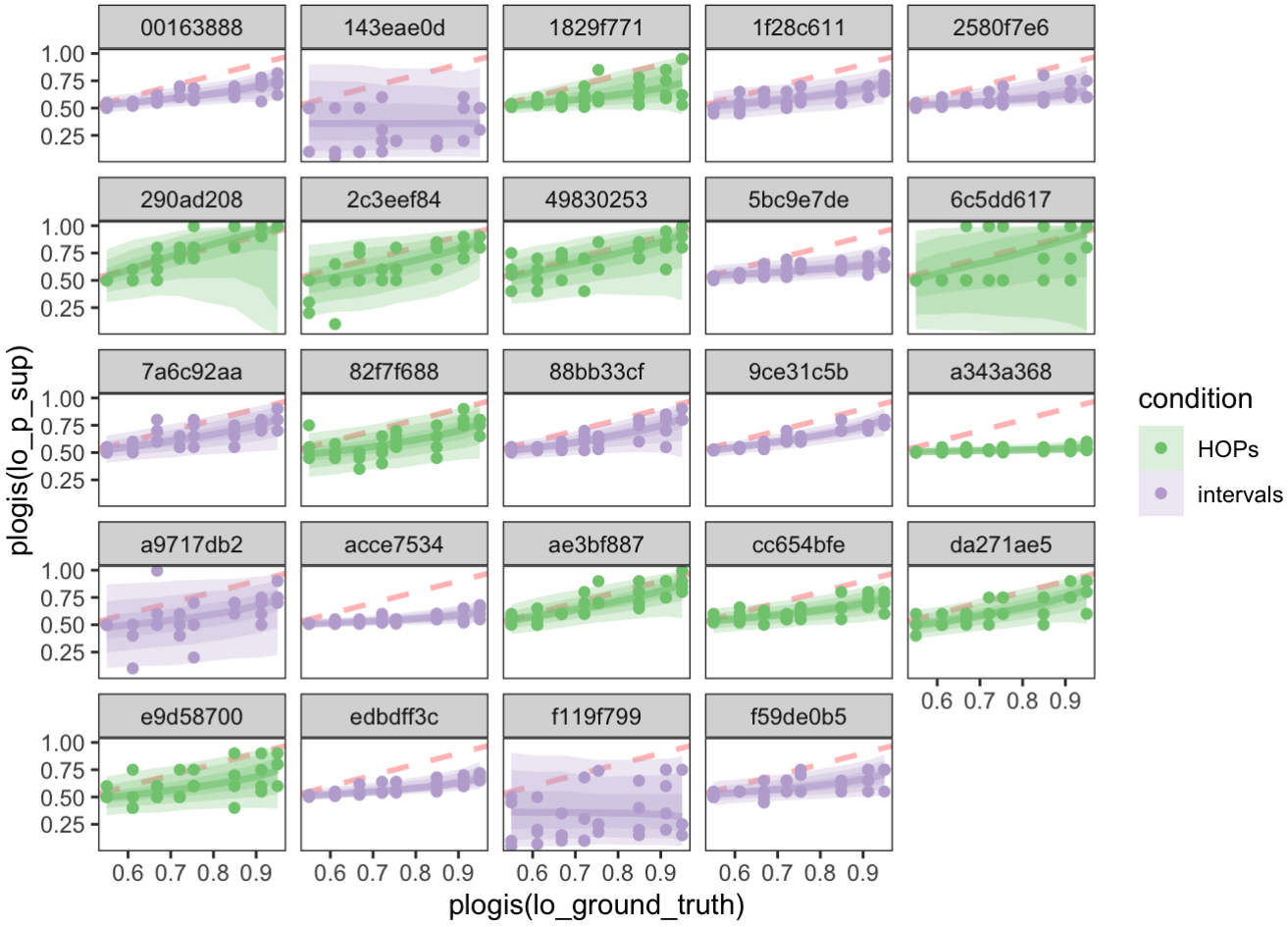
Let's take a look at predictions per worker and visualization condition to get a more granular sense of our model fit.

```
model_df %>%
  group_by(lo_ground_truth, worker_id, means, sd_diff, condition, trial, start_means) %
>%
  add_predicted_draws(m.max) %>%
  ggplot(aes(x = lo_ground_truth, y = lo_p_sup, color = condition, fill = condition)) +
  geom_abline(intercept = 0, slope = 1, size = 1, alpha = .3, color = "red", linetype =
"dashed") + # ground truth
  stat_lineribbon(aes(y = .prediction), .width = c(.95, .80, .50), alpha = .25) +
  geom_point(data = model_df) +
  scale_fill_brewer(type = "qual", palette = 1) +
  scale_color_brewer(type = "qual", palette = 1) +
  coord_cartesian(xlim = quantile(model_df$lo_ground_truth, c(0, 1)),
                  ylim = quantile(model_df$lo_p_sup, c(0, 1))) +
  theme_bw() +
  theme(panel.grid = element_blank()) +
  facet_wrap(~ worker_id)
```



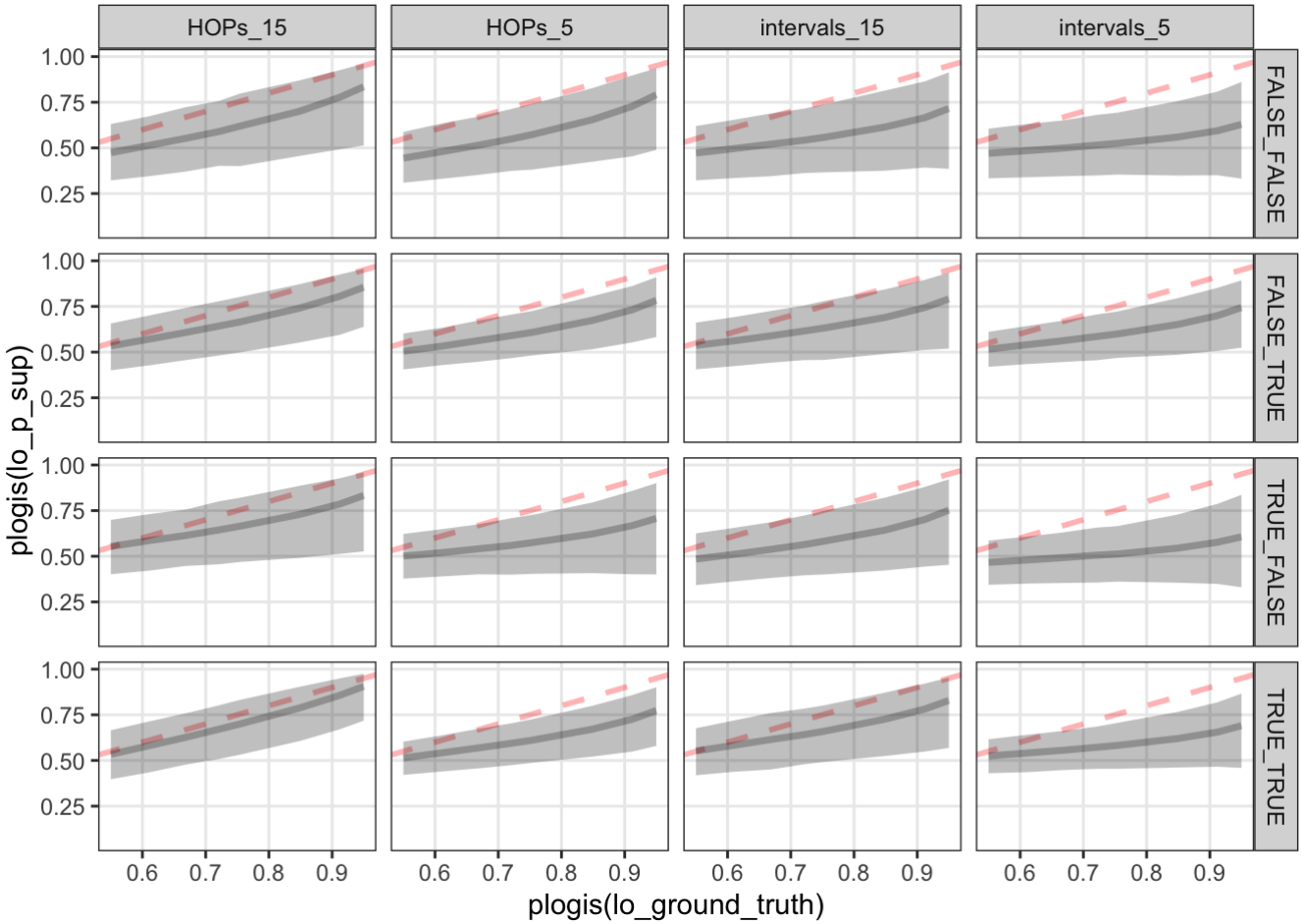
What does this look like in probability units?

```
model_df %>%
  group_by(lo_ground_truth, worker_id, means, sd_diff, condition, trial, start_means) %
>%
  add_predicted_draws(m.max) %>%
  ggplot(aes(x = plogis(lo_ground_truth), y = plogis(lo_p_sup), color = condition, fill
= condition)) +
  geom_abline(intercept = 0, slope = 1, size = 1, alpha = .3, color = "red", linetype =
"dashed") + # ground truth
  stat_lineribbon(aes(y = plogis(.prediction)), .width = c(.95, .80, .50), alpha = .25)
 +
  geom_point(data = model_df) +
  scale_fill_brewer(type = "qual", palette = 1) +
  scale_color_brewer(type = "qual", palette = 1) +
  coord_cartesian(xlim = quantile(plogis(model_df$lo_ground_truth), c(0, 1)),
                  ylim = quantile(plogis(model_df$lo_p_sup), c(0, 1))) +
  theme_bw() +
  theme(panel.grid = element_blank()) +
  facet_wrap(~ worker_id)
```



Perhaps the best way to understand predictions from this model is to plot posterior predictions for each condition, instead of just slopes. Let's take a look at predictions per condition in probability units for the average worker.

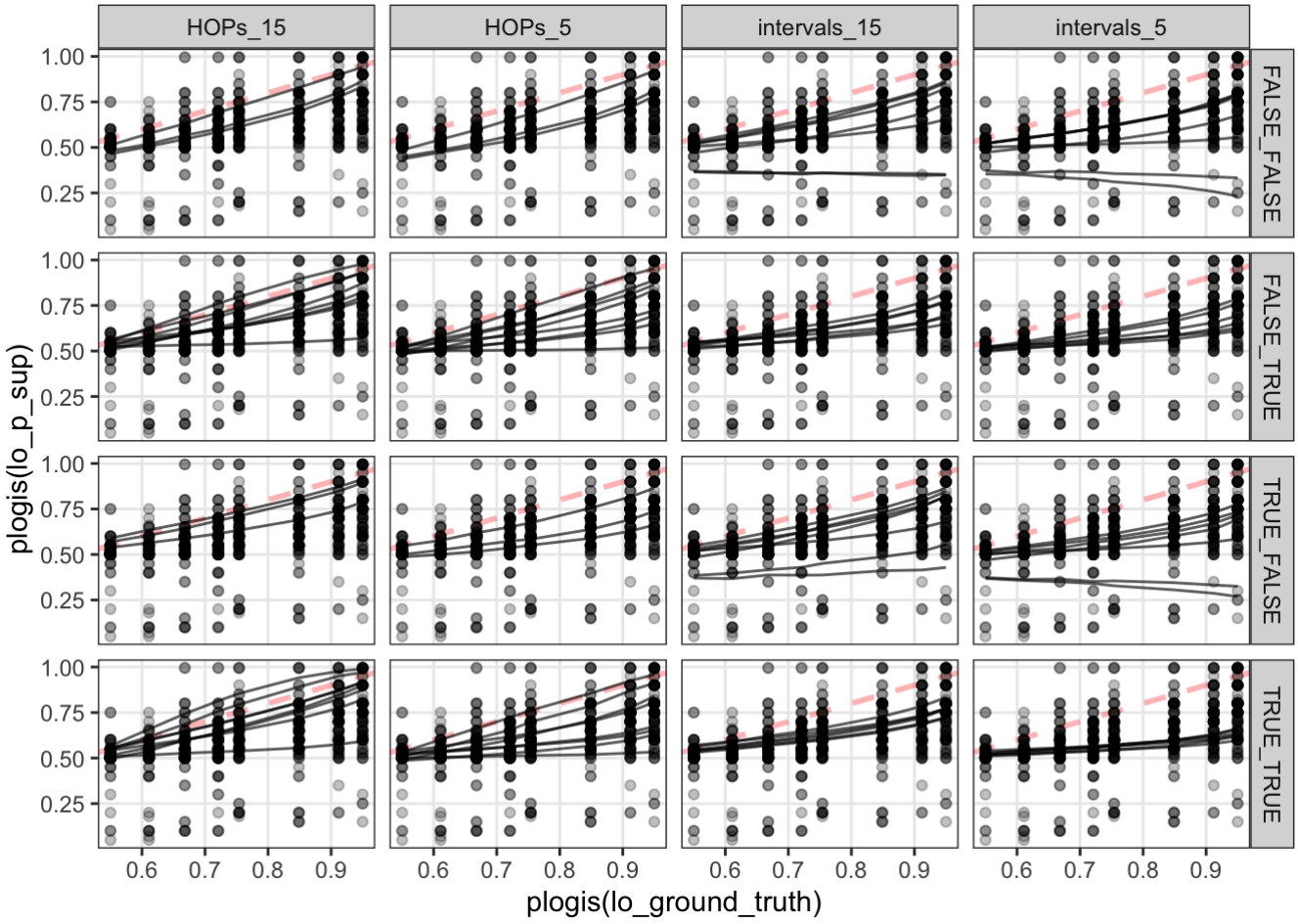
```
model_df %>%
  group_by(lo_ground_truth, means, sd_diff, condition, trial, start_means) %>%
  add_predicted_draws(m.max, re_formula = NA) %>%
  unite(vis_cond, condition, sd_diff) %>%
  unite(means_present, means, start_means) %>%
  ggplot(aes(x = plogis(lo_ground_truth), y = plogis(lo_p_sup))) +
  geom_abline(intercept = 0, slope = 1, size = 1, alpha = .3, color = "red", linetype =
"dashed") + # ground truth
  stat_lineribbon(aes(y = plogis(.prediction)), .width = c(.95), alpha = .25, fill = "bl
ack", show.legend = FALSE) +
  coord_cartesian(xlim = quantile(plogis(model_df$lo_ground_truth), c(0, 1)),
                  ylim = quantile(plogis(model_df$lo_p_sup), c(0, 1))) +
  theme_bw() +
  theme(panel.grid.minor = element_blank()) +
  facet_grid(means_present ~ vis_cond)
```



We can see that intervals lead to less biased perceptions, especially when means are added in the second block.

Let's also look at a spaghetti plot of average predictions per worker.

```
model_df %>%
  add_predicted_draws(m.max) %>%
  group_by(lo_ground_truth, worker_id, means, sd_diff, condition, start_means) %>% # mar
ginalize over trial order
  summarize(avg_pred = weighted.mean(.prediction)) %>%
  unite(vis_cond, condition, sd_diff) %>%
  unite(means_present, means, start_means) %>%
  ggplot(aes(x = plogis(lo_ground_truth), y = plogis(lo_p_sup), group = worker_id)) +
  geom_abline(intercept = 0, slope = 1, size = 1, alpha = .3, color = "red", linetype =
"dashed") + # ground truth
  geom_line(aes(y = plogis(avg_pred)), alpha = .65) +
  geom_point(data = model_df, alpha = 0.25) +
  coord_cartesian(xlim = quantile(plogis(model_df$lo_ground_truth), c(0, 1)),
                  ylim = quantile(plogis(model_df$lo_p_sup), c(0, 1))) +
  theme_bw() +
  theme(panel.grid.minor = element_blank()) +
  facet_grid(means_present ~ vis_cond)
```


HOPs_15
HOPs_5
intervals_15
intervals_5
FALSE_FALSE
FALSE_TRUE
TRUE_FALSE
TRUE_TRUE
plogis(lo_p_sup)
plogis(lo_ground_truth)
1.00
0.75
0.50
0.25
0.6
0.7
0.8
0.9