Pilot Analysis: Linear Log Odds Model of Probability of Superiority

In this document, we build a (non-censored) 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. With this approach a censored model may not be necessary. Since we removed the loss frame trials, 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(),
##   condition = col_character(),
##   start_means = col_character(),
##   numeracy = col_integer(),
##   gender = col_character(),
##   age = col_character(),
##   education = col_character(),
##   chart_use = col_character(),
##   intervene = col_integer(),
##   outcome = col_character(),
##   pSup = 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,
    p_award_with = pAwardWith,
    p_award_without = pAwardWithout,
    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") %>%
  # add a variable to note whether the chart they viewed showed means
  mutate(means = as.factor((start_means == "True" & as.numeric(trial) < 16) | (start_mea
ns == "False" & as.numeric(trial) >= 16)))

head(responses_df)
```

```
## # A tibble: 6 x 30
##   worker_id batch condition baseline award_value exchange start_means
##   <chr>     <int> <chr>        <dbl>       <dbl>    <dbl> <chr>
## 1 c3de4118      4 intervals      0.5        2.25      0.2 False
## 2 c3de4118      4 intervals      0.5        2.25      0.2 False
## 3 c3de4118      4 intervals      0.5        2.25      0.2 False
## 4 c3de4118      4 intervals      0.5        2.25      0.2 False
## 5 c3de4118      4 intervals      0.5        2.25      0.2 False
## 6 c3de4118      4 intervals      0.5        2.25      0.2 False
## # … with 23 more variables: total_bonus <dbl>, duration <dbl>,
## #   numeracy <int>, gender <chr>, age <chr>, education <chr>,
## #   chart_use <chr>, accountValue <dbl>, ground_truth <dbl>,
## #   intervene <int>, outcome <chr>, pAwardCurrent <dbl>, pAwardNew <dbl>,
## #   p_award_with <dbl>, p_award_without <dbl>, p_superiority <int>,
## #   payoff <dbl>, resp_time <dbl>, start_time <dbl>, trial <chr>,
## #   trial_dur <dbl>, trial_idx <chr>, means <fct>
```

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_llo <- 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)
  )
```

Now, lets apply our exclusion criteria.

```
# determine exclusions
exclude_df <- model_df_llo %>%
  # 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 to modeling data set
model_df_llo <- model_df_llo %>% left_join(exclude_df, by = "worker_id") %>% filter(excl
ude == FALSE)
```

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_llo %>% select(lo_ground_truth) %>% summarize(mean = mean(lo_ground_truth))
```

```
## # A tibble: 1 x 1
##    mean
##   <dbl>
## 1  1.96
```

```
# get_prior(data = model_df_llo, 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_llo, family = "gaussian",
              lo_p_sup ~ 1,
              prior = c(prior(normal(1.96, 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
```

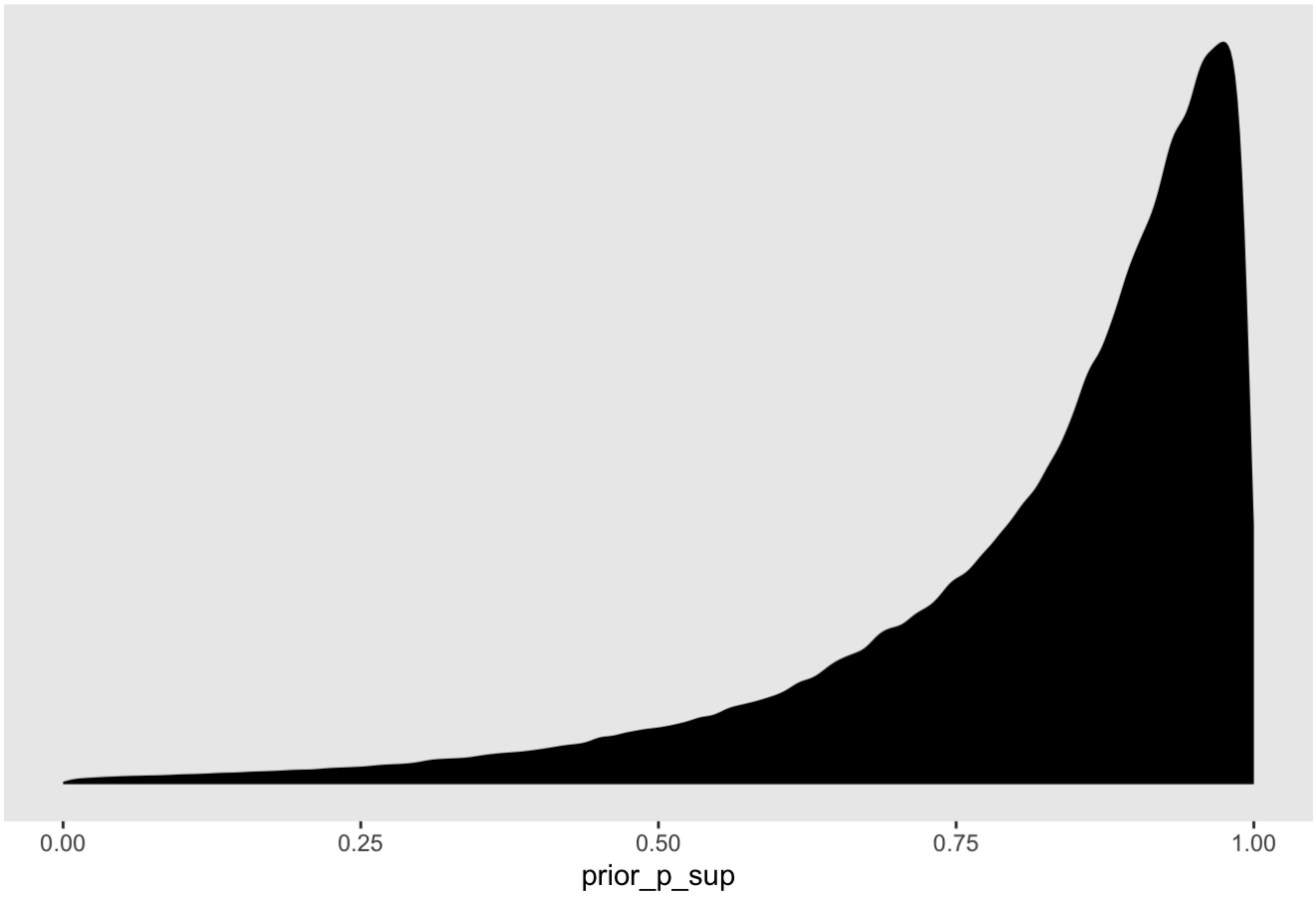
```
## Warning: There were 11 divergent transitions after warmup. Increasing adapt_delta abo
ve 0.8 may help. See
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

Let's look at our prior predictive distribution. For this intercept model, it should be approximately flat with a peak at 50% where the intercept is located.

```
# prior predictive check
model_df_llo %>%
  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



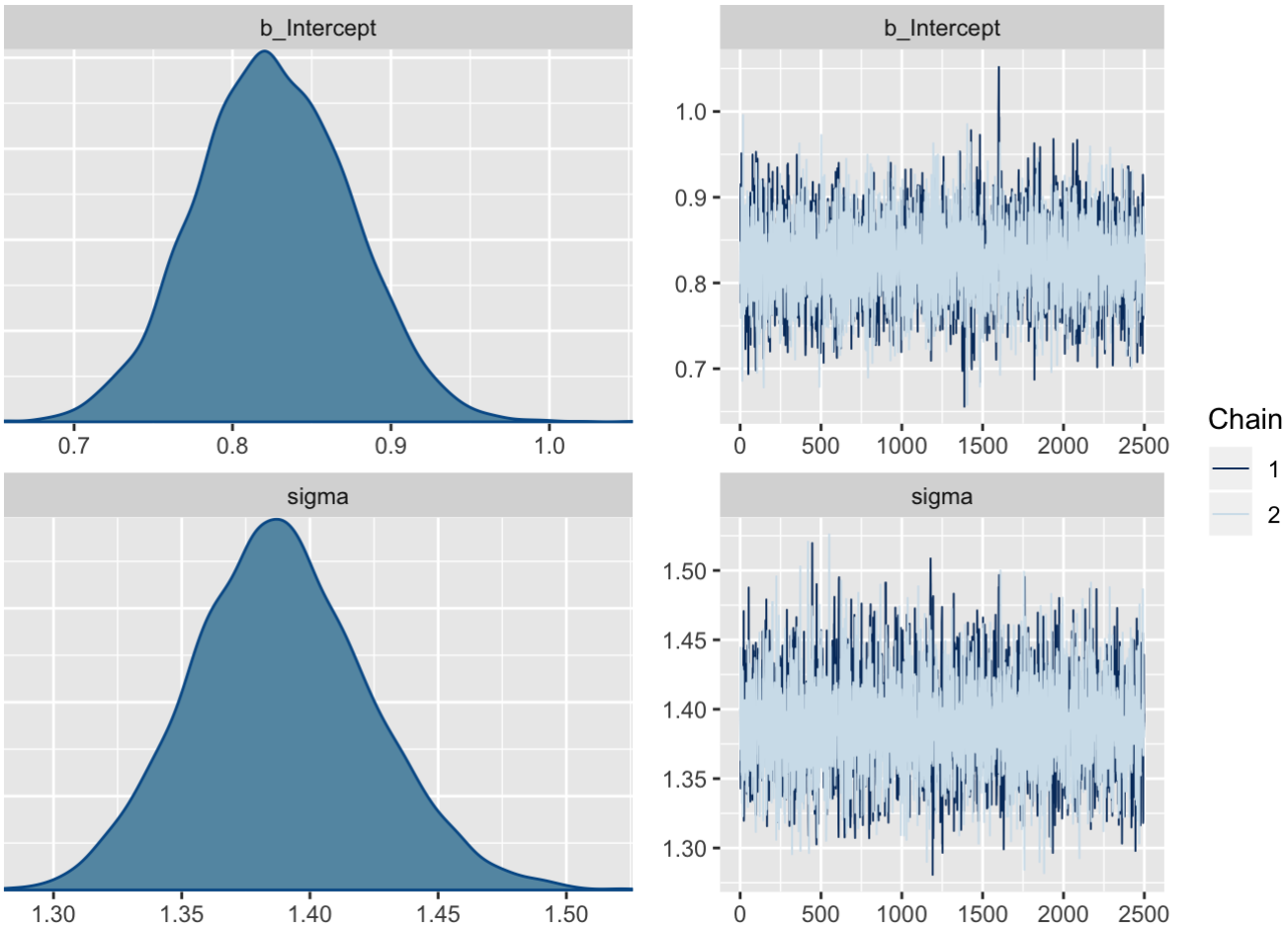
Now, let's fit the model to data. This is just trying to estimate the mean response regardless of the ground truth.

```
# starting as simple as possible: learn the distribution of lo_p_sup
m.lo_p_sup <- brm(data = model_df_llo, family = "gaussian",
              lo_p_sup ~ 1,
              prior = c(prior(normal(1.96, 1), class = Intercept),
                        prior(normal(0, 1), class = sigma)),
              iter = 3000, warmup = 500, chains = 2, cores = 2,
              file = "model-fits/lo_mdl")
```

Check diagnostics:

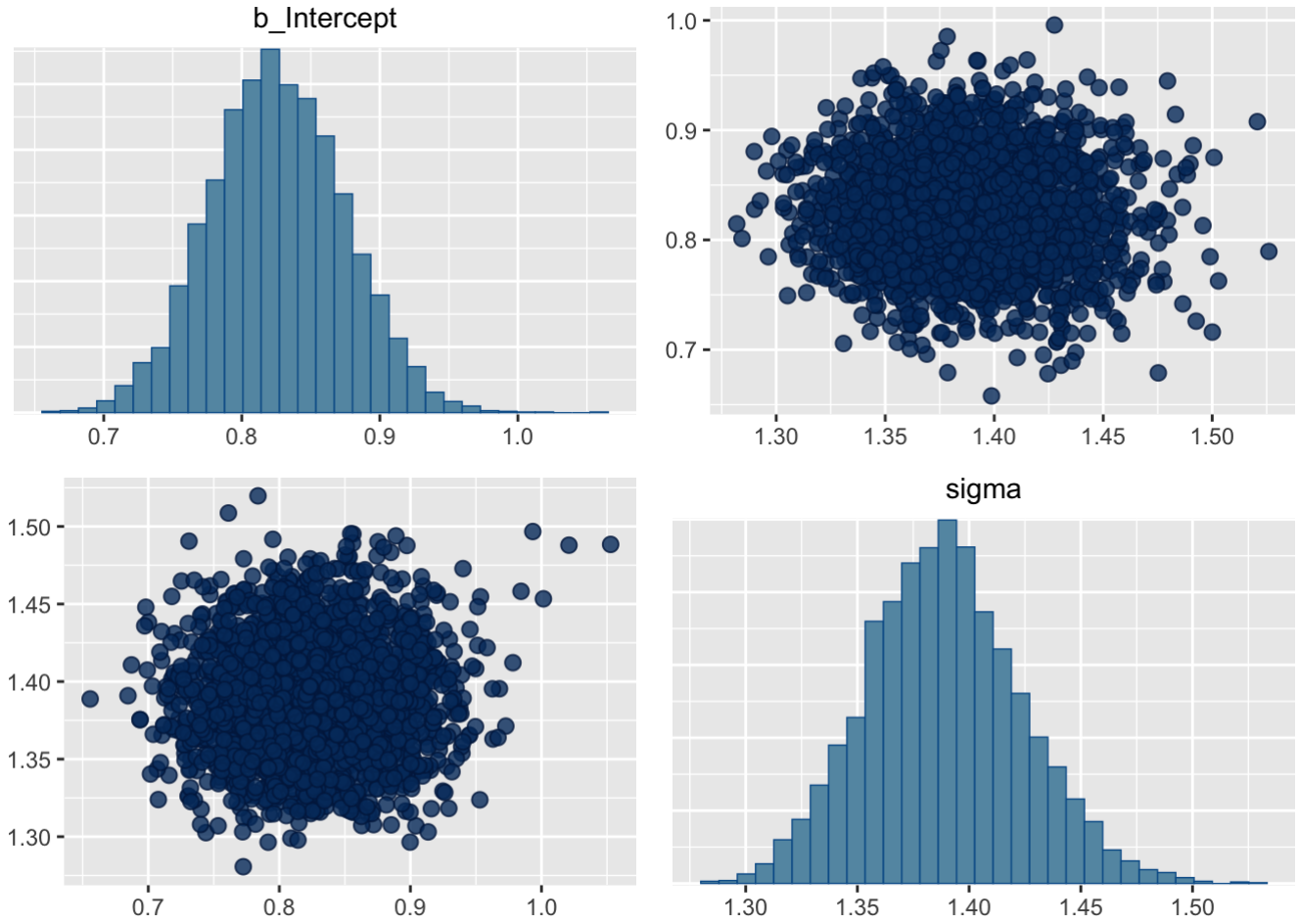
- Trace plots

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



- Pairs plot

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



- Summary

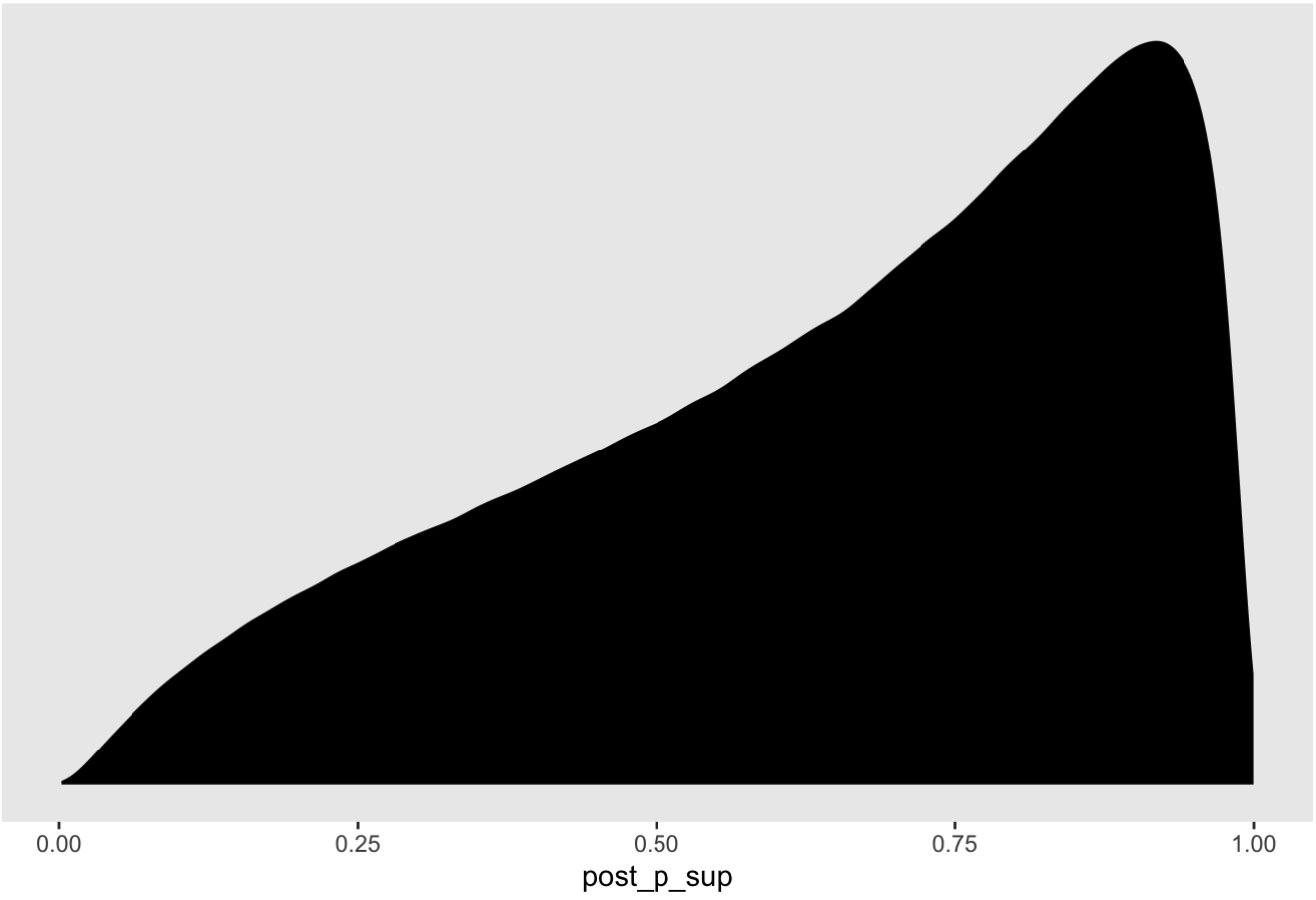
```
# model summary
print(m.lo_p_sup)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: lo_p_sup ~ 1
##    Data: model_df_llo (Number of observations: 840)
## 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     0.83      0.05     0.73     0.92       4388 1.00
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## sigma     1.39      0.03     1.32     1.46       4344 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_llo %>%
  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())
```

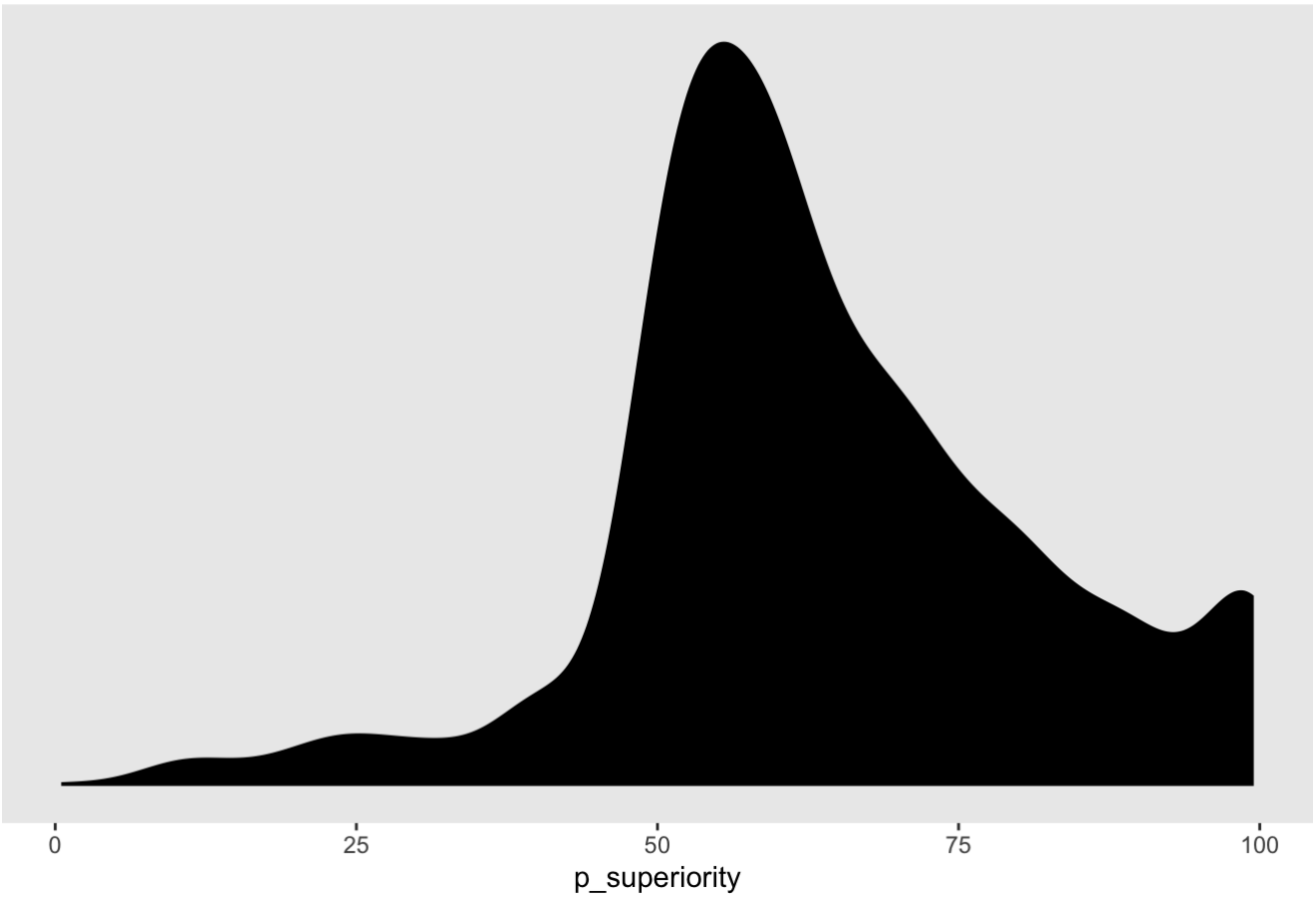
Posterior predictive distribution for probability of superiority



How do these predictions compare to the observed data?

```
# data density
model_df_llo %>%
  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



Obviously, our model is not sensitive to the ground truth.

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.

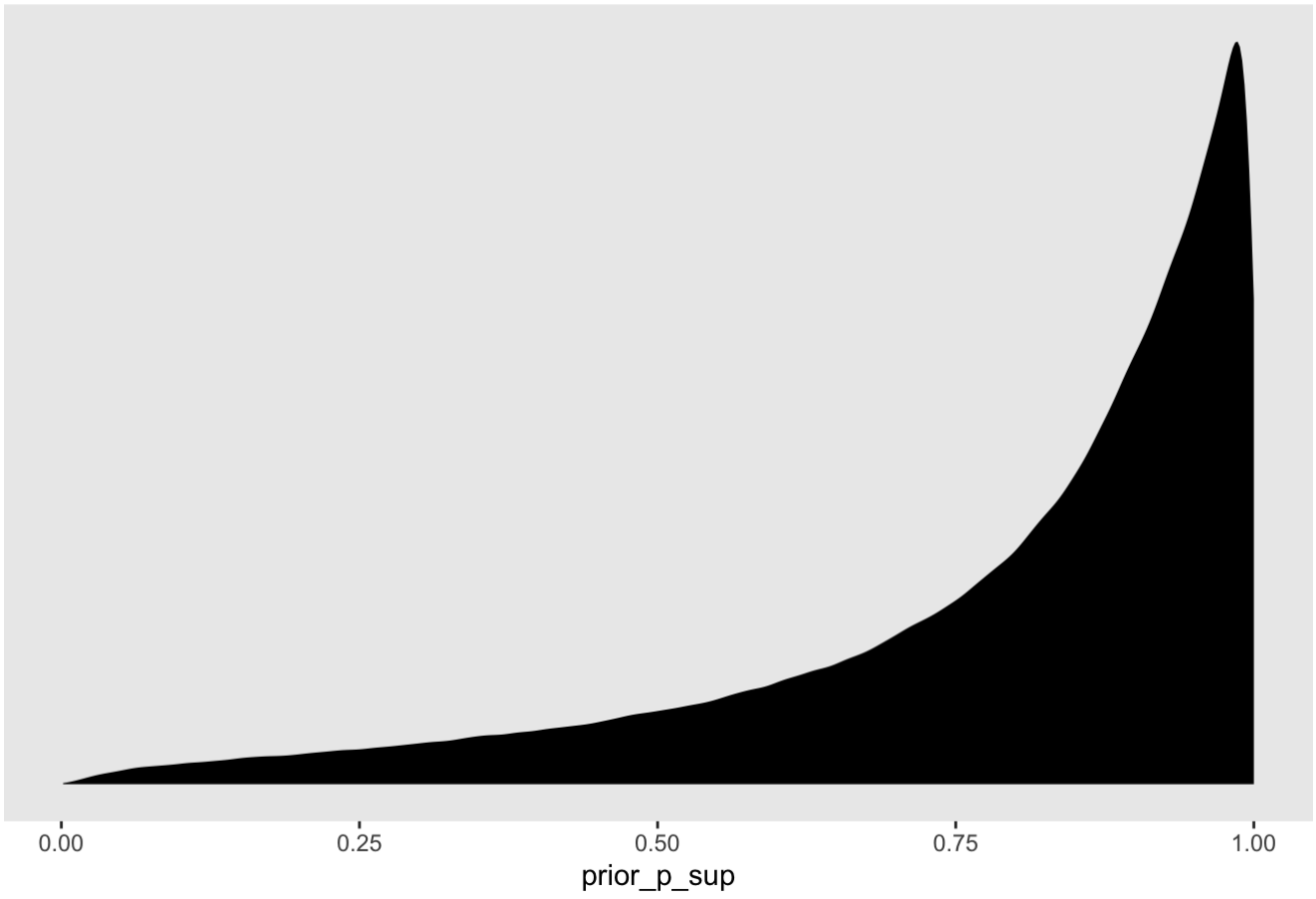
```
# get_prior(data = model_df_llo, family = "gaussian", formula = lo_p_sup ~ lo_ground_tru
th)

# simple LLO model
prior.llo_p_sup <- brm(data = model_df_llo, family = "gaussian",
                       lo_p_sup ~ lo_ground_truth,
                       prior = c(prior(normal(1, 0.5), class = b),
                                 prior(normal(1.96, 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 across a broader range of values.

```
# prior predictive check
model_df_llo %>%
  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())
```



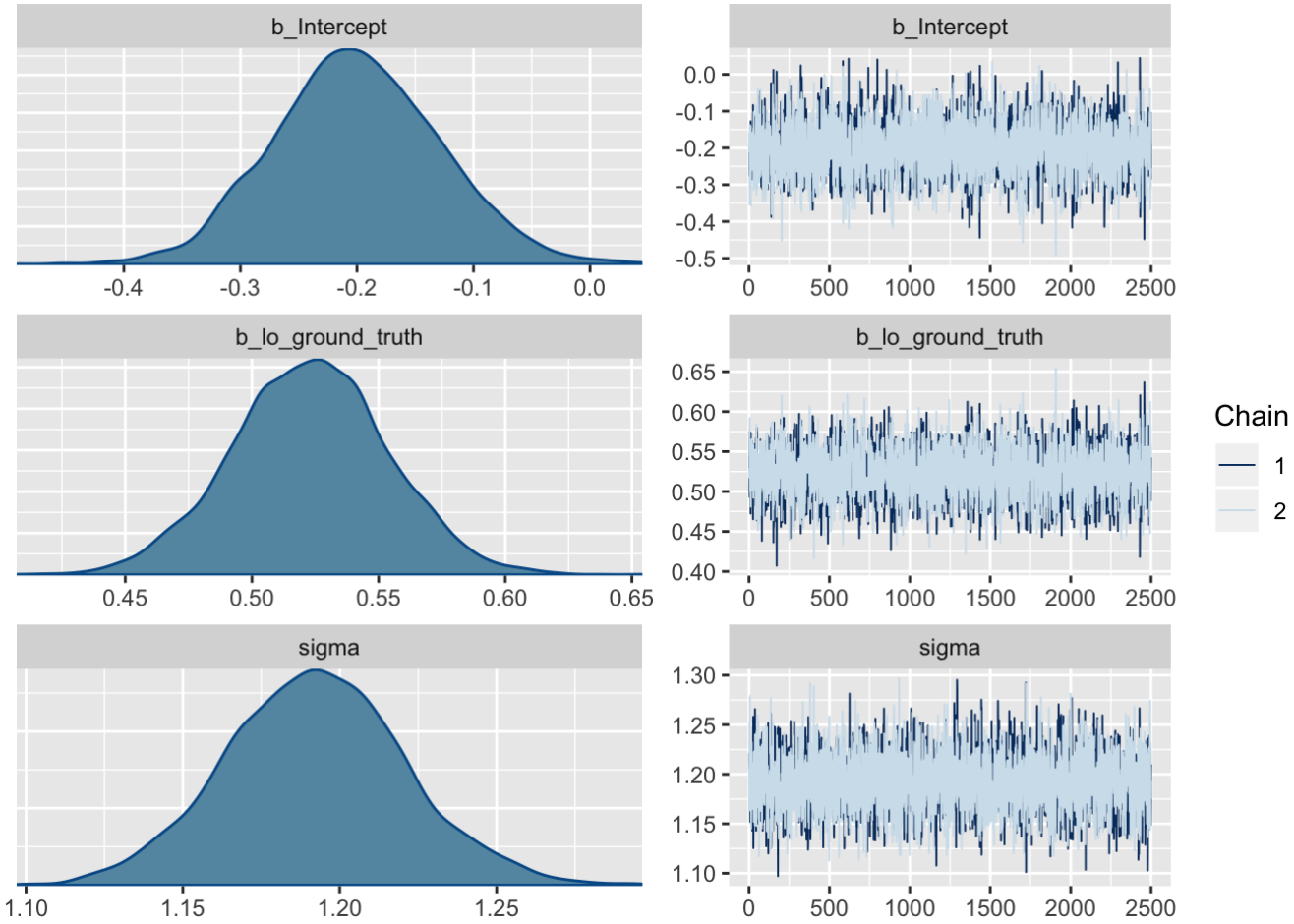
Now let's fit the model to data.

```
# simple LLO model
m.llo_p_sup <- brm(data = model_df_llo, family = "gaussian",
                   lo_p_sup ~ lo_ground_truth,
                   prior = c(prior(normal(1, 0.5), class = b),
                             prior(normal(1.96, 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:

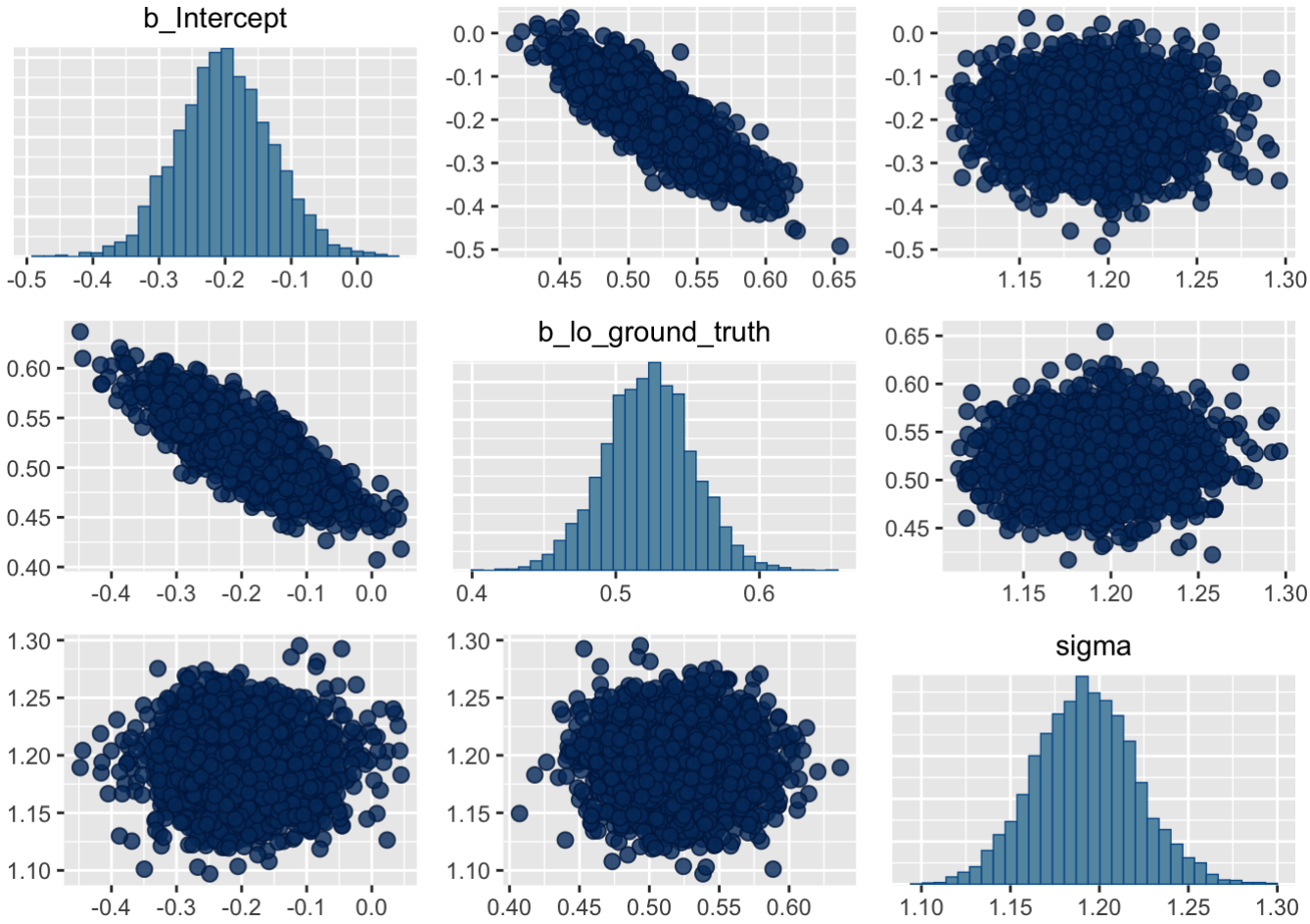
- Trace plots

```
# 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

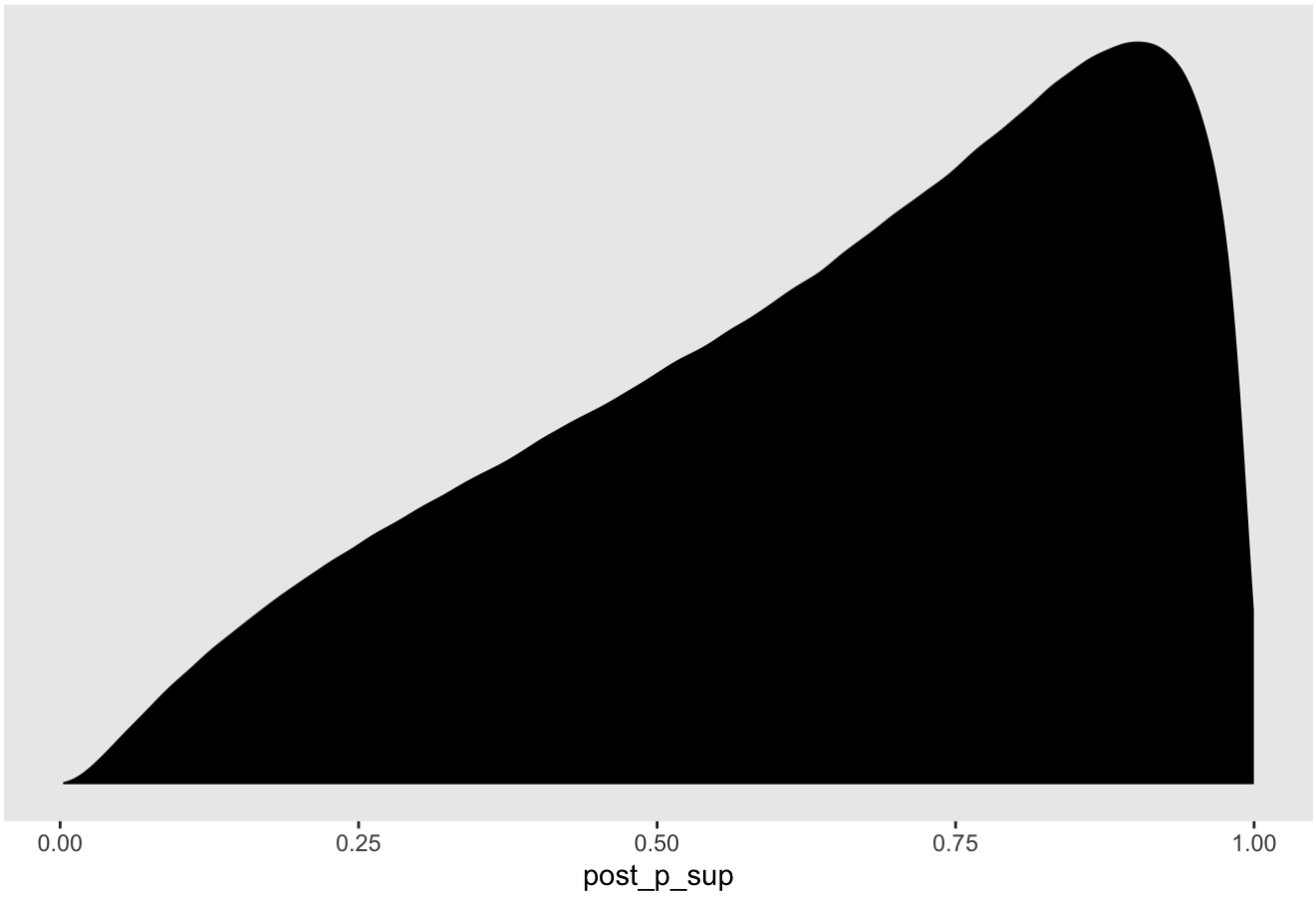
```
# model summary
print(m.llo_p_sup)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: lo_p_sup ~ lo_ground_truth
##    Data: model_df_llo (Number of observations: 840)
## 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          -0.20      0.07    -0.33    -0.06       4066 1.00
## lo_ground_truth     0.52      0.03     0.46     0.58       3756 1.00
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## sigma     1.19      0.03     1.14     1.25       4535 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_llo %>%
  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())
```

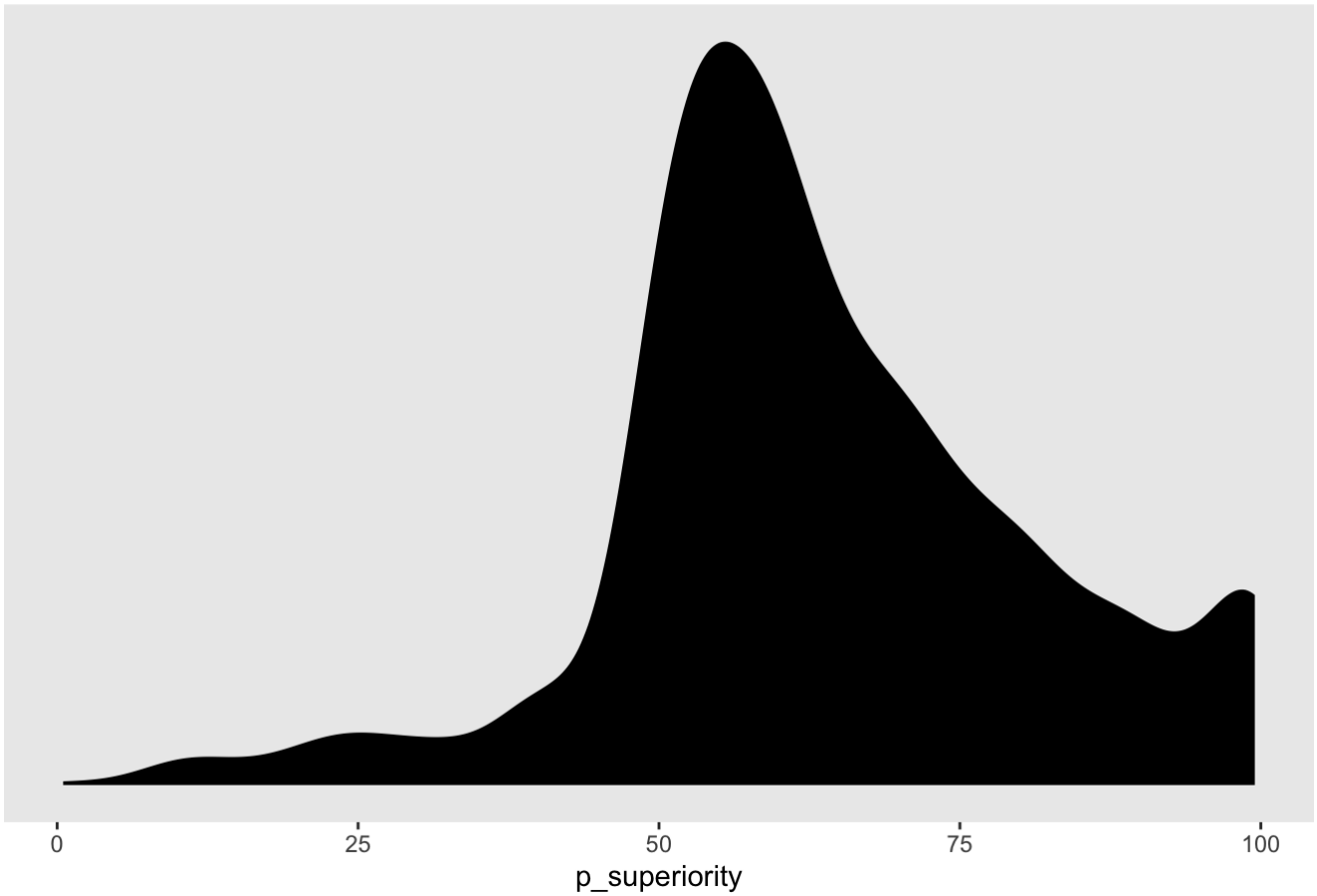
Posterior predictive distribution for probability of superiority



How do these predictions compare to the observed data?

```
# data density
model_df_llo %>%
  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. 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 the variability of 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_llo, family = "gaussian", formula = bf(lo_p_sup ~ (1 + lo_gr
ound_truth|sharecor|worker_id) + lo_ground_truth, sigma ~ (1|sharecor|worker_id)))

# hierarchical LLO model
prior.wrkr.llo_p_sup <- brm(data = model_df_llo, family = "gaussian",
                            formula = bf(lo_p_sup ~  (1 + lo_ground_truth|sharecor|worke
r_id) + lo_ground_truth,
                                         sigma ~ (1|sharecor|worker_id)),
                            prior = c(prior(normal(1, 0.5), class = b),
                                      prior(normal(1.96, 1), class = Intercept),
                                      prior(normal(0, 0.1), class = sd, group = worker_i
d),
                                      prior(normal(0, 0.1), class = sd, dpar = sigma),
                                      prior(lkj(4), class = cor)),
                            sample_prior = "only",
                            iter = 3000, warmup = 500, chains = 2, cores = 2)
```

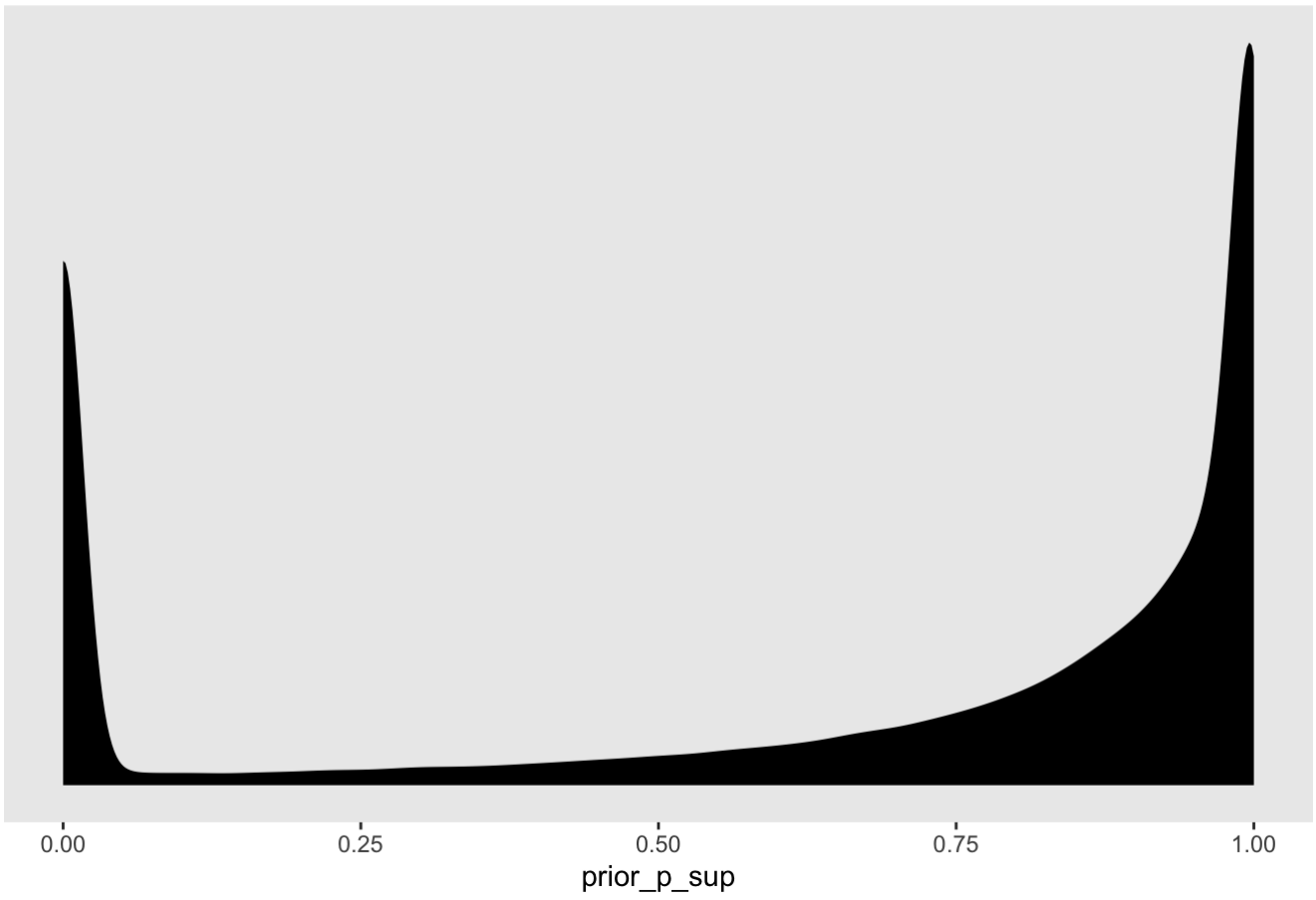
```
## Compiling the C++ model
```

```
## Start sampling
```

```
# prior.wrkr.llo_p_sup <- brm(data = model_df_llo, family = "gaussian",
#                             formula = bf(lo_p_sup ~  0 + intercept + (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(0, 0.02), class = b, coef = interce
pt),
#                                       prior(normal(0, 0.1), class = sd, group = worker
_id),
#                                       prior(gamma(2, 80), class = sd, group = worker_i
d, coef = Intercept), # zero-avoiding
#                                       prior(normal(0, 0.1), class = sd, dpar = sigma),
#                                       prior(lkj(4), class = cor)),
#                             sample_prior = "only",
#                             iter = 3000, warmup = 500, chains = 2, cores = 2)
```

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.

```
# prior predictive check
model_df_llo %>%
  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())
```



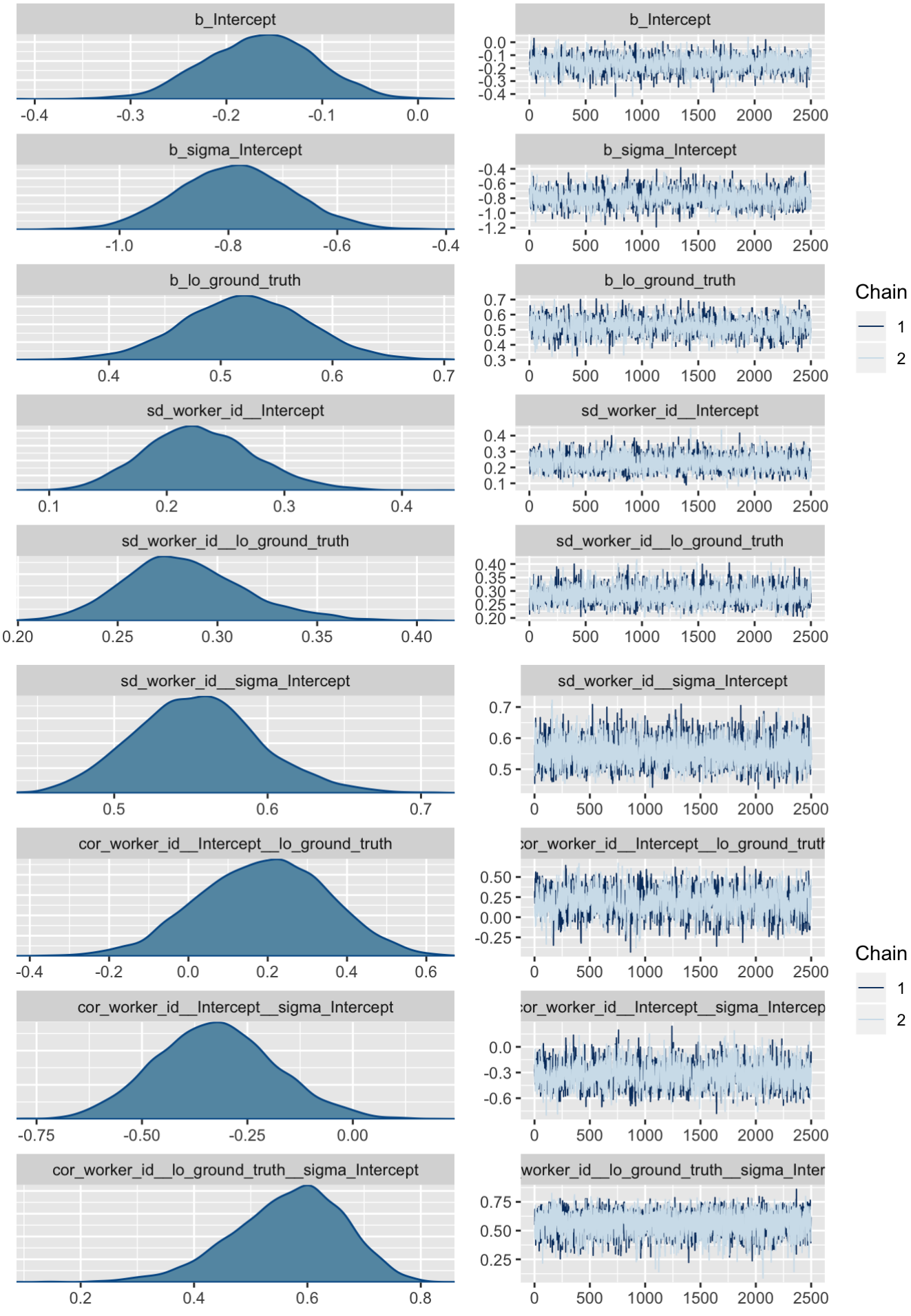
Now, let's fit the model to our data.

```
# hierarchical LLO model
m.wrkr.llo_p_sup <- brm(data = model_df_llo, family = "gaussian",
                        formula = bf(lo_p_sup ~  (1 + lo_ground_truth|sharecor|worker_i
d) + lo_ground_truth,
                                     sigma ~ (1|sharecor|worker_id)),
                        prior = c(prior(normal(1, 0.5), class = b),
                                  prior(normal(1.96, 1), class = Intercept),
                                  prior(normal(0, 0.1), class = sd, group = worker_id),
                                  prior(normal(0, 0.1), 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:

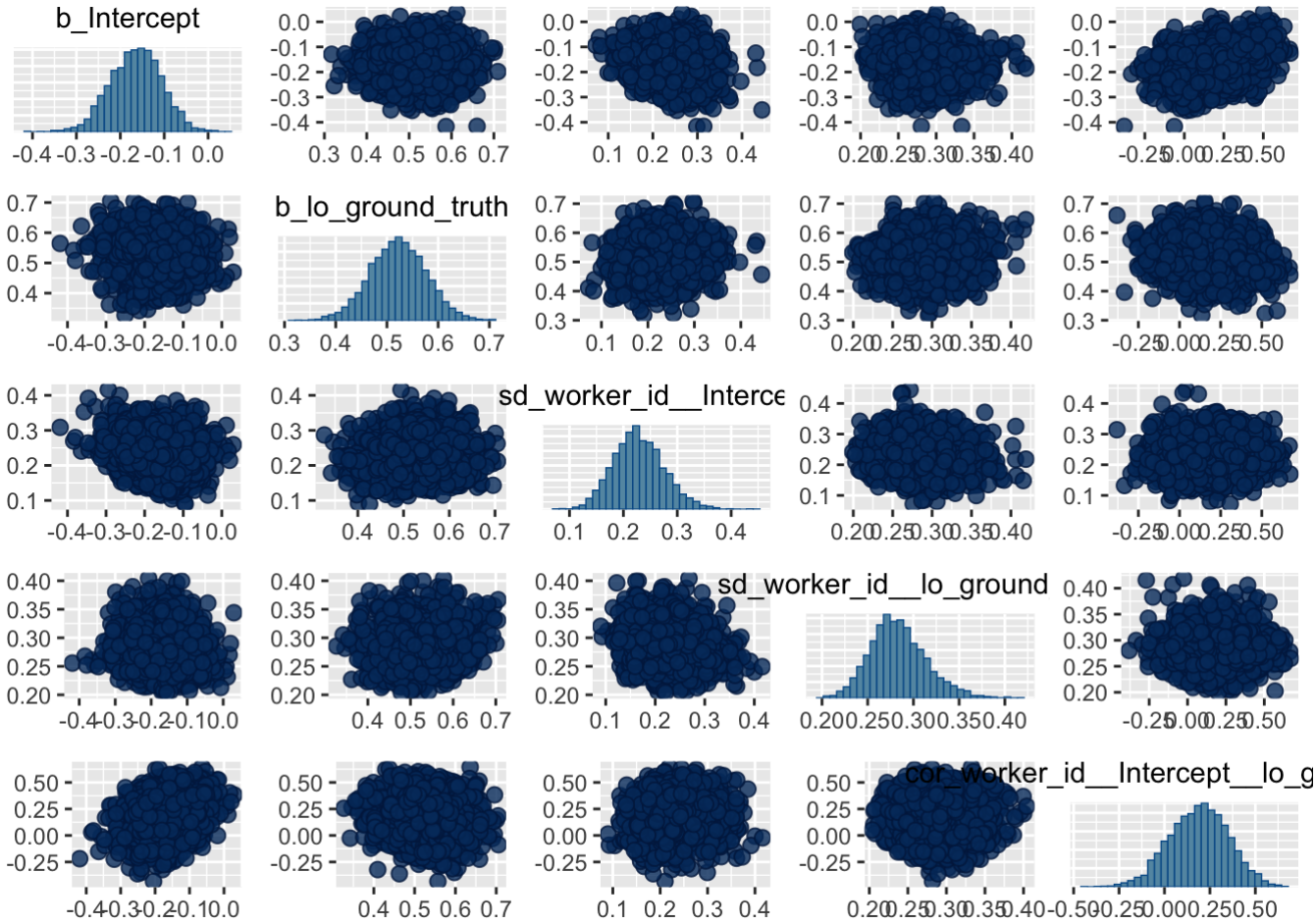
- Trace plots

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

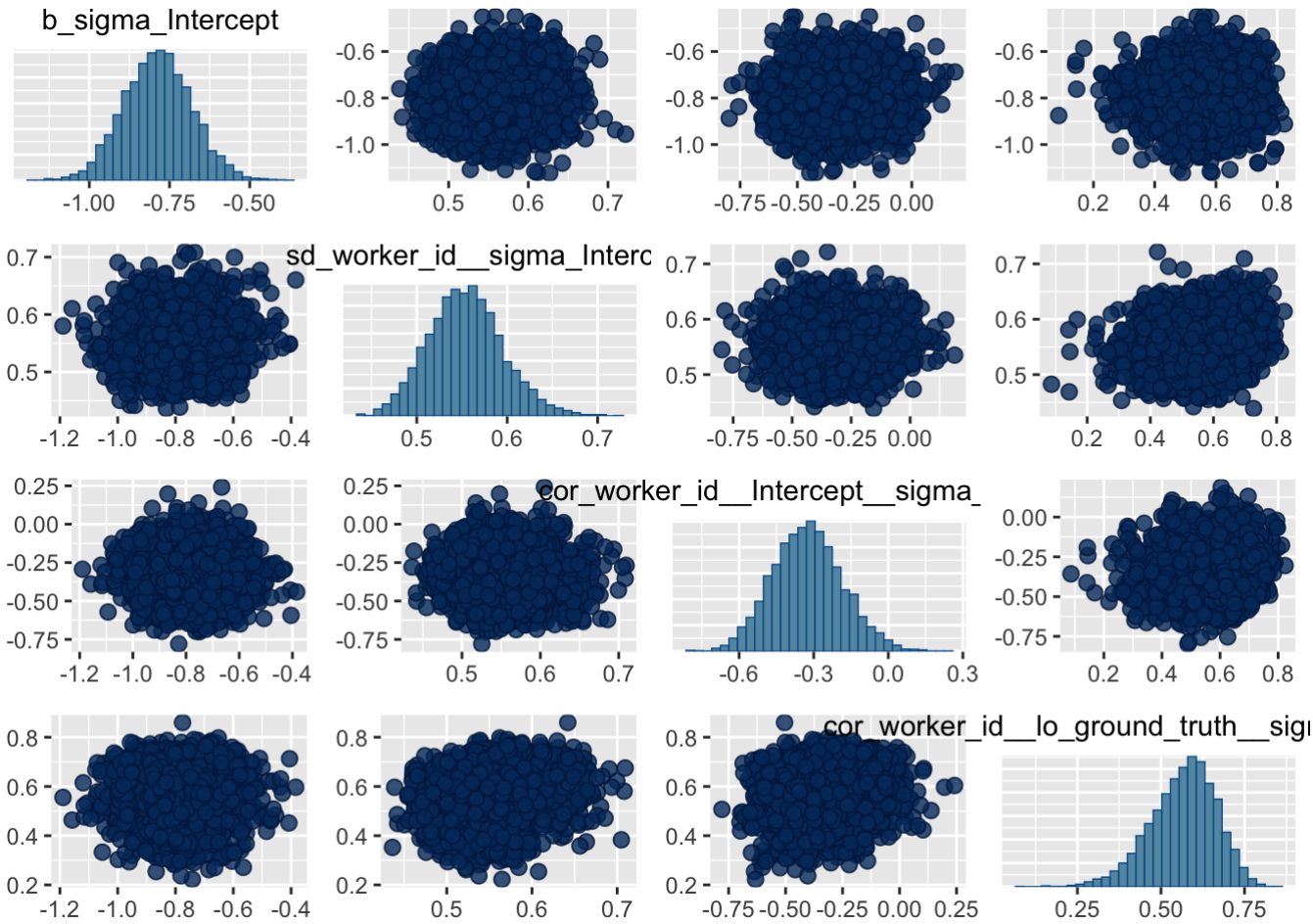

b_Intercept
b_sigma_Intercept
b_lo_ground_truth
sd_worker_id__Intercept
sd_worker_id__lo_ground_truth
sd_worker_id__sigma_Intercept
cor_worker_id__Intercept__lo_ground_truth
cor_worker_id__Intercept__sigma_Intercept
cor_worker_id__lo_ground_truth__sigma_Intercept
Chain
1
2

- Pairs plot

```
# pairs plot (LLO params)
pairs(m.wrkr.llo_p_sup, exact_match = TRUE, pars = c("b_Intercept", "b_lo_ground_truth",
"sd_worker_id__Intercept", "sd_worker_id__lo_ground_truth", "cor_worker_id__Intercept__l
o_ground_truth"))
```




```
# pairs plot (sigma params)
pairs(m.wrkr.llo_p_sup, exact_match = TRUE, pars = c("b_sigma_Intercept", "sd_worker_id_
_sigma_Intercept", "cor_worker_id__Intercept__sigma_Intercept", "cor_worker_id__lo_groun
d_truth__sigma_Intercept"))
```



- Summary

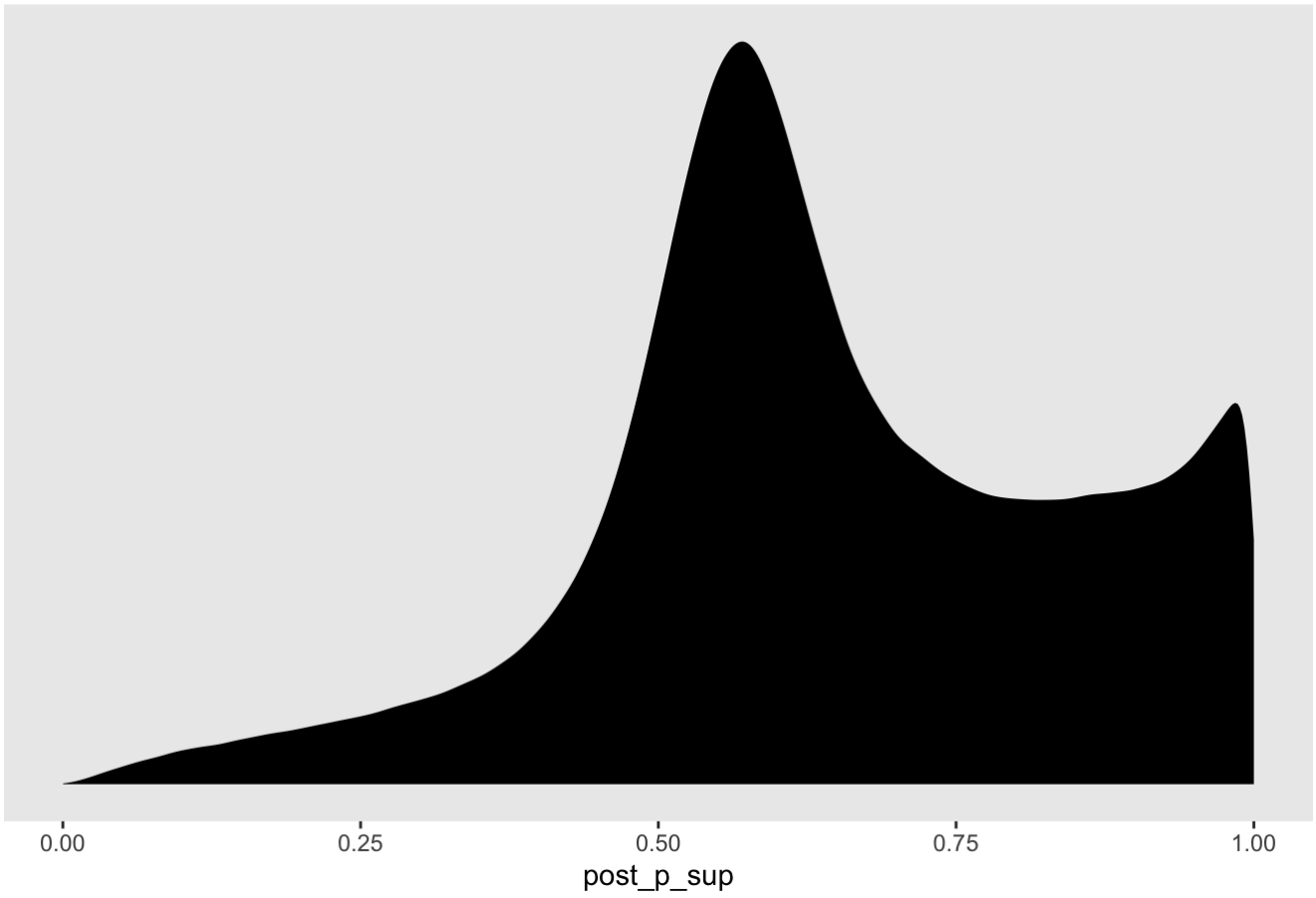
```
# model summary
print(m.wrkr.llo_p_sup)
```

```
##  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_llo (Number of observations: 840)
## Samples: 2 chains, each with iter = 3000; warmup = 500; thin = 1;
##          total post-warmup samples = 5000
##
## Group-Level Effects:
## ~worker_id (Number of levels: 28)
##                                     Estimate Est.Error l-95% CI u-95% CI
## sd(Intercept)                           0.23      0.05     0.14     0.33
## sd(lo_ground_truth)                     0.28      0.03     0.23     0.35
## sd(sigma_Intercept)                     0.55      0.04     0.48     0.64
## cor(Intercept,lo_ground_truth)          0.19      0.16    -0.13     0.50
## cor(Intercept,sigma_Intercept)         -0.32      0.14    -0.59    -0.04
## cor(lo_ground_truth,sigma_Intercept)    0.57      0.10     0.36     0.73
##                                     Eff.Sample Rhat
## sd(Intercept)                             2454 1.00
## sd(lo_ground_truth)                       2211 1.00
## sd(sigma_Intercept)                       4239 1.00
## cor(Intercept,lo_ground_truth)            1319 1.00
## cor(Intercept,sigma_Intercept)            1806 1.00
## cor(lo_ground_truth,sigma_Intercept)       4294 1.00
##
## Population-Level Effects:
##                 Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## Intercept          -0.16      0.06    -0.28    -0.05       1886 1.00
## sigma_Intercept    -0.79      0.11    -0.99    -0.58       1687 1.00
## lo_ground_truth     0.52      0.06     0.41     0.63       1127 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_llo %>%
  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())
```

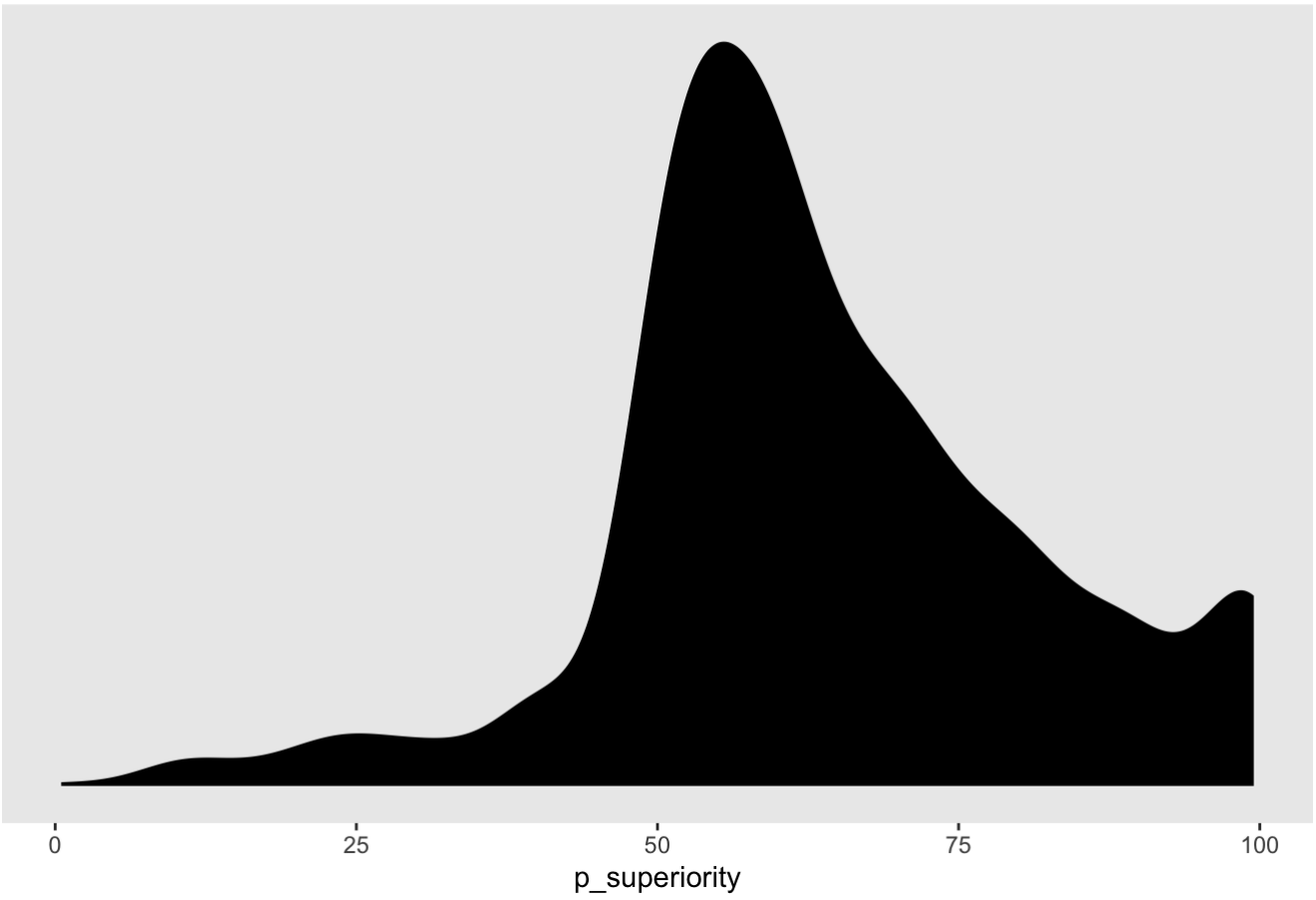
Posterior predictive distribution for probability of superiority



How do these predictions compare to the observed data?

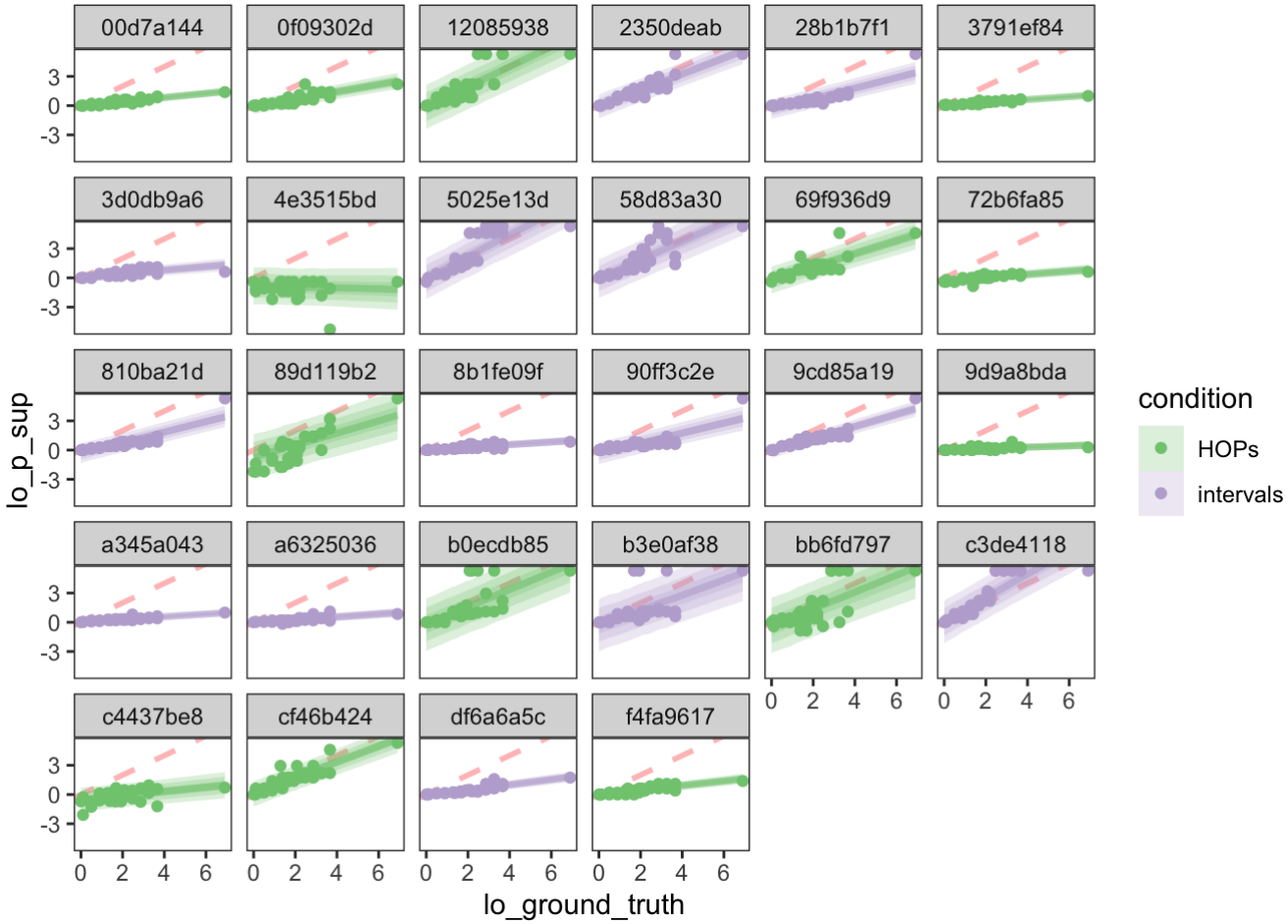
```
# data density
model_df_llo %>%
  ggplot(aes(x = p_superiority)) +
  geom_density(fill = "black", size = 0) +
  scale_y_continuous(NULL, breaks = NULL) +
  labs(subtitle = "Data distribution for probability of superiority") +
  theme(panel.grid = element_blank())
```

Data distribution for probability of superiority



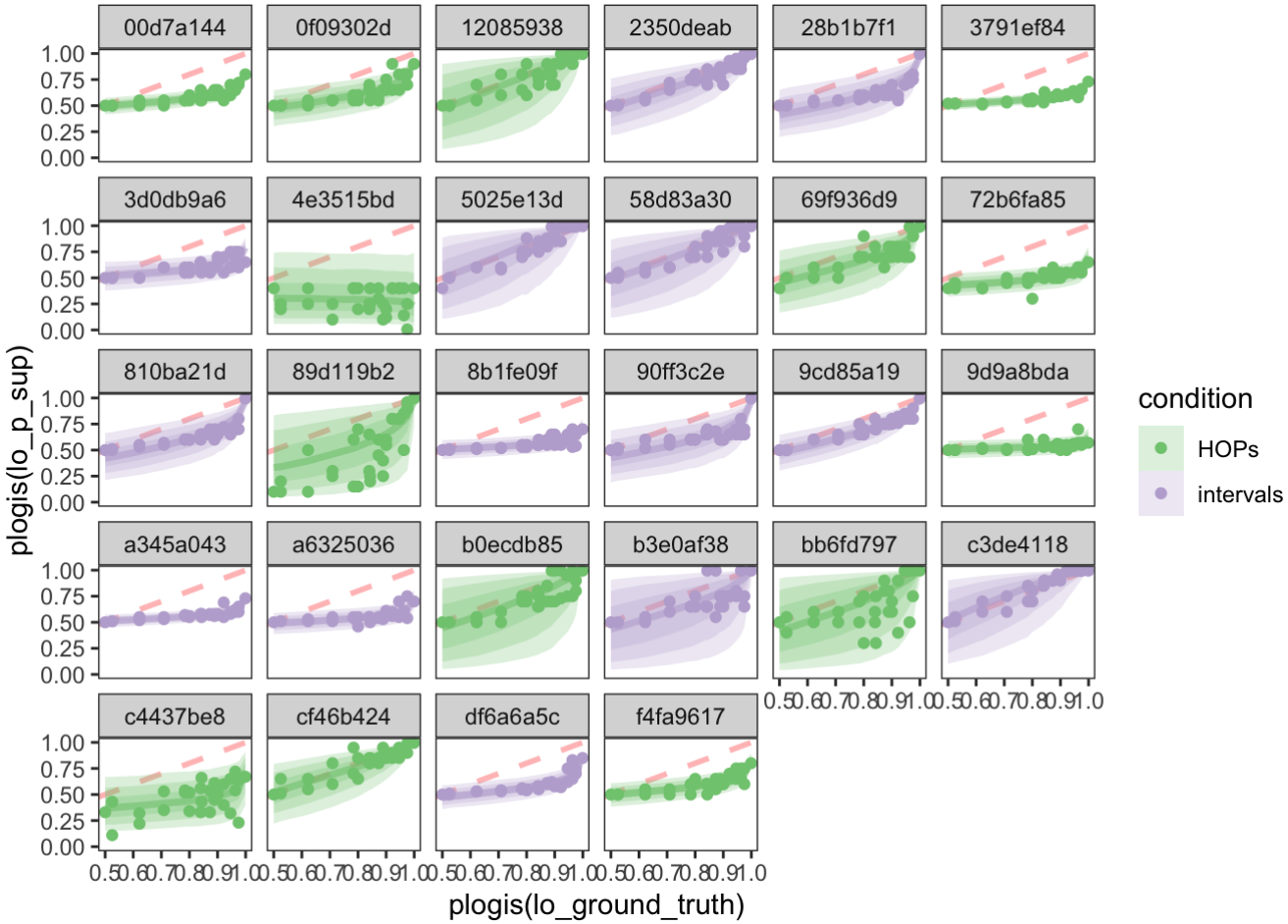
Let's look at posterior predictions per worker to get a more detailed sense of fit quality.

```
model_df_llo %>%
  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_llo) +
  scale_fill_brewer(type = "qual", palette = 1) +
  scale_color_brewer(type = "qual", palette = 1) +
  coord_cartesian(xlim = quantile(model_df_llo$lo_ground_truth, c(0, 1)),
                  ylim = quantile(model_df_llo$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_llo %>%
  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_llo) +
  scale_fill_brewer(type = "qual", palette = 1) +
  scale_color_brewer(type = "qual", palette = 1) +
  coord_cartesian(xlim = quantile(plogis(model_df_llo$lo_ground_truth), c(0, 1)),
                  ylim = quantile(plogis(model_df_llo$lo_p_sup), c(0, 1))) +
  theme_bw() +
  theme(panel.grid = element_blank()) +
  facet_wrap(~ worker_id)
```



Overall, our model predictions are matching the density of our data better than any other model so far. Let's add our experimental manipulations as predictors.

Fixed Effects for the Presence 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.

Before we fit the model to our data, let's check that our priors seem reasonable. The only prior we add here is for the fixed effect of means present/absent on residual variance. We keep this prior a little wider than the others for sigma since we want to allow effects to vary by condition.

```
# get_prior(data = model_df_llo, family = "gaussian", formula = bf(lo_p_sup ~  (1 + lo_g
round_truth|sharecor|worker_id) + lo_ground_truth*means, sigma ~ (1|sharecor|worker_id)
 + means))

# hierarchical LLO model with a fixed effect for the presence/absence of the mean
prior.wrkr.means.llo_p_sup <- brm(data = model_df_llo, family = "gaussian",
                                  formula = bf(lo_p_sup ~  (1 + lo_ground_truth|sharecor
|worker_id) + lo_ground_truth*means,
                                               sigma ~ (1|sharecor|worker_id) + means),
                                  prior = c(prior(normal(1, 0.5), class = b),
                                            prior(normal(1.96, 1), class = Intercept),
                                            prior(normal(0, 0.1), class = sd, group = wo
rker_id),
                                            prior(normal(0, 0.2), class = b, dpar = sigm
a), # added prior
                                            prior(normal(0, 0.1), class = sd, dpar = sig
ma),
                                            prior(lkj(4), class = cor)),
                                  sample_prior = "only",
                                  iter = 3000, warmup = 500, chains = 2, cores = 2)
```

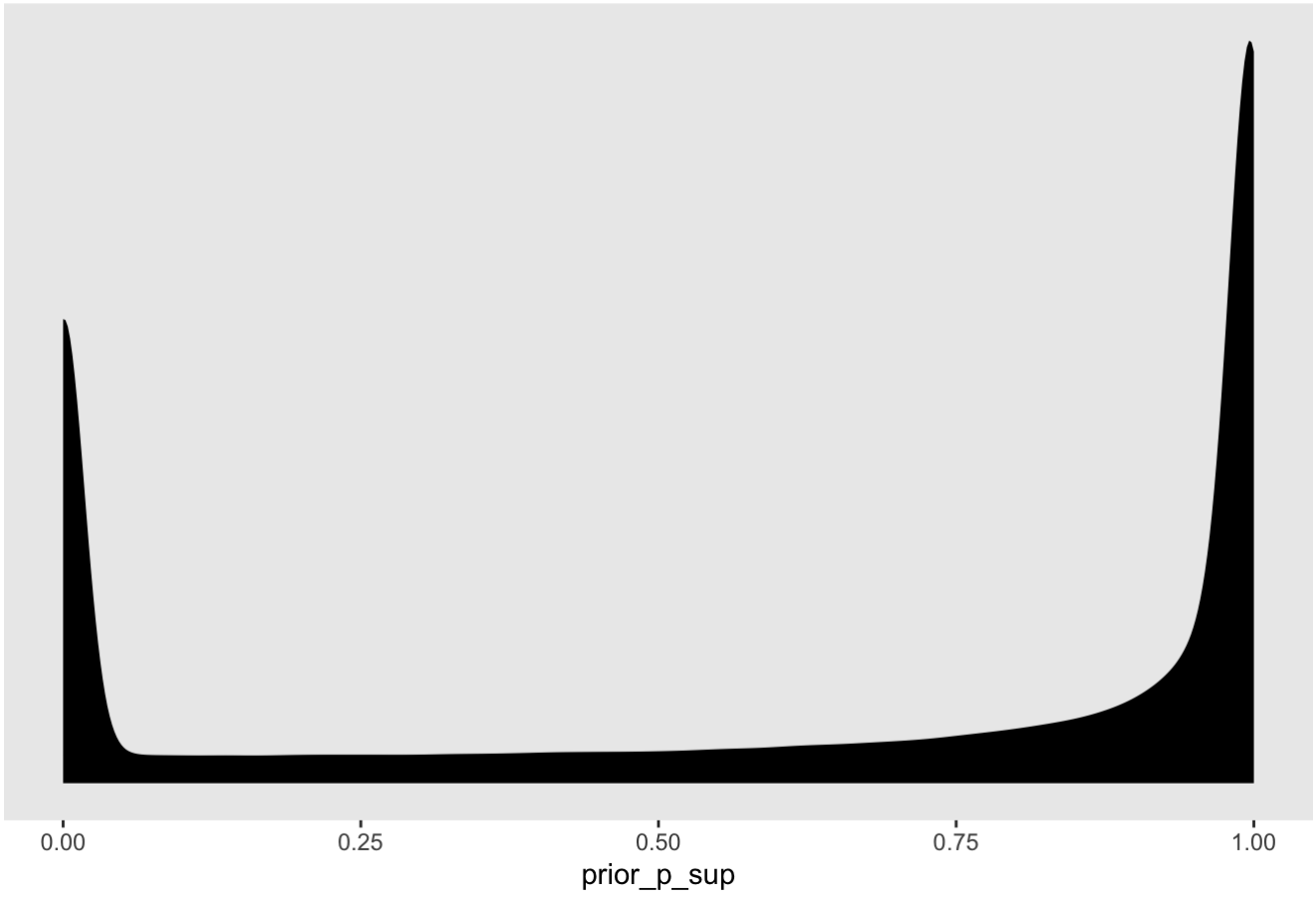
```
## Compiling the C++ model
```

```
## Start sampling
```

```
# prior.wrkr.means.llo_p_sup <- brm(data = model_df_llo, family = "gaussian",
#                                   formula = bf(lo_p_sup ~  0 + intercept + (lo_ground_truth|
worker_id) + lo_ground_truth:means,
#                                                sigma ~ (1|worker_id) + means),
#                                   prior = c(prior(normal(1, 0.5), class = b),
#                                             prior(normal(0, 0.02), class = b, coef = interce
pt),
#                                             prior(normal(0, 0.1), class = sd, group = worker
_id),
#                                             prior(normal(0, 0.1), class = sd, group = worker
_id, coef = Intercept),
#                                             prior(normal(0, 0.2), class = b, dpar = sigma),
 # added prior
#                                             prior(normal(0, 0.1), class = sd, dpar = sigma),
#                                             prior(lkj(4), class = cor)),
#                                   sample_prior = "only",
#                                   iter = 3000, warmup = 500, chains = 2, cores = 2)
```

Let's look at our prior predictive distribution. This should look about the same as our last model.

```
# prior predictive check
model_df_llo %>%
  select(lo_ground_truth, worker_id, means) %>%
  add_predicted_draws(prior.wrkr.means.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())
```



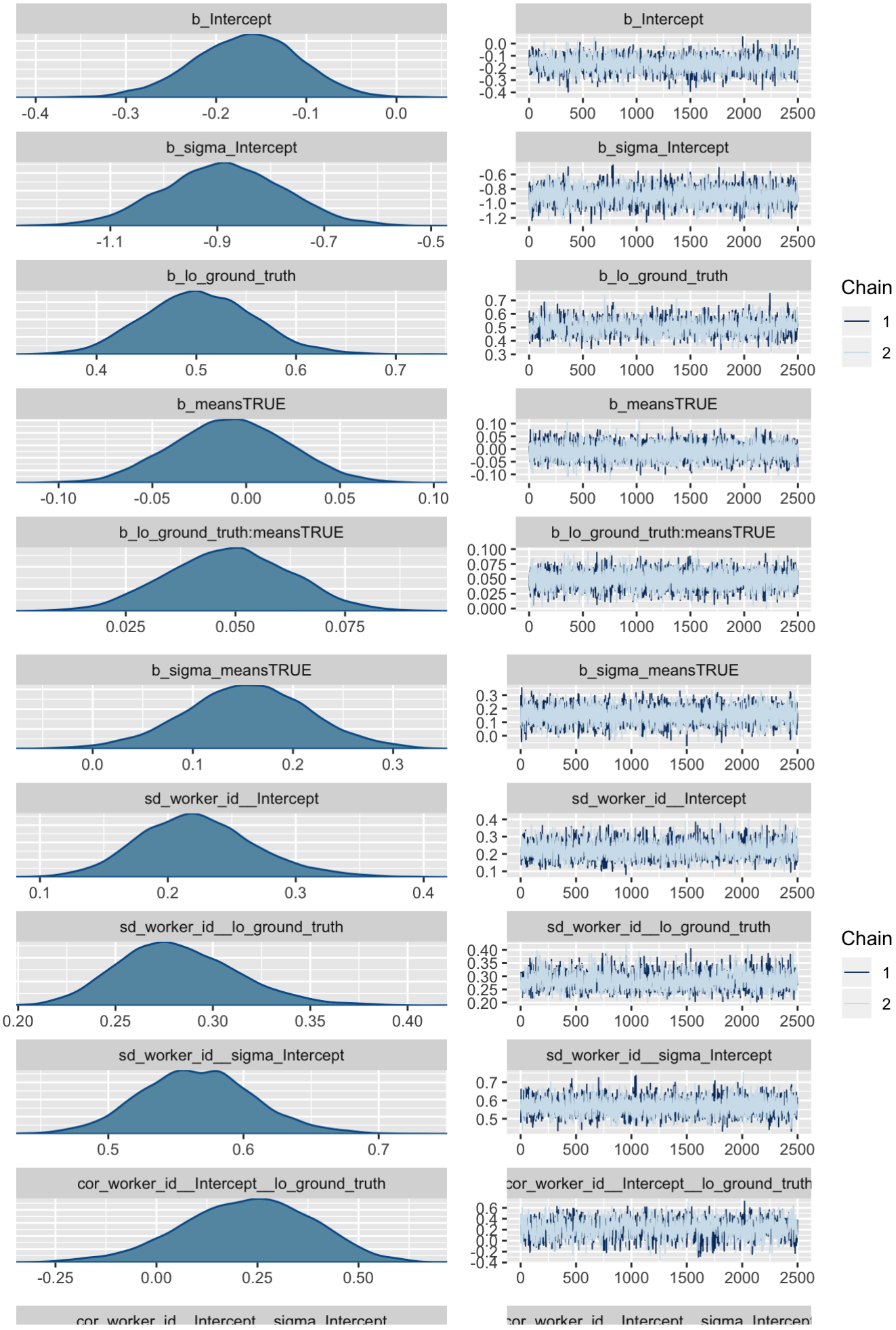
Now, let's fit the model to our data.

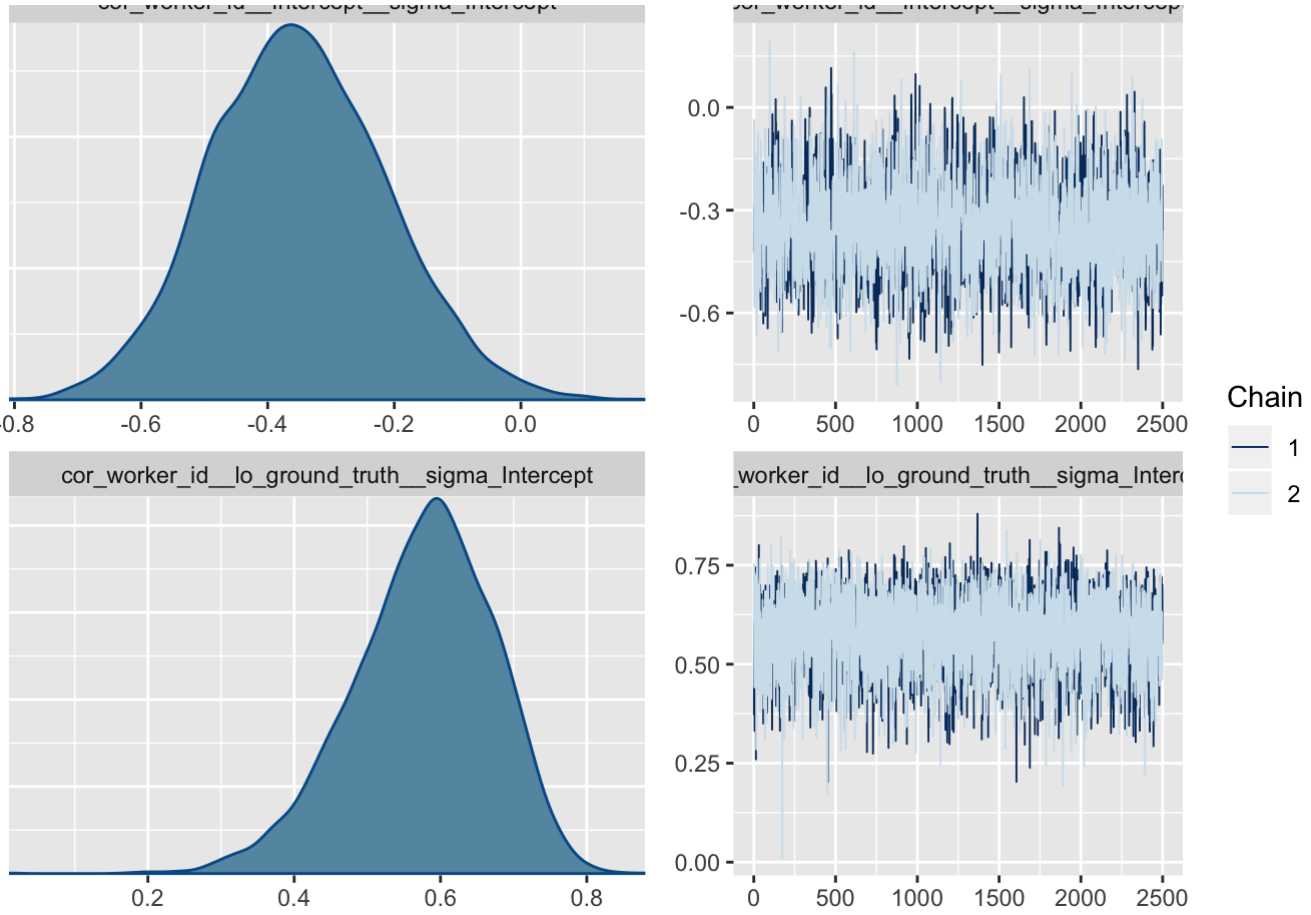
```
# hierarchical LLO model with fixed effects on slope and residual variance conditioned o
n the presence/absence of the mean
m.wrkr.means.llo_p_sup <- brm(data = model_df_llo, family = "gaussian",
                              formula = bf(lo_p_sup ~  (1 + lo_ground_truth|sharecor|wor
ker_id) + lo_ground_truth*means,
                                           sigma ~ (1|sharecor|worker_id) + means),
                              prior = c(prior(normal(1, 0.5), class = b),
                                        prior(normal(1.96, 1), class = Intercept),
                                        prior(normal(0, 0.1), class = sd, group = worker
_id),
                                        prior(normal(0, 0.2), class = b, dpar = sigma),
# added prior
                                        prior(normal(0, 0.1), 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")
```

Check diagnostics:

- Trace plots

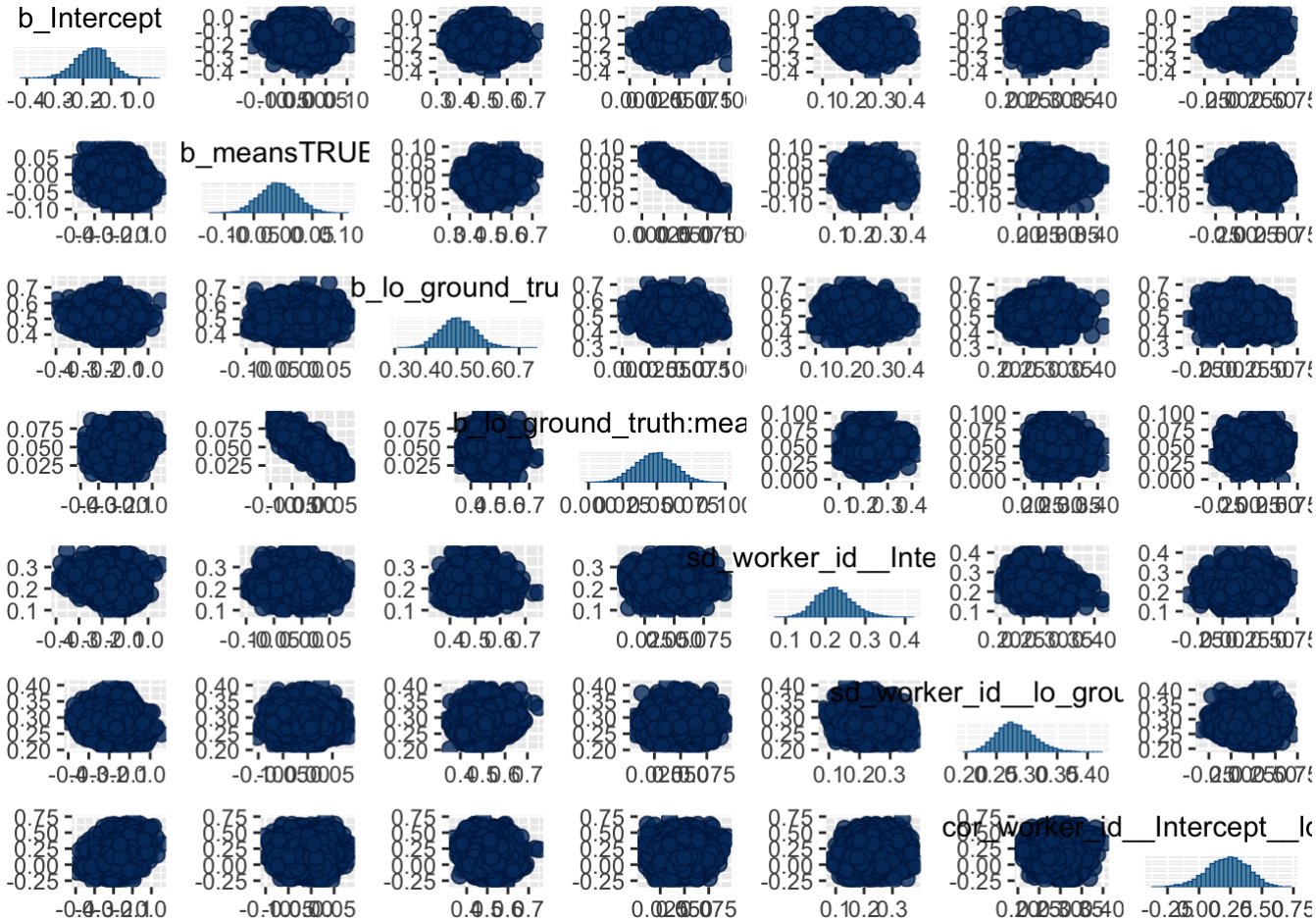
```
# trace plots
plot(m.wrkr.means.llo_p_sup)
```


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

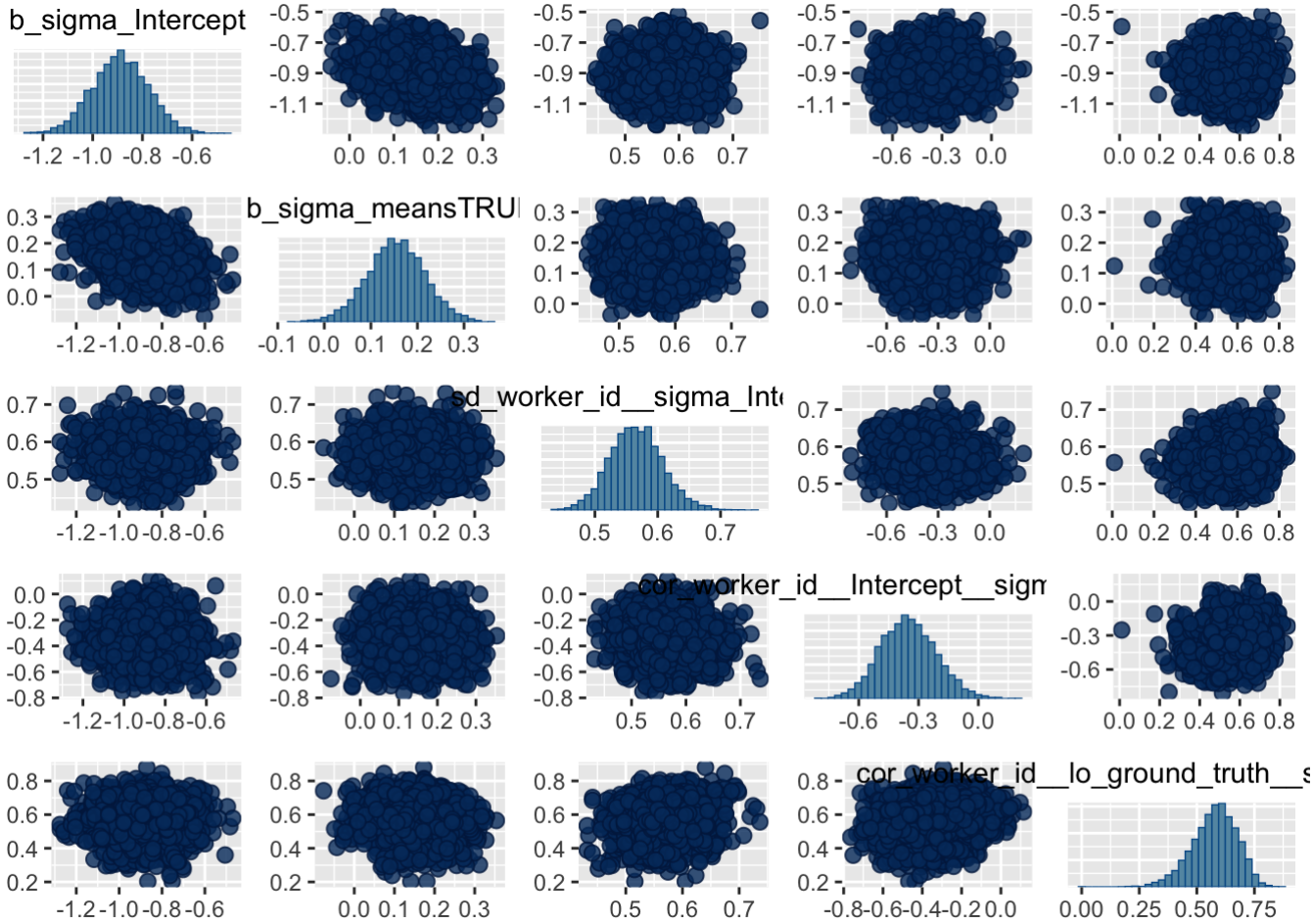


- Pairs plot

```
# pairs plot (LLO params)
pairs(m.wrkr.means.llo_p_sup, exact_match = TRUE, pars = c("b_Intercept", "b_meansTRUE",
"b_lo_ground_truth", "b_lo_ground_truth:meansTRUE", "sd_worker_id__Intercept", "sd_worke
r_id__lo_ground_truth", "cor_worker_id__Intercept__lo_ground_truth"))
```




```
# pairs plot (sigma params)
pairs(m.wrkr.means.llo_p_sup, exact_match = TRUE, pars = c("b_sigma_Intercept", "b_sigma
_meansTRUE", "sd_worker_id__sigma_Intercept", "cor_worker_id__Intercept__sigma_Intercep
t", "cor_worker_id__lo_ground_truth__sigma_Intercept"))
```



- Summary

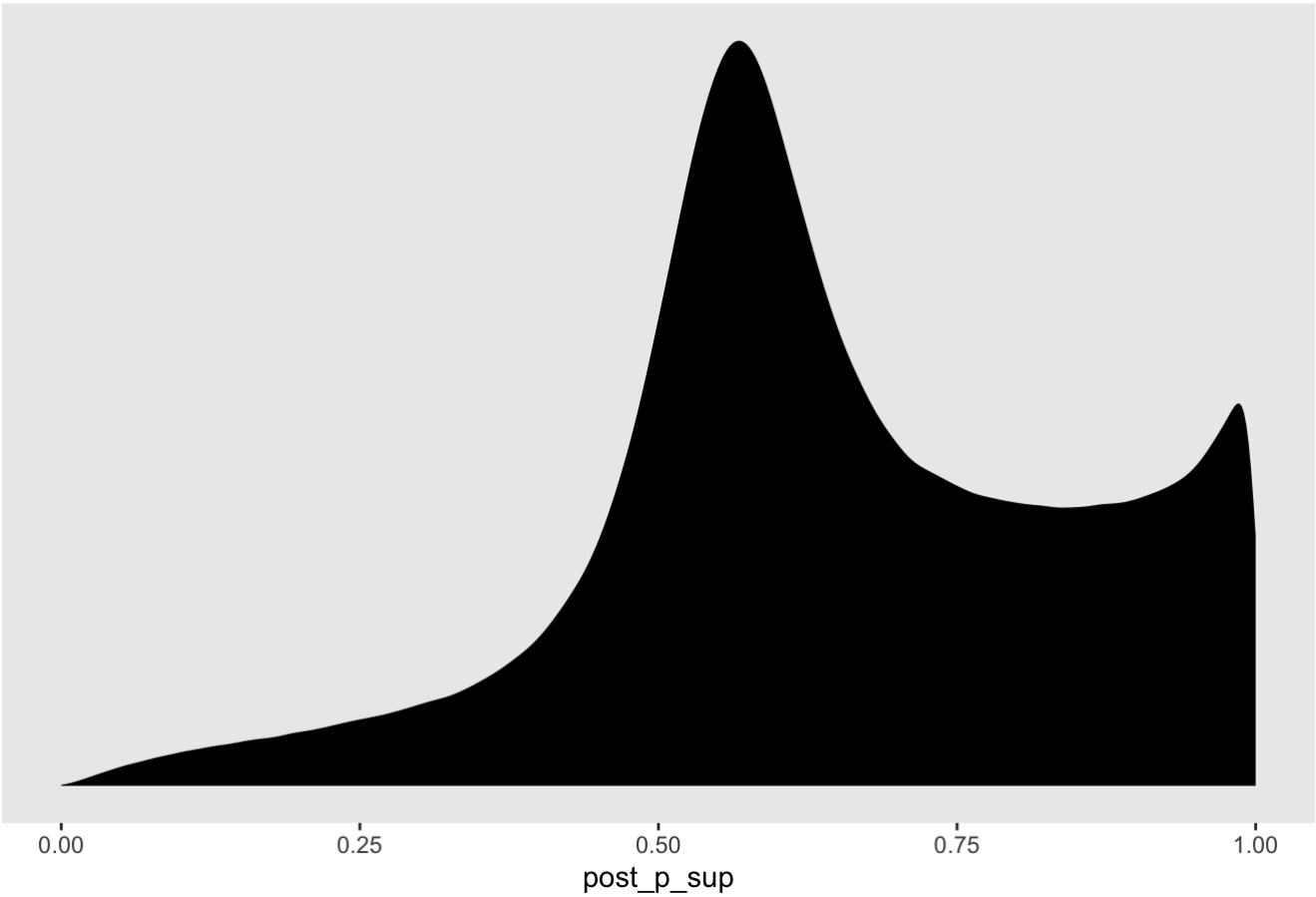
```
# model summary
print(m.wrkr.means.llo_p_sup)
```

```
##  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) + means
##    Data: model_df_llo (Number of observations: 840)
## Samples: 2 chains, each with iter = 3000; warmup = 500; thin = 1;
##          total post-warmup samples = 5000
##
## Group-Level Effects:
## ~worker_id (Number of levels: 28)
##                                     Estimate Est.Error l-95% CI u-95% CI
## sd(Intercept)                           0.22      0.05     0.14     0.32
## sd(lo_ground_truth)                     0.28      0.03     0.23     0.35
## sd(sigma_Intercept)                     0.57      0.04     0.49     0.65
## cor(Intercept,lo_ground_truth)          0.23      0.16    -0.10     0.52
## cor(Intercept,sigma_Intercept)         -0.35      0.14    -0.62    -0.07
## cor(lo_ground_truth,sigma_Intercept)    0.58      0.10     0.37     0.74
##                                     Eff.Sample Rhat
## sd(Intercept)                             2657 1.00
## sd(lo_ground_truth)                       2302 1.00
## sd(sigma_Intercept)                       3549 1.00
## cor(Intercept,lo_ground_truth)            1215 1.00
## cor(Intercept,sigma_Intercept)            1831 1.00
## cor(lo_ground_truth,sigma_Intercept)       5001 1.00
##
## Population-Level Effects:
##                           Estimate Est.Error l-95% CI u-95% CI Eff.Sample
## Intercept                    -0.17      0.06    -0.29    -0.05       2089
## sigma_Intercept              -0.89      0.11    -1.11    -0.67       2245
## lo_ground_truth               0.50      0.05     0.40     0.62       1467
## meansTRUE                    -0.01      0.03    -0.07     0.05       5945
## lo_ground_truth:meansTRUE     0.05      0.01     0.02     0.08       6178
## sigma_meansTRUE               0.15      0.06     0.03     0.28       8225
##                           Rhat
## Intercept                 1.00
## sigma_Intercept           1.00
## lo_ground_truth           1.00
## meansTRUE                 1.00
## lo_ground_truth:meansTRUE 1.00
## sigma_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_llo %>%
  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())
```

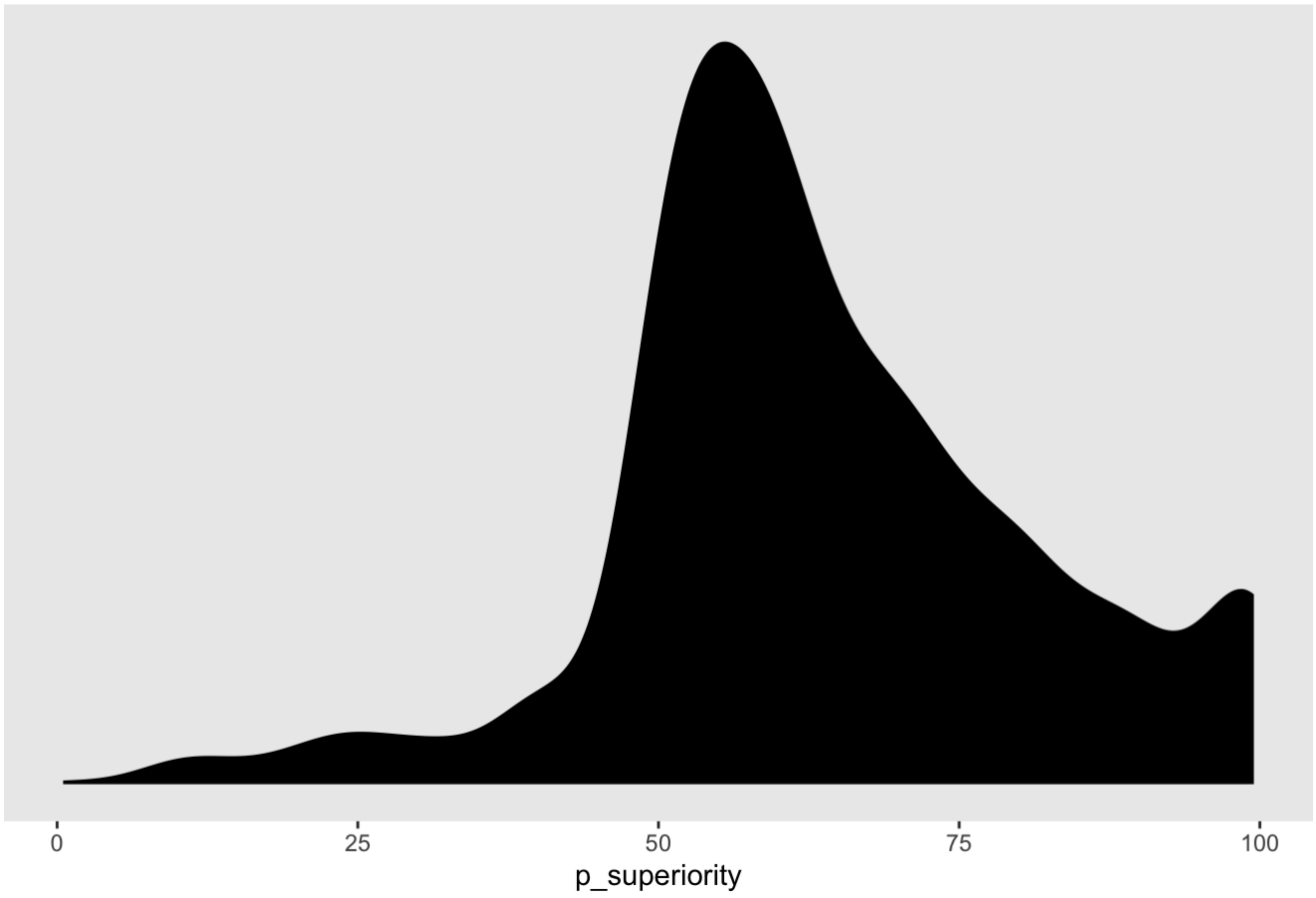
Posterior predictive distribution for probability of superiority



How do these predictions compare to the observed data?

```
# data density
model_df_llo %>%
  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())
```

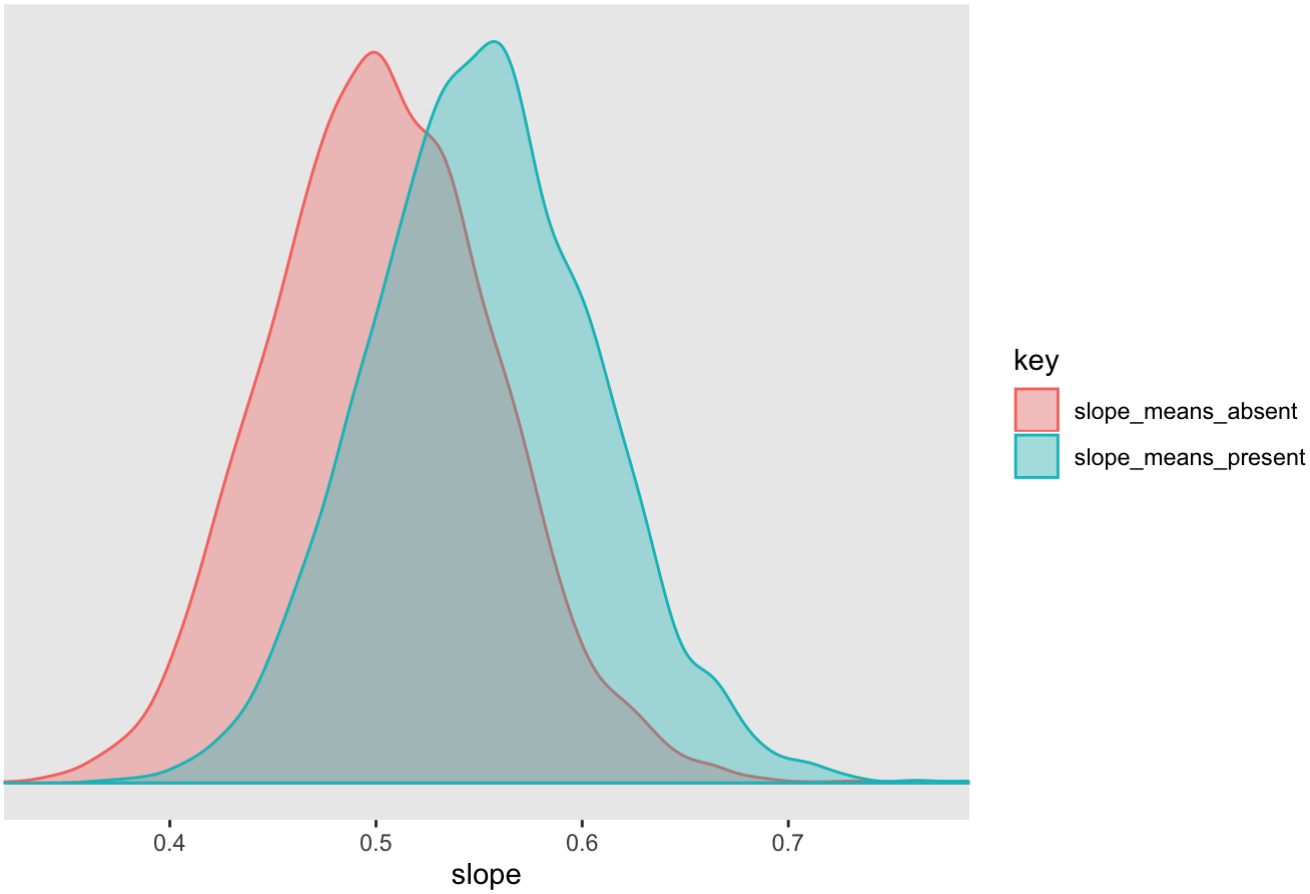
Data distribution for probability of superiority



What does the posterior for the slope look like when means are present vs absent?

```
# use posterior samples to define distributions for slope
posterior_samples(m.wrkr.means.llo_p_sup) %>%
  transmute(slope_means_present = `b_lo_ground_truth` +`b_lo_ground_truth:meansTRUE`,
            slope_means_absent = `b_lo_ground_truth`) %>%
  gather(key, value) %>%
  ggplot(aes(x = value, group = key, color = key, fill = key)) +
  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 present/absent") +
  theme(panel.grid = element_blank())
```

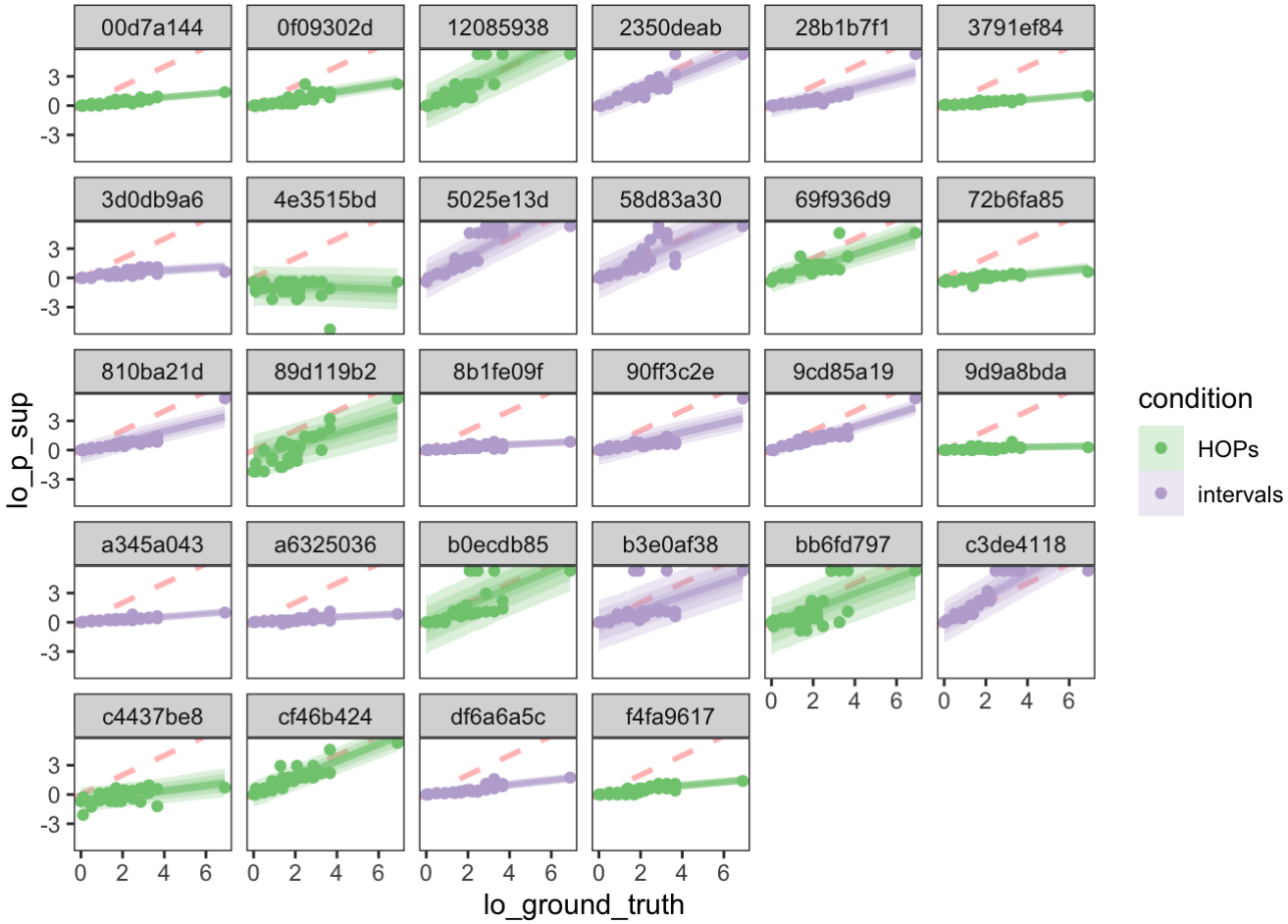
Posterior for slopes for means present/absent



Recall that a slope of 1 reflects zero bias. This suggests that users are biased toward responses of 50% regardless of the presence of extrinsic means.

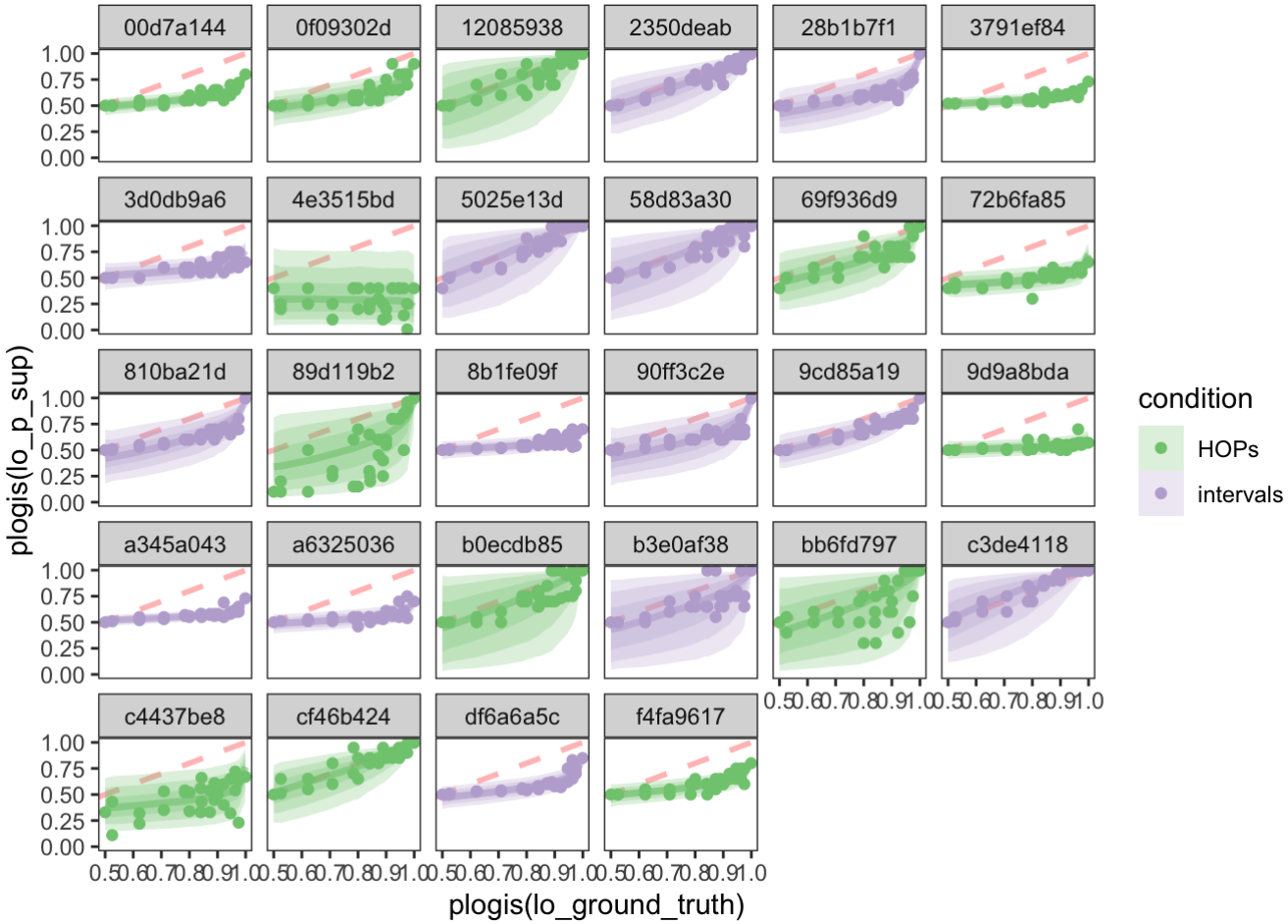
Let's take a look at predictions per worker and visualization condition to get a more granular sense of our model fit.

```
model_df_llo %>%
  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_llo) +
  scale_fill_brewer(type = "qual", palette = 1) +
  scale_color_brewer(type = "qual", palette = 1) +
  coord_cartesian(xlim = quantile(model_df_llo$lo_ground_truth, c(0, 1)),
                  ylim = quantile(model_df_llo$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_llo %>%
  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_llo) +
  scale_fill_brewer(type = "qual", palette = 1) +
  scale_color_brewer(type = "qual", palette = 1) +
  coord_cartesian(xlim = quantile(plogis(model_df_llo$lo_ground_truth), c(0, 1)),
                  ylim = quantile(plogis(model_df_llo$lo_p_sup), c(0, 1))) +
  theme_bw() +
  theme(panel.grid = element_blank()) +
  facet_wrap(~ worker_id)
```



Fixed Effects per 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. We will remove the effect of the mean for this model, and then add it to the next one.

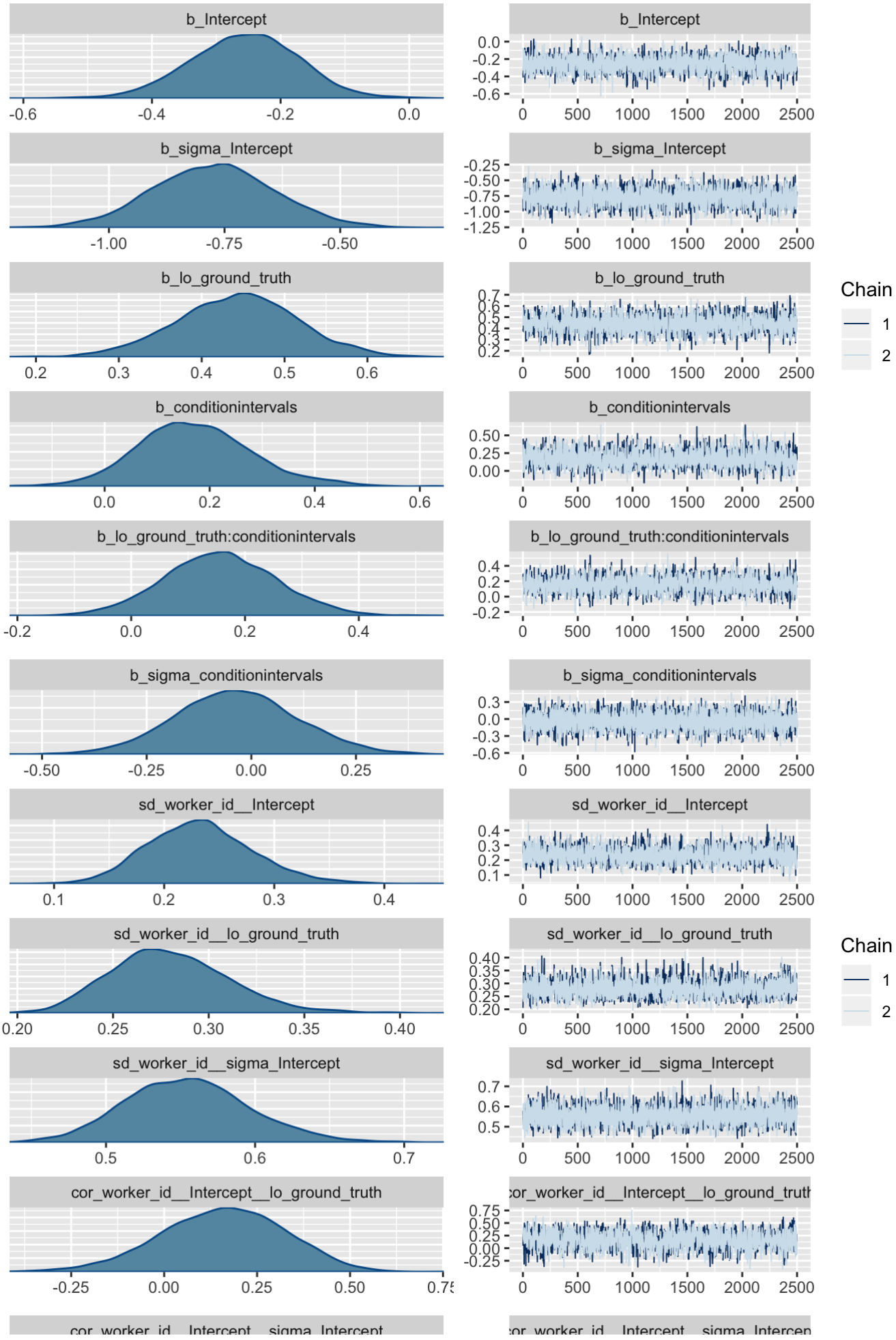
We use the same priors as we did for the previous model. Now, let's fit the model to our data.

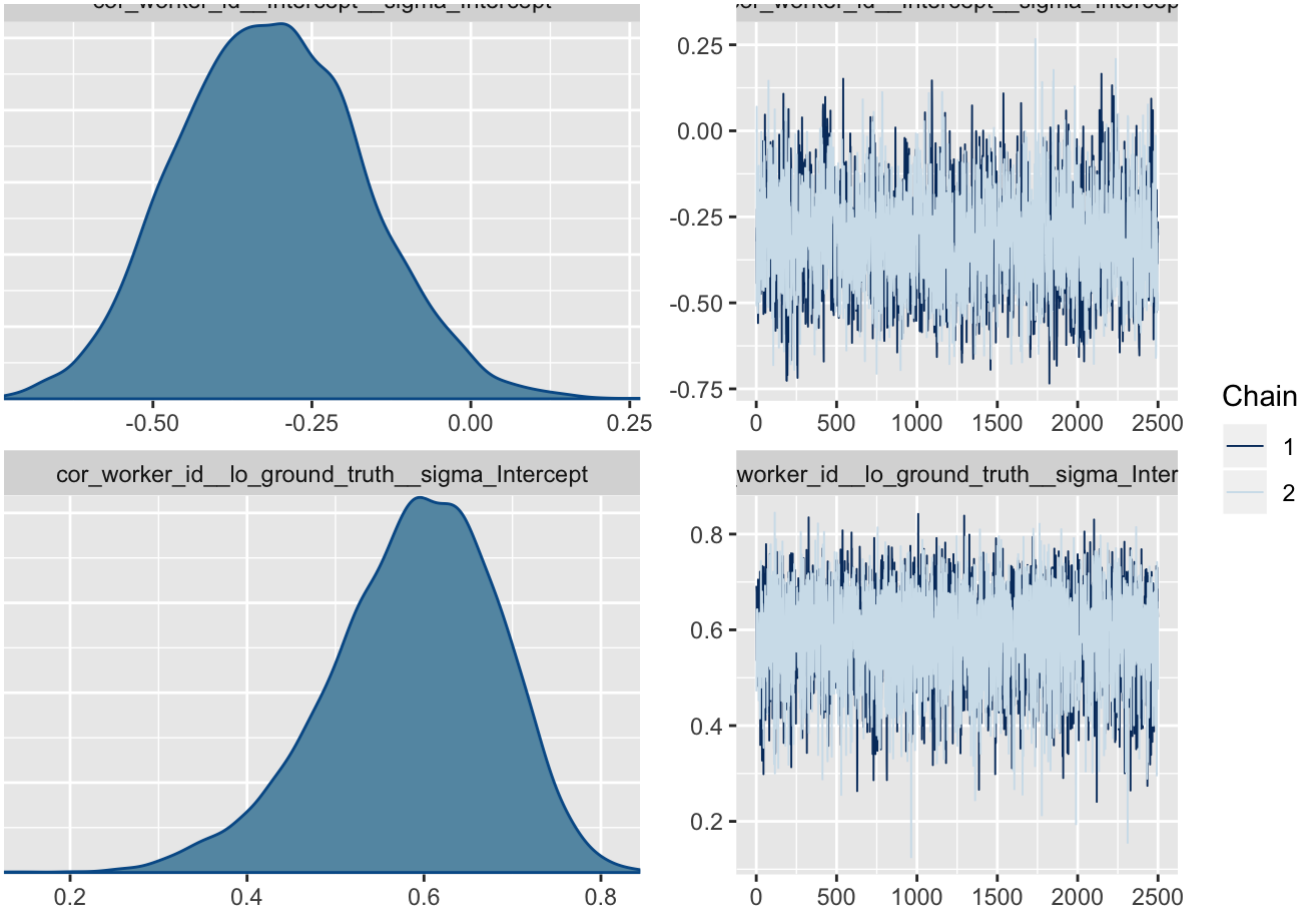
```
# hierarchical LLO model with fixed effects on slope and residual variance per visualiza
tion condition
m.wrkr.vis.llo_p_sup <- brm(data = model_df_llo, family = "gaussian",
                            formula = bf(lo_p_sup ~  (1 + lo_ground_truth|sharecor|worke
r_id) + lo_ground_truth*condition,
                                         sigma ~ (1|sharecor|worker_id) + condition),
                            prior = c(prior(normal(1, 0.5), class = b),
                                      prior(normal(1.96, 1), class = Intercept),
                                      prior(normal(0, 0.1), class = sd, group = worker_i
d),
                                      prior(normal(0, 0.2), class = b, dpar = sigma), #
 added prior
                                      prior(normal(0, 0.1), 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_cens-wrkr_vis")
```

Check diagnostics:

- Trace plots

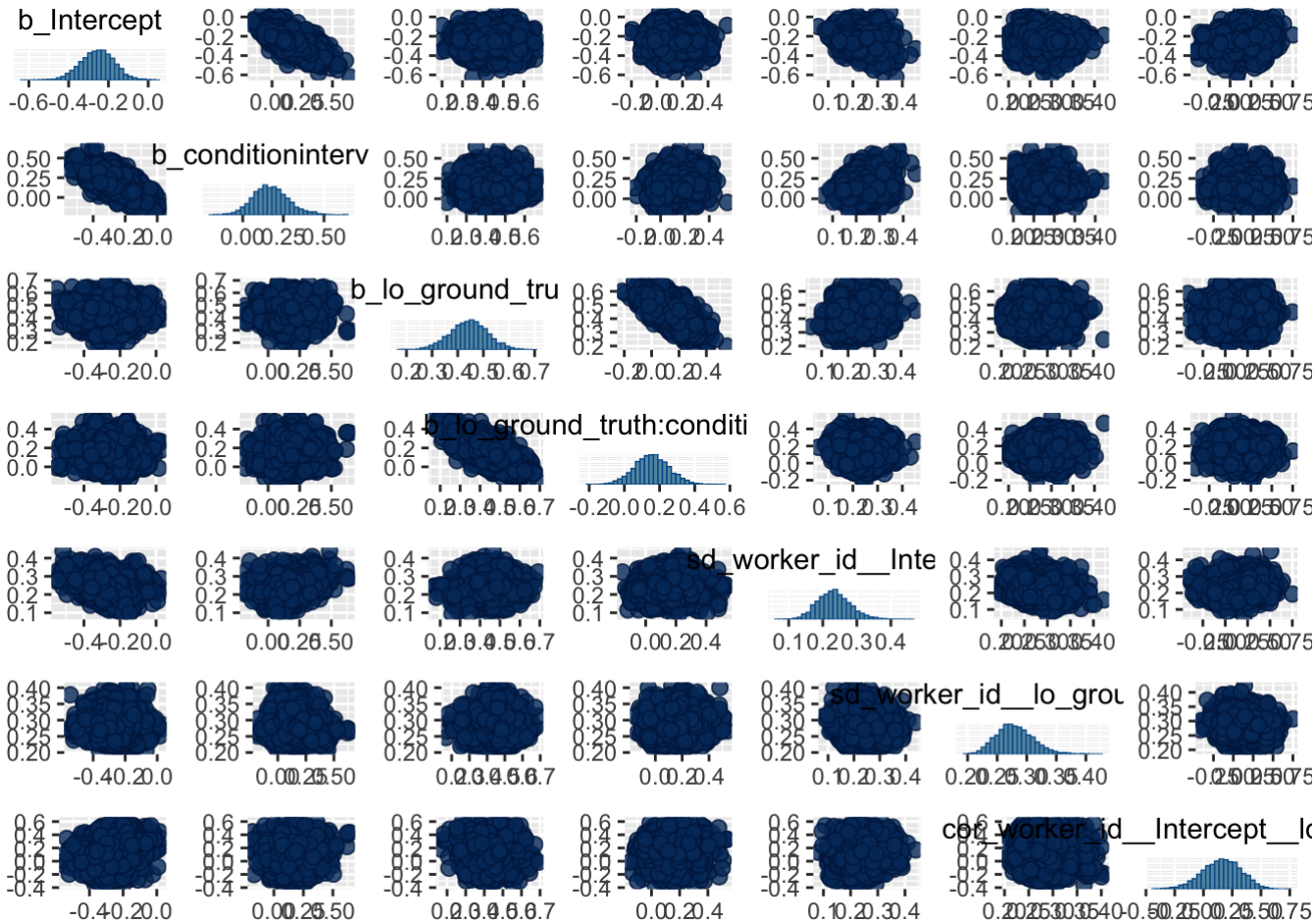
```
# trace plots
plot(m.wrkr.vis.llo_p_sup)
```


b_Intercept
b_sigma_Intercept
b_lo_ground_truth
b_conditionintervals
b_lo_ground_truth:conditionintervals
b_sigma_conditionintervals
sd_worker_id__Intercept
sd_worker_id__lo_ground_truth
sd_worker_id__sigma_Intercept
cor_worker_id__Intercept__lo_ground_truth
Chain
1
2

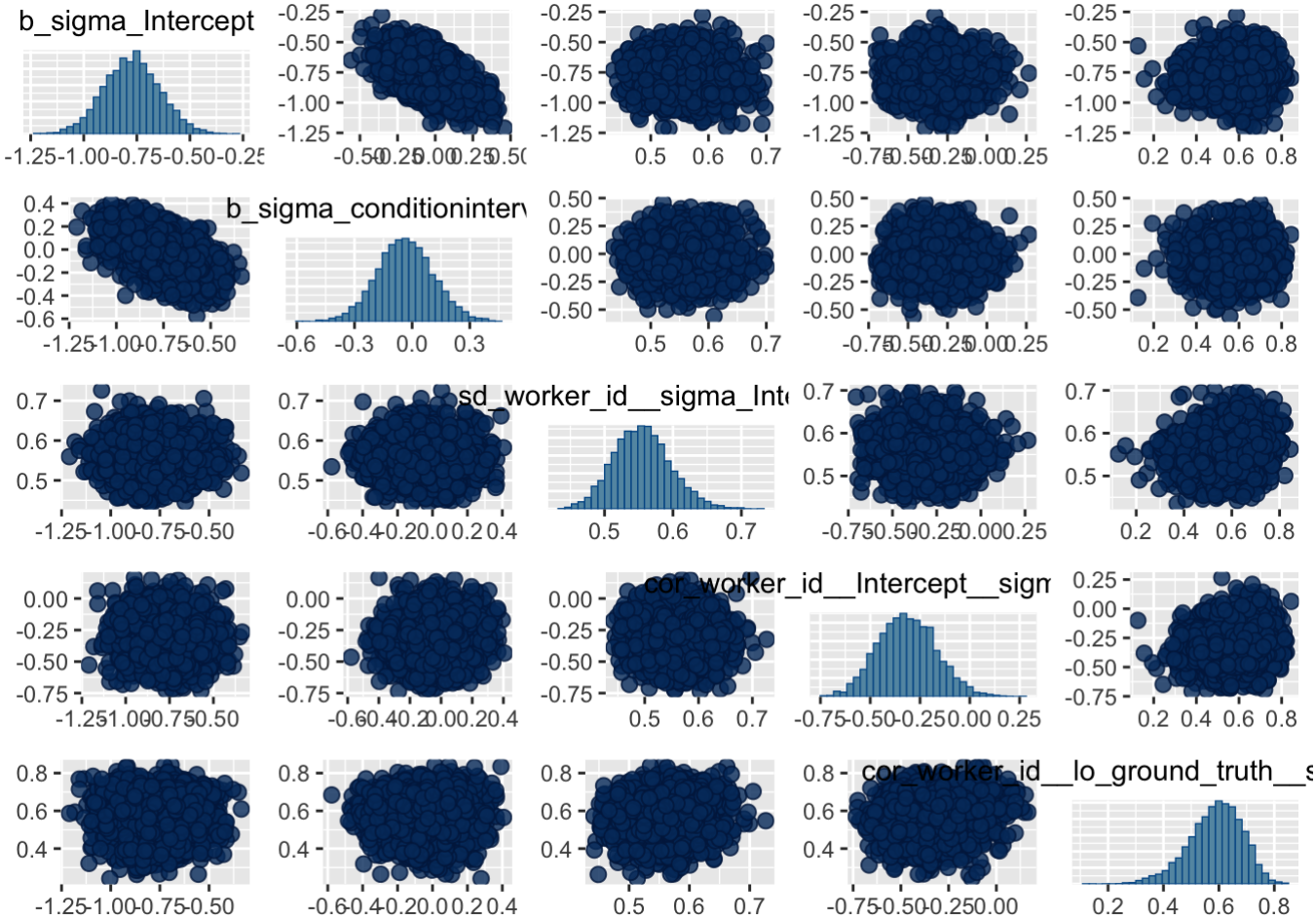


- Pairs plot

```
# pairs plot (LLO params)
pairs(m.wrkr.vis.llo_p_sup, exact_match = TRUE, pars = c("b_Intercept", "b_conditioninte
rvals", "b_lo_ground_truth", "b_lo_ground_truth:conditionintervals", "sd_worker_id__Inte
rcept", "sd_worker_id__lo_ground_truth", "cor_worker_id__Intercept__lo_ground_truth"))
```




```
# pairs plot (sigma params)
pairs(m.wrkr.vis.llo_p_sup, exact_match = TRUE, pars = c("b_sigma_Intercept", "b_sigma_c
onditionintervals", "sd_worker_id__sigma_Intercept", "cor_worker_id__Intercept__sigma_In
tercept", "cor_worker_id__lo_ground_truth__sigma_Intercept"))
```



- Summary

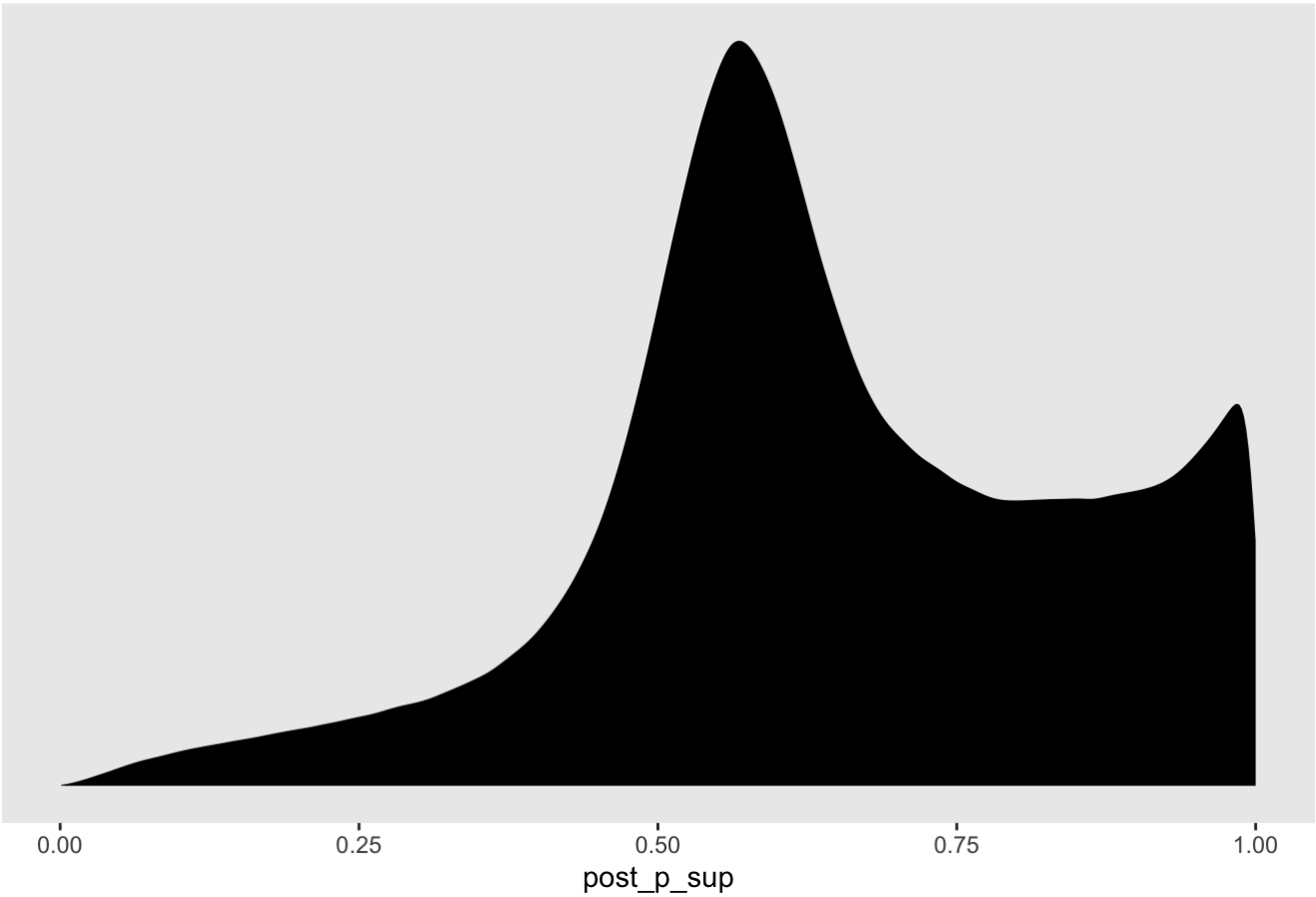
```
# model summary
print(m.wrkr.vis.llo_p_sup)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = log
## Formula: lo_p_sup ~ (1 + lo_ground_truth | sharecor | worker_id) + lo_ground_truth *
condition
##          sigma ~ (1 | sharecor | worker_id) + condition
##    Data: model_df_llo (Number of observations: 840)
## Samples: 2 chains, each with iter = 3000; warmup = 500; thin = 1;
##          total post-warmup samples = 5000
##
## Group-Level Effects:
## ~worker_id (Number of levels: 28)
##                                     Estimate Est.Error l-95% CI u-95% CI
## sd(Intercept)                           0.23      0.05     0.15     0.33
## sd(lo_ground_truth)                     0.28      0.03     0.23     0.35
## sd(sigma_Intercept)                     0.55      0.04     0.48     0.64
## cor(Intercept,lo_ground_truth)          0.16      0.17    -0.18     0.46
## cor(Intercept,sigma_Intercept)         -0.31      0.15    -0.59    -0.02
## cor(lo_ground_truth,sigma_Intercept)    0.59      0.10     0.37     0.75
##                                     Eff.Sample Rhat
## sd(Intercept)                             2258 1.00
## sd(lo_ground_truth)                       2041 1.00
## sd(sigma_Intercept)                       3991 1.00
## cor(Intercept,lo_ground_truth)            1221 1.00
## cor(Intercept,sigma_Intercept)            2041 1.00
## cor(lo_ground_truth,sigma_Intercept)       4303 1.00
##
## Population-Level Effects:
##                                   Estimate Est.Error l-95% CI u-95% CI
## Intercept                            -0.26      0.08    -0.42    -0.09
## sigma_Intercept                      -0.77      0.13    -1.03    -0.51
## lo_ground_truth                       0.44      0.07     0.30     0.58
## conditionintervals                    0.17      0.11    -0.03     0.41
## lo_ground_truth:conditionintervals    0.16      0.10    -0.03     0.35
## sigma_conditionintervals             -0.04      0.15    -0.32     0.26
##                                   Eff.Sample Rhat
## Intercept                               2691 1.00
## sigma_Intercept                         2845 1.00
## lo_ground_truth                         1878 1.00
## conditionintervals                      2512 1.00
## lo_ground_truth:conditionintervals      1902 1.00
## sigma_conditionintervals                3765 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_llo %>%
  select(lo_ground_truth, worker_id, condition) %>%
  add_predicted_draws(m.wrkr.vis.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())
```

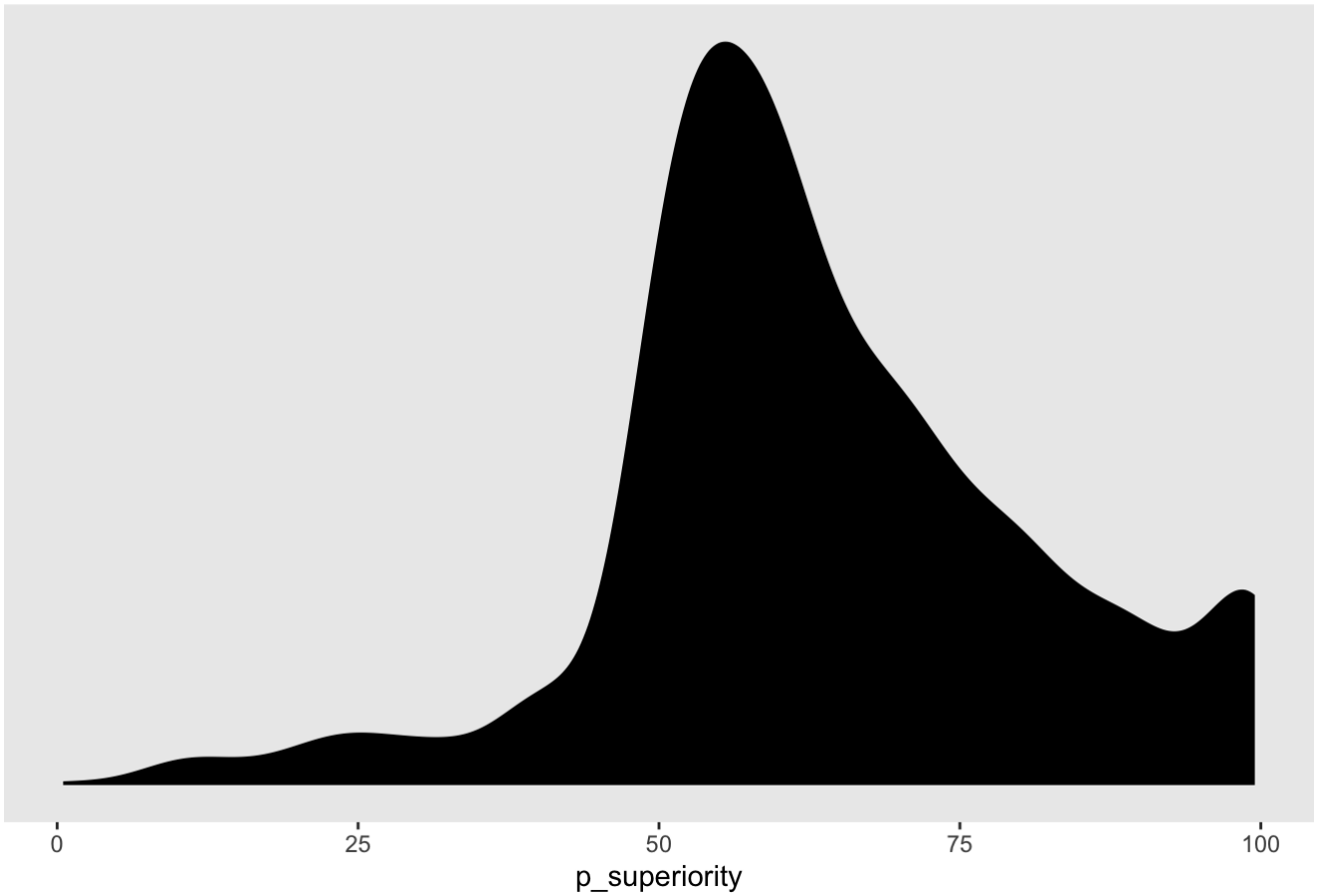
Posterior predictive distribution for probability of superiority



How do these predictions compare to the observed data?

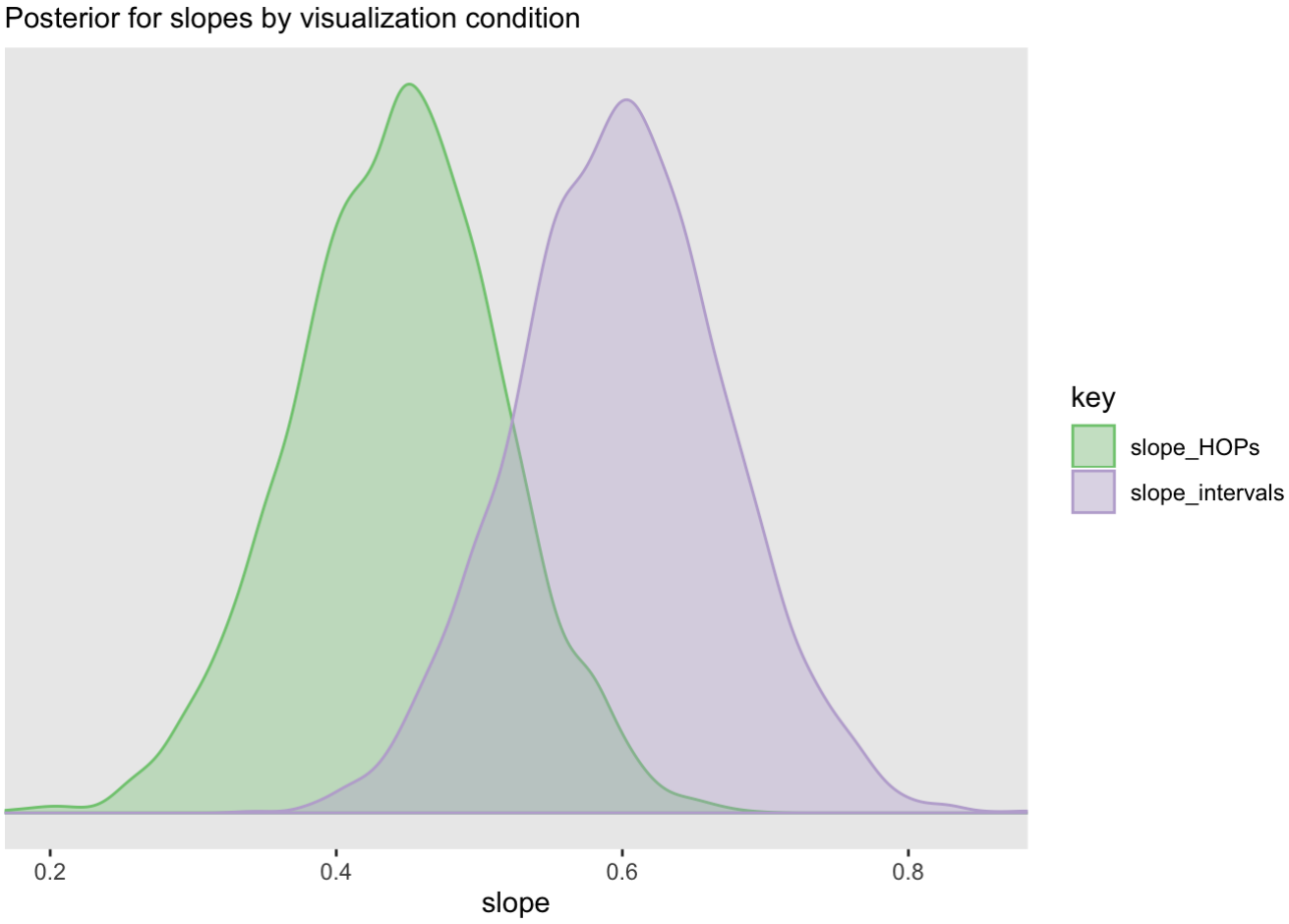
```
# data density
model_df_llo %>%
  ggplot(aes(x = p_superiority)) +
  geom_density(fill = "black", size = 0) +
  scale_y_continuous(NULL, breaks = NULL) +
  labs(subtitle = "Data distribution for probability of superiority") +
  theme(panel.grid = element_blank())
```

Data distribution for probability of superiority



What does the posterior for the slope in each visualization condition look like?

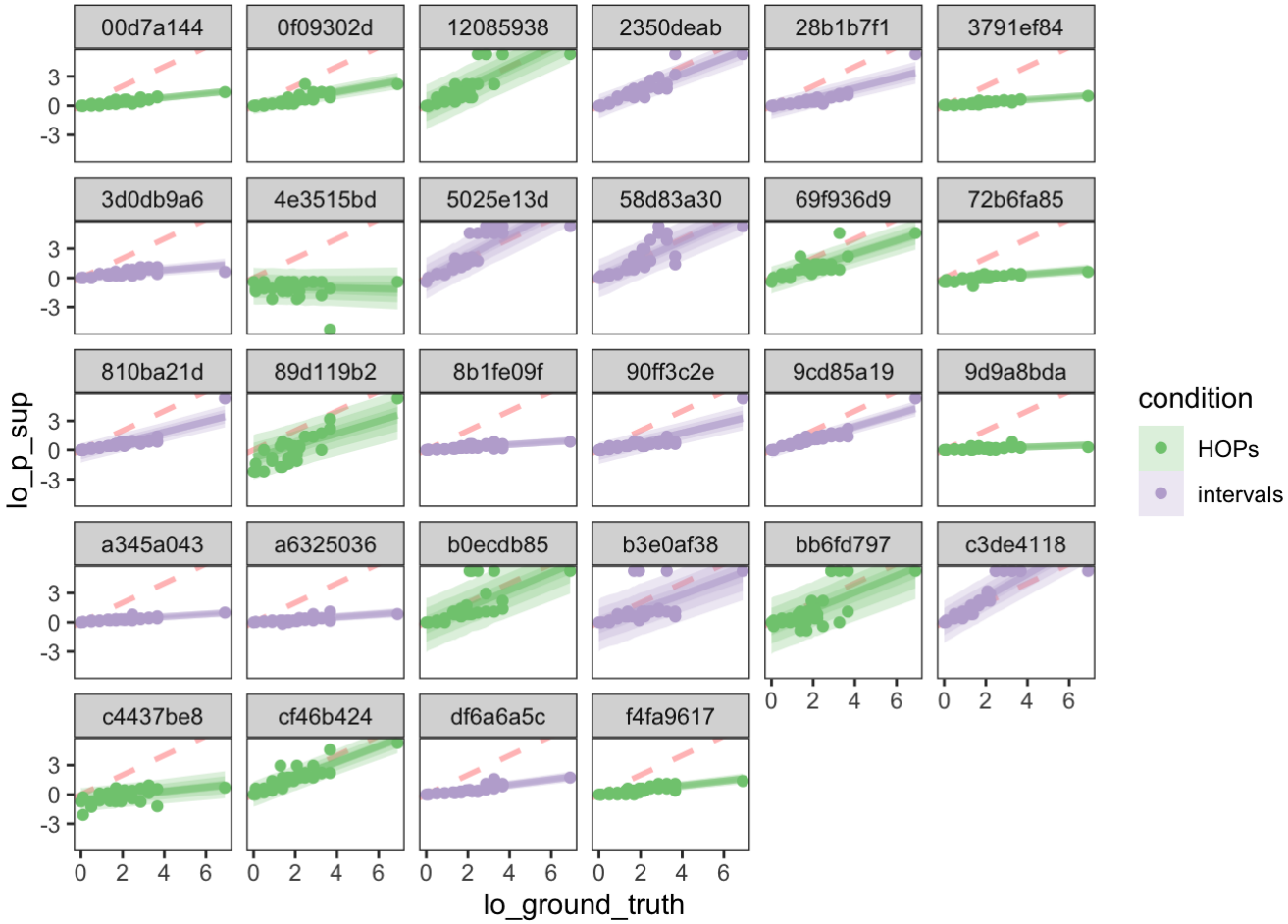
```
# use posterior samples to define distributions for the slope in each visualization cond
ition
posterior_samples(m.wrkr.vis.llo_p_sup) %>%
  transmute(slope_HOPs = `b_lo_ground_truth`,
            slope_intervals = `b_lo_ground_truth` + `b_lo_ground_truth:conditioninterval
s`) %>%
  gather(key, value) %>%
  ggplot(aes(x = value, group = key, color = key, fill = key)) +
  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 reflects zero bias. This suggests that users are biased toward responses of 50% in both conditions but moreso with HOPs ignoring the presence/absence of the mean.

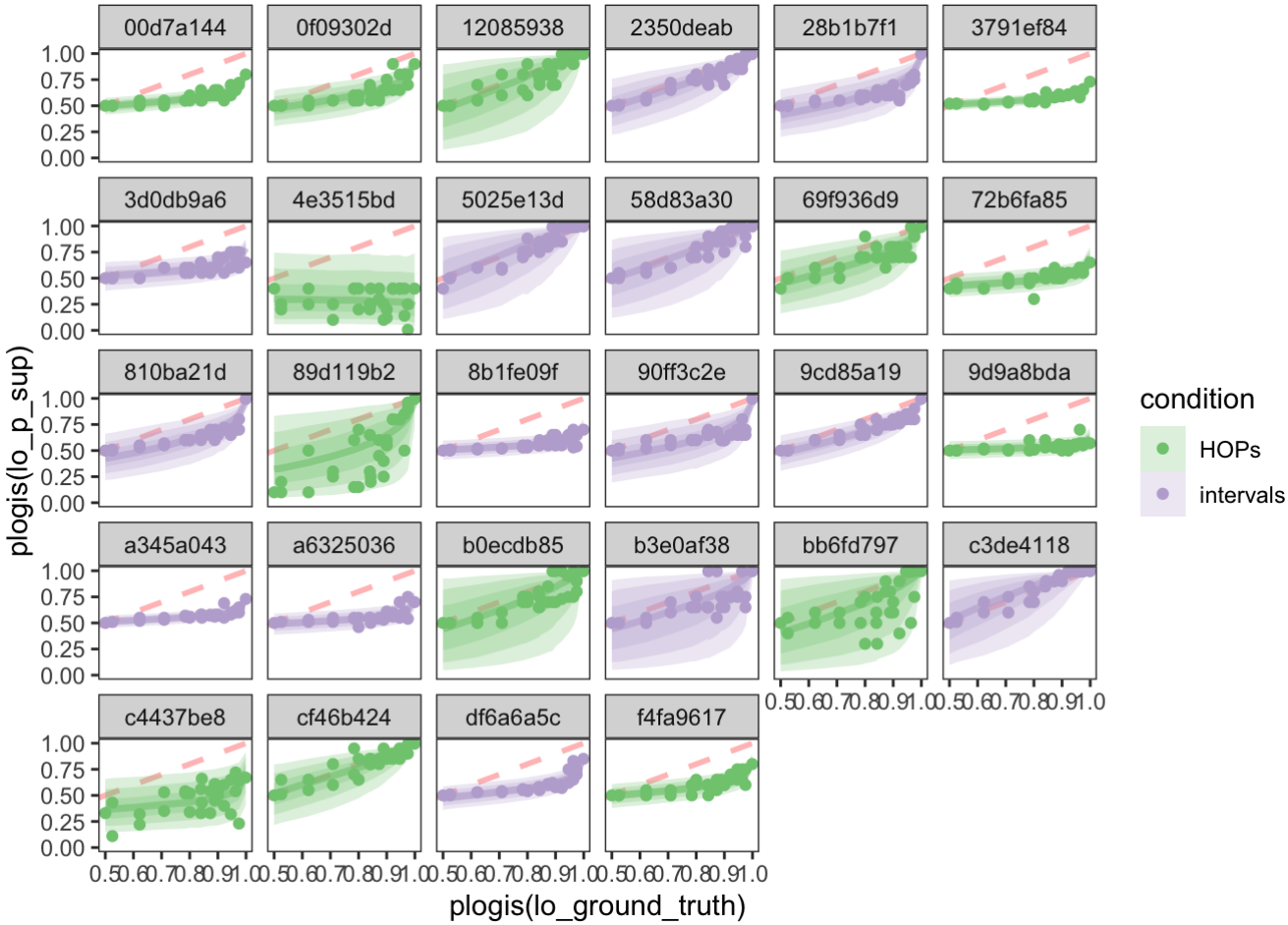
Let's take a look at predictions per worker and visualization condition to get a more granular sense of our model fit.

```
model_df_llo %>%
  group_by(lo_ground_truth, worker_id, condition) %>%
  add_predicted_draws(m.wrkr.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_llo) +
  scale_fill_brewer(type = "qual", palette = 1) +
  scale_color_brewer(type = "qual", palette = 1) +
  coord_cartesian(xlim = quantile(model_df_llo$lo_ground_truth, c(0, 1)),
                  ylim = quantile(model_df_llo$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_llo %>%
  group_by(lo_ground_truth, worker_id, condition) %>%
  add_predicted_draws(m.wrkr.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_llo) +
  scale_fill_brewer(type = "qual", palette = 1) +
  scale_color_brewer(type = "qual", palette = 1) +
  coord_cartesian(xlim = quantile(plogis(model_df_llo$lo_ground_truth), c(0, 1)),
                  ylim = quantile(plogis(model_df_llo$lo_p_sup), c(0, 1))) +
  theme_bw() +
  theme(panel.grid = element_blank()) +
  facet_wrap(~ worker_id)
```



Interaction Between Presence/Absence of the Mean and Visualization Condition

Let's put all our predictors into one model.

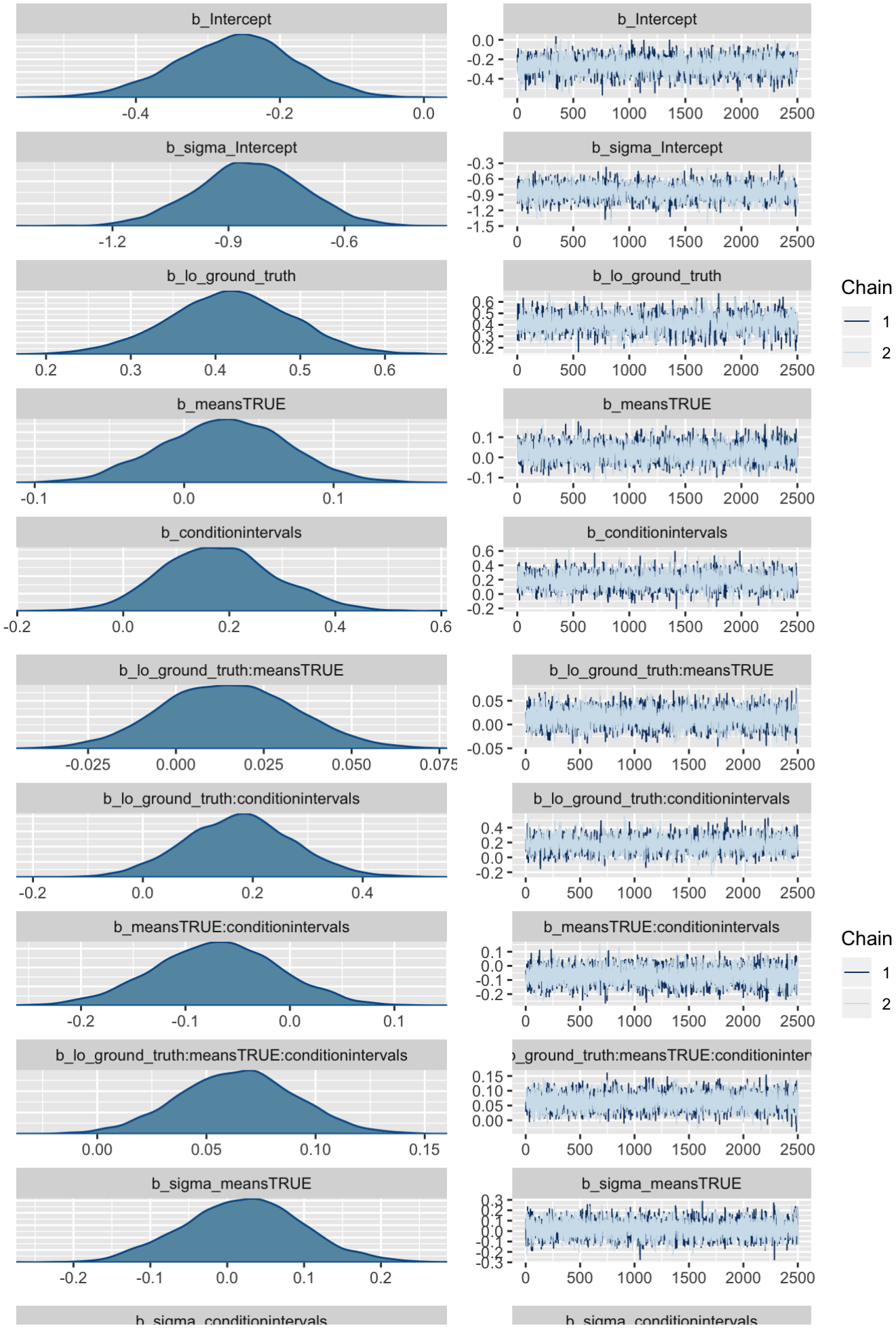
We use the same priors as we did for the previous model. Now, let's fit the model to our data.

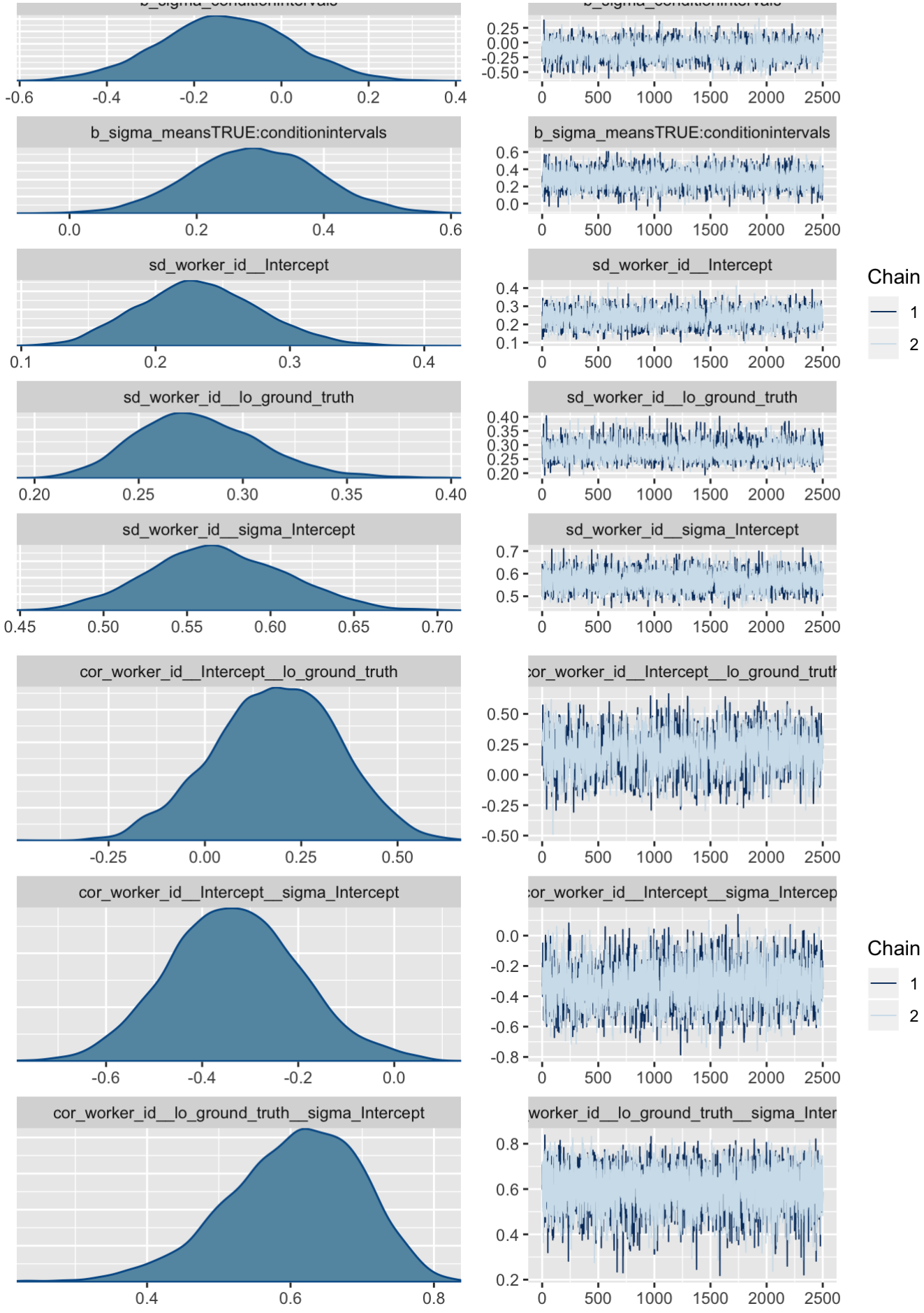
```
# hierarchical LLO model with fixed effects on slope and residual variance per visualiza
tion condition
m.wrkr.means.vis.llo_p_sup <- brm(data = model_df_llo, family = "gaussian",
                                  formula = bf(lo_p_sup ~  (1 + lo_ground_truth|sharecor
|worker_id) + lo_ground_truth*means*condition,
                                               sigma ~ (1|sharecor|worker_id) + means*co
ndition),
                                  prior = c(prior(normal(1, 0.5), class = b),
                                            prior(normal(1.96, 1), class = Intercept),
                                            prior(normal(0, 0.1), class = sd, group = wo
rker_id),
                                            prior(normal(0, 0.2), class = b, dpar = sigm
a), # added prior
                                            prior(normal(0, 0.1), class = sd, dpar = sig
ma),
                                            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_vis")
```

Check diagnostics:

- Trace plots

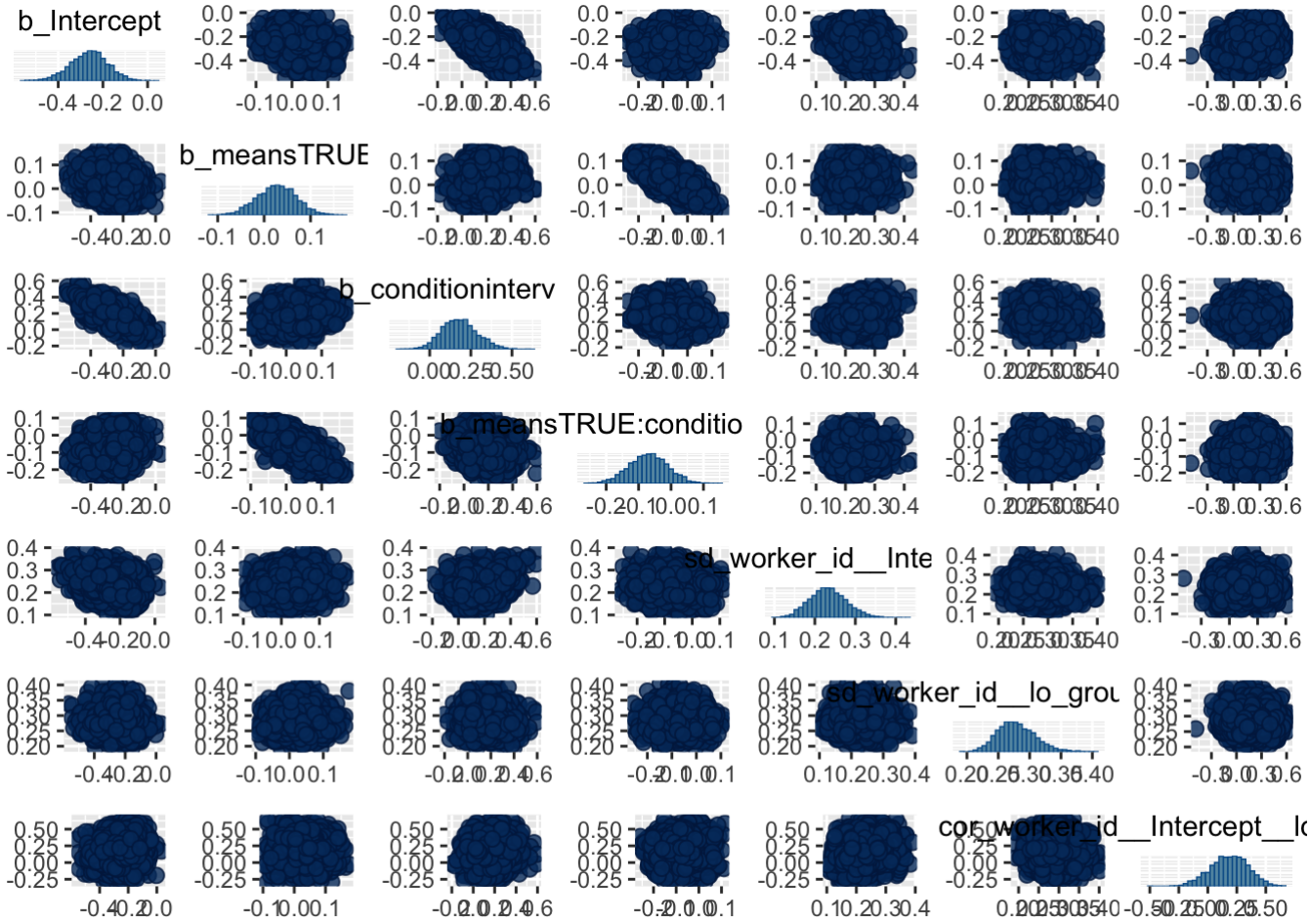
```
# trace plots
plot(m.wrkr.means.vis.llo_p_sup)
```


b_Intercept
b_sigma_Intercept
b_lo_ground_truth
b_meansTRUE
b_conditionintervals
b_lo_ground_truth:meansTRUE
b_lo_ground_truth:conditionintervals
b_meansTRUE:conditionintervals
b_lo_ground_truth:meansTRUE:conditionintervals
b_sigma_meansTRUE
b_sigma_conditionintervals
Chain
1
2

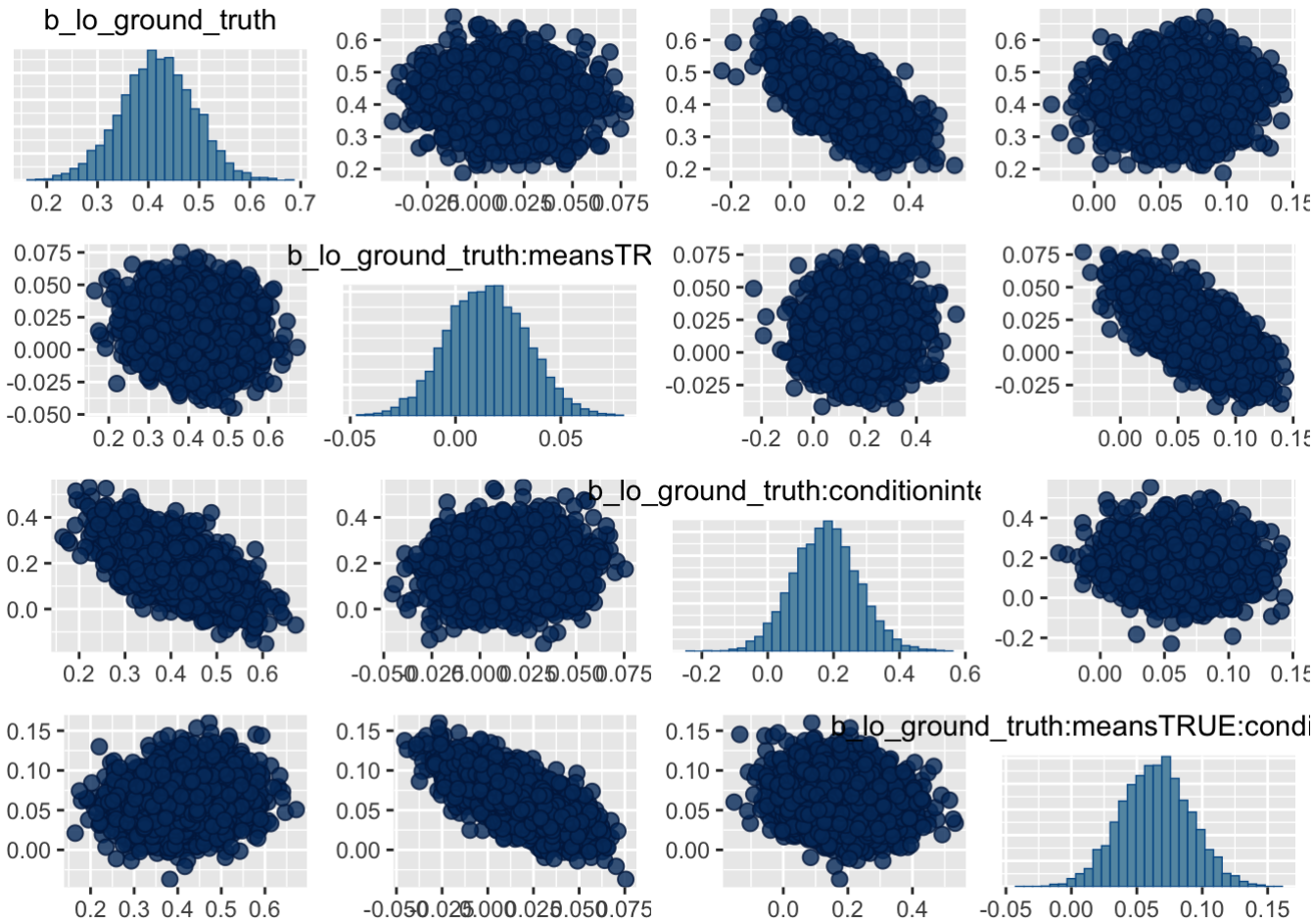

b_sigma_meansTRUE:conditionintervals
sd_worker_id__Intercept
sd_worker_id__lo_ground_truth
sd_worker_id__sigma_Intercept
cor_worker_id__Intercept__lo_ground_truth
cor_worker_id__Intercept__sigma_Intercept
cor_worker_id__lo_ground_truth__sigma_Intercept
Chain
1
2

- Pairs plot

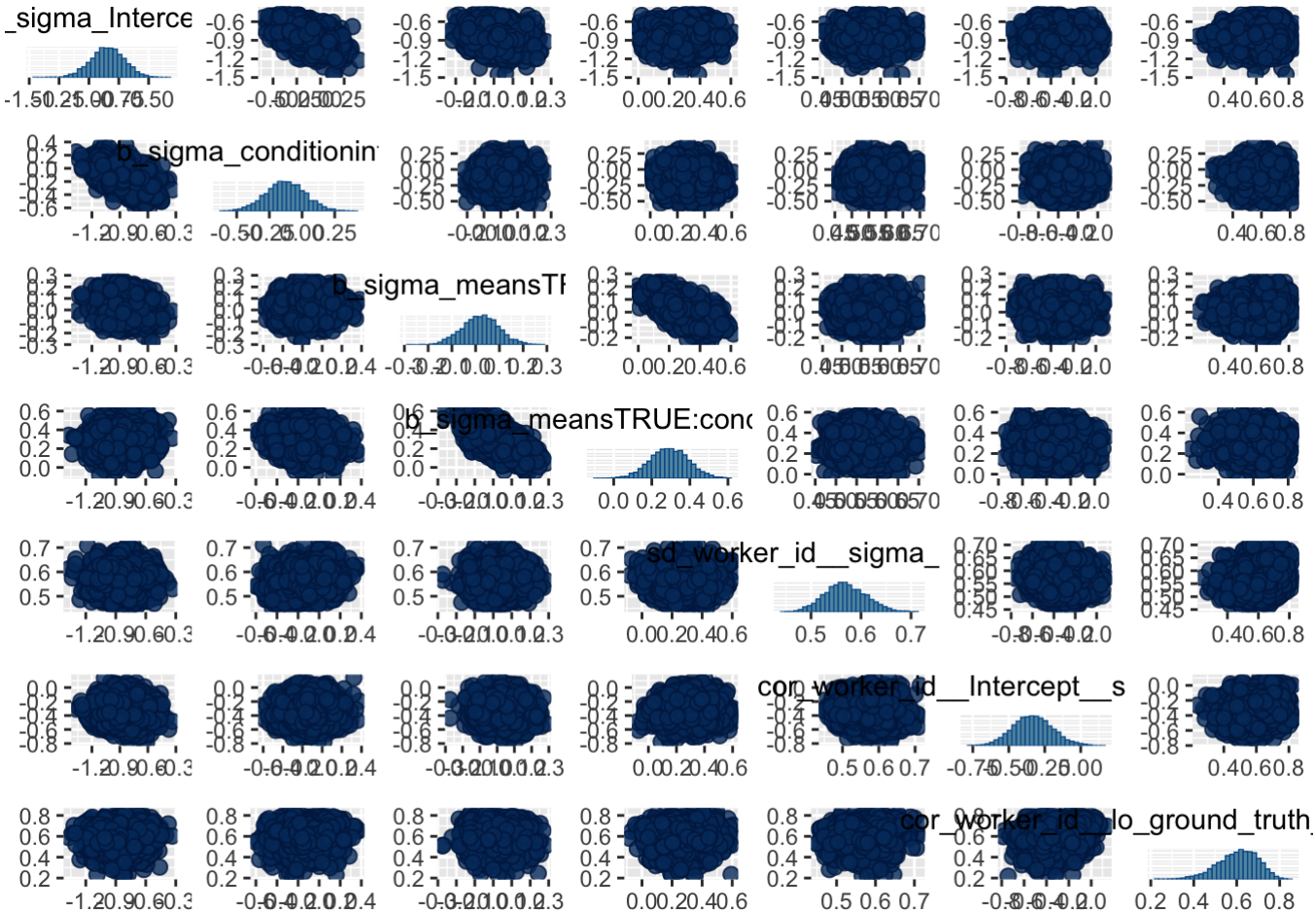
```
# pairs plot (LLO intercept params and random effects)
pairs(m.wrkr.means.vis.llo_p_sup, exact_match = TRUE, pars = c("b_Intercept", "b_meansTR
UE", "b_conditionintervals", "b_meansTRUE:conditionintervals", "sd_worker_id__Intercept"
, "sd_worker_id__lo_ground_truth", "cor_worker_id__Intercept__lo_ground_truth"))
```




```
# pairs plot (LLO slope params)
pairs(m.wrkr.means.vis.llo_p_sup, exact_match = TRUE, pars = c("b_lo_ground_truth", "b_l
o_ground_truth:meansTRUE", "b_lo_ground_truth:conditionintervals", "b_lo_ground_truth:me
ansTRUE:conditionintervals"))
```




```
# pairs plot (sigma params)
pairs(m.wrkr.means.vis.llo_p_sup, exact_match = TRUE, pars = c("b_sigma_Intercept", "b_s
igma_conditionintervals", "b_sigma_meansTRUE", "b_sigma_meansTRUE:conditionintervals",
"sd_worker_id__sigma_Intercept", "cor_worker_id__Intercept__sigma_Intercept", "cor_worke
r_id__lo_ground_truth__sigma_Intercept"))
```



We're still seeing multicollinearity for the effect of the presence/absence of the mean on slopes in the LLO model.

- Summary

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

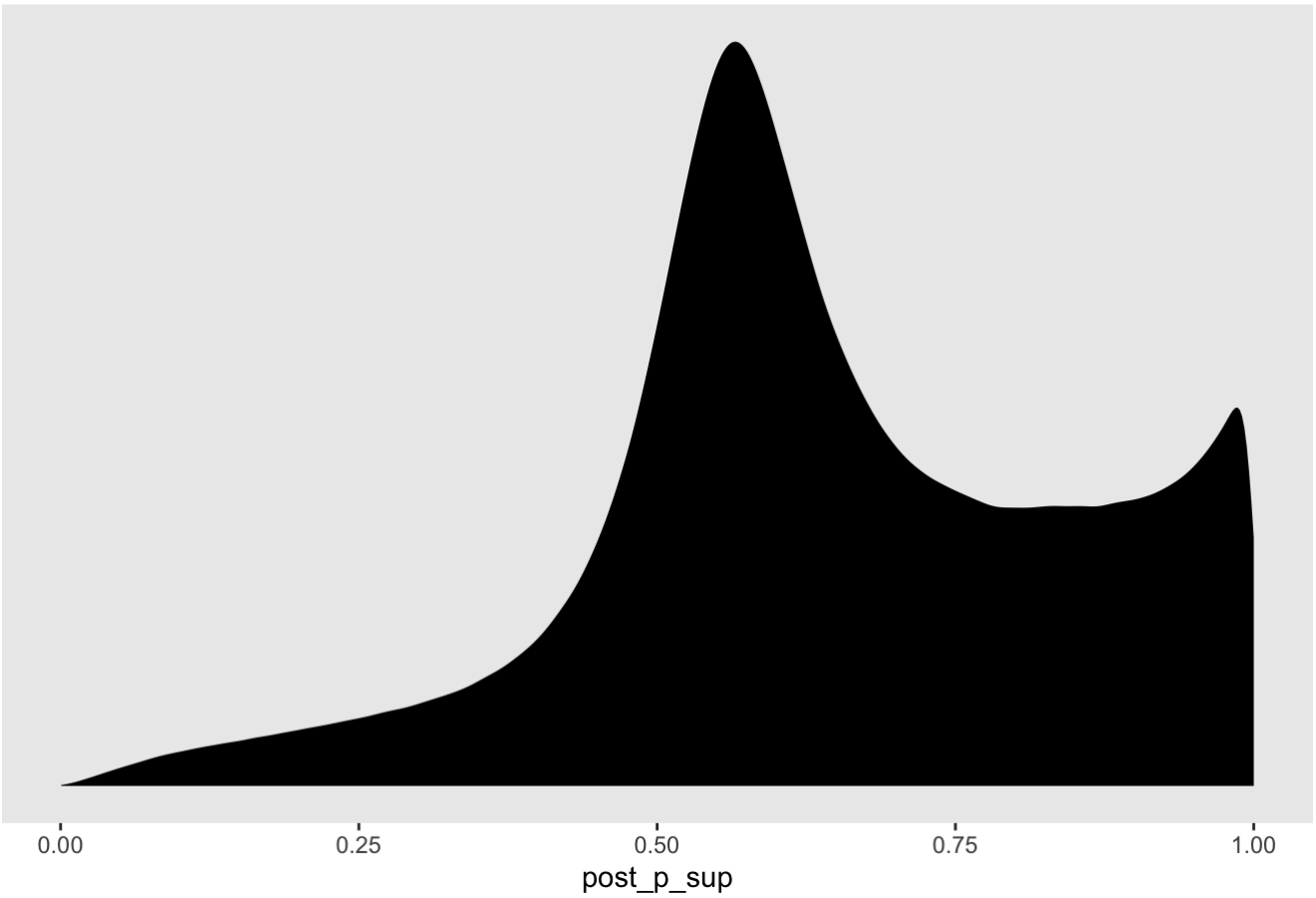
```
##  Family: gaussian
##   Links: mu = identity; sigma = log
## Formula: lo_p_sup ~ (1 + lo_ground_truth | sharecor | worker_id) + lo_ground_truth *
means * condition
##          sigma ~ (1 | sharecor | worker_id) + means * condition
##    Data: model_df_llo (Number of observations: 840)
## Samples: 2 chains, each with iter = 3000; warmup = 500; thin = 1;
##          total post-warmup samples = 5000
##
## Group-Level Effects:
## ~worker_id (Number of levels: 28)
##                                     Estimate Est.Error l-95% CI u-95% CI
## sd(Intercept)                           0.23      0.05     0.15     0.32
## sd(lo_ground_truth)                     0.28      0.03     0.23     0.35
## sd(sigma_Intercept)                     0.57      0.04     0.49     0.65
## cor(Intercept,lo_ground_truth)          0.18      0.16    -0.14     0.49
## cor(Intercept,sigma_Intercept)         -0.33      0.14    -0.59    -0.05
## cor(lo_ground_truth,sigma_Intercept)    0.60      0.09     0.40     0.76
##                                     Eff.Sample Rhat
## sd(Intercept)                             2754 1.00
## sd(lo_ground_truth)                       3011 1.00
## sd(sigma_Intercept)                       4408 1.00
## cor(Intercept,lo_ground_truth)            1613 1.00
## cor(Intercept,sigma_Intercept)            2567 1.00
## cor(lo_ground_truth,sigma_Intercept)       5325 1.00
##
## Population-Level Effects:
##                                           Estimate Est.Error l-95% CI
## Intercept                                    -0.26      0.08    -0.43
## sigma_Intercept                              -0.84      0.14    -1.12
## lo_ground_truth                               0.42      0.07     0.27
## meansTRUE                                     0.03      0.04    -0.05
## conditionintervals                            0.18      0.11    -0.02
## lo_ground_truth:meansTRUE                     0.02      0.02    -0.02
## lo_ground_truth:conditionintervals            0.18      0.10    -0.01
## meansTRUE:conditionintervals                 -0.07      0.06    -0.19
## lo_ground_truth:meansTRUE:conditionintervals   0.06      0.03     0.01
## sigma_meansTRUE                               0.02      0.08    -0.13
## sigma_conditionintervals                     -0.13      0.15    -0.42
## sigma_meansTRUE:conditionintervals            0.29      0.10     0.10
##                                           u-95% CI Eff.Sample Rhat
## Intercept                                    -0.11       3386 1.00
## sigma_Intercept                              -0.58       3289 1.00
## lo_ground_truth                               0.56       2287 1.00
## meansTRUE                                     0.11       5400 1.00
## conditionintervals                            0.40       3139 1.00
## lo_ground_truth:meansTRUE                     0.05       5301 1.00
## lo_ground_truth:conditionintervals            0.37       2298 1.00
## meansTRUE:conditionintervals                  0.04       5345 1.00
## lo_ground_truth:meansTRUE:conditionintervals  0.12       5547 1.00
## sigma_meansTRUE                               0.17       7456 1.00
## sigma_conditionintervals                      0.16       4441 1.00
## sigma_meansTRUE:conditionintervals            0.49       6799 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.

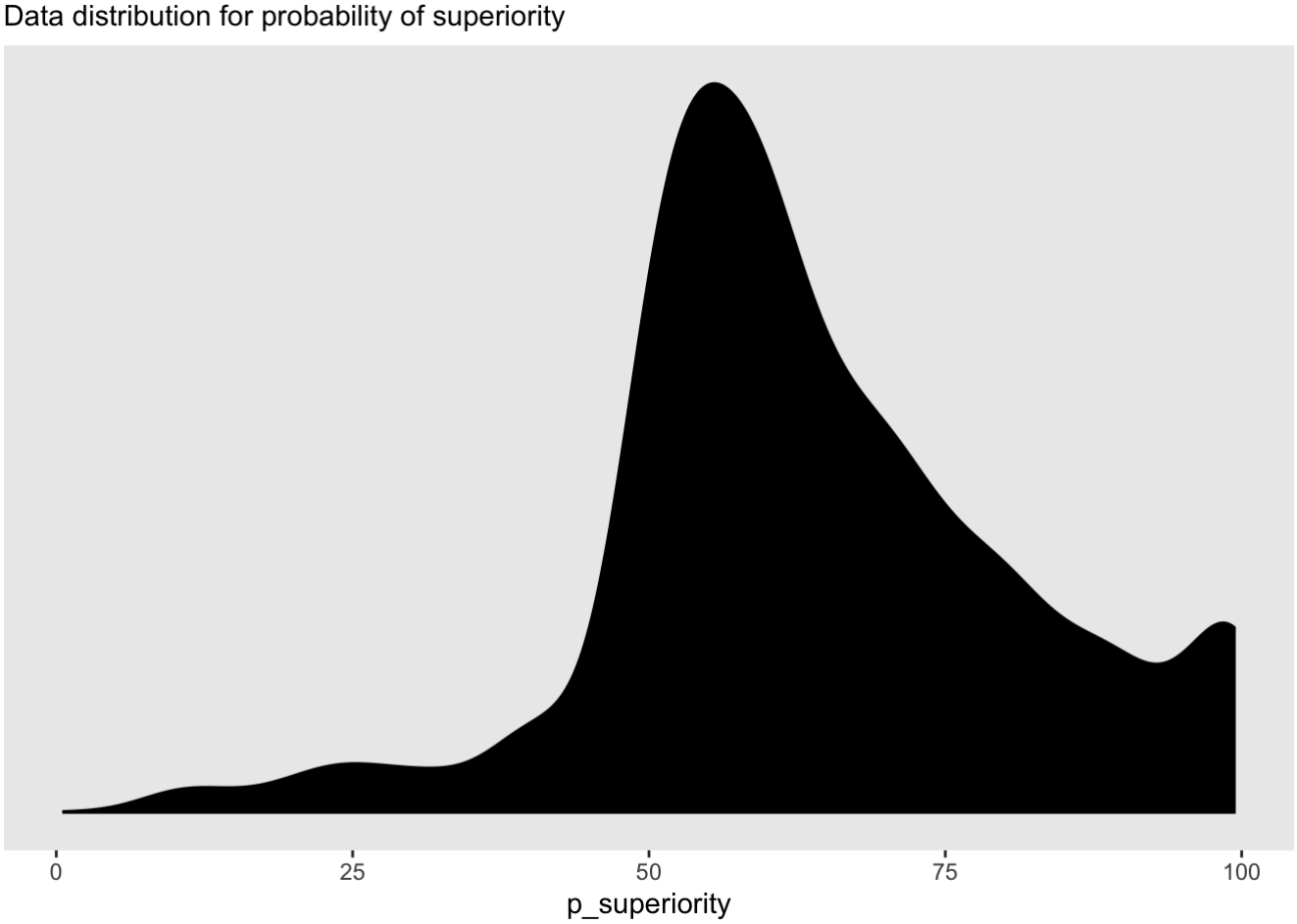
```
# posterior predictive check
model_df_llo %>%
  select(lo_ground_truth, worker_id, means, condition) %>%
  add_predicted_draws(m.wrkr.means.vis.llo_p_sup, prediction = "lo_p_sup", seed = 1234)
 %>%
  mutate(
    # transform to probability units
    post_p_sup = plogis(lo_p_sup)
  ) %>%
  ggplot(aes(x = 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



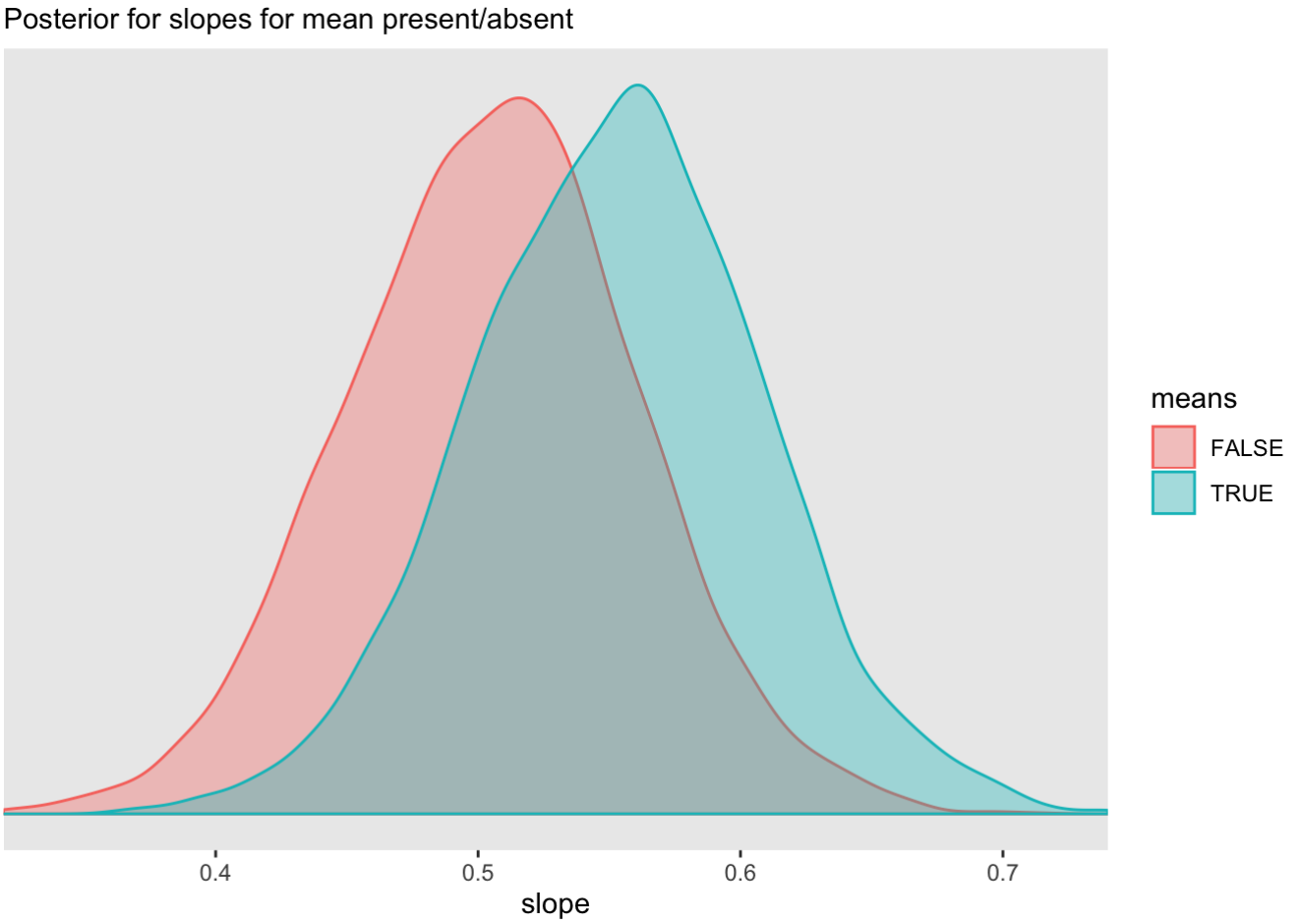
How do these predictions compare to the observed data?

```
# data density
model_df_llo %>%
  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())
```



What does the posterior for the slope look like when means are present vs absent, marginalizing across visualization conditions? Now that we have a more 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.

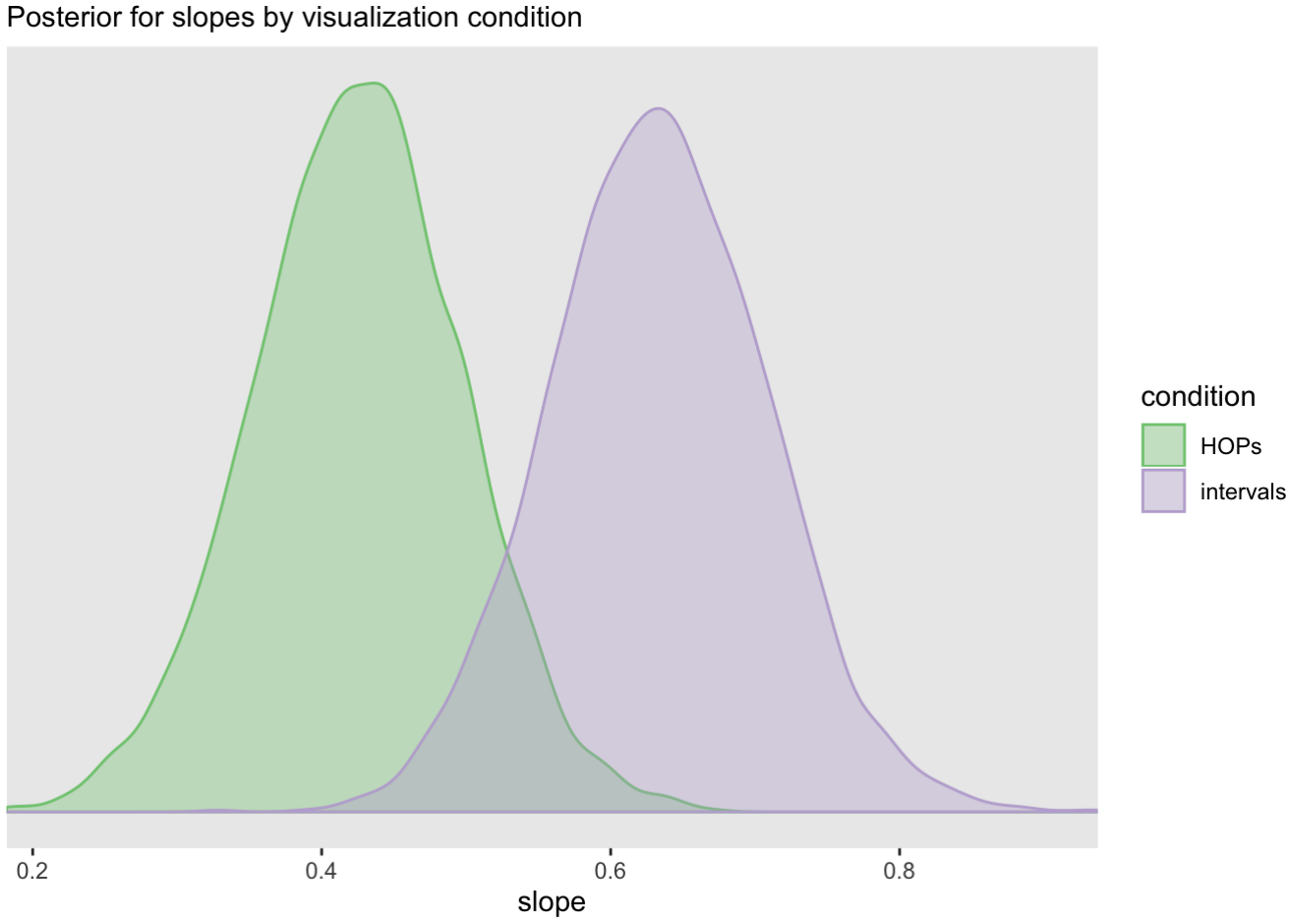
```
model_df_llo %>%
  group_by(means, condition) %>%
  data_grid(lo_ground_truth = c(0, 1)) %>%          # get fitted draws (in log odds units) only for ground truth of 0 and 1
  add_fitted_draws(m.wrkr.means.vis.llo_p_sup, re_formula = NA) %>%
  compare_levels(.value, by = lo_ground_truth) %>%  # calculate the difference between fits at 1 and 0 (i.e., slope)
  rename(slope = .value) %>%
  group_by(means, .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())
```



This effect suggests that adding means has a debiasing effect on average (marginalizing across visualization conditions), exactly the opposite of what we expected to see. This seems to be because the mean difference is a good heuristic for probability of superiority when variance is constant across visualized predictions (as it was in this pilot).

What does the posterior for the slope in each visualization condition look like, marginalizing across the presence/absence of the mean?

```
model_df_llo %>%
  group_by(means, 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.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) %>%
  group_by(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 = 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())
```

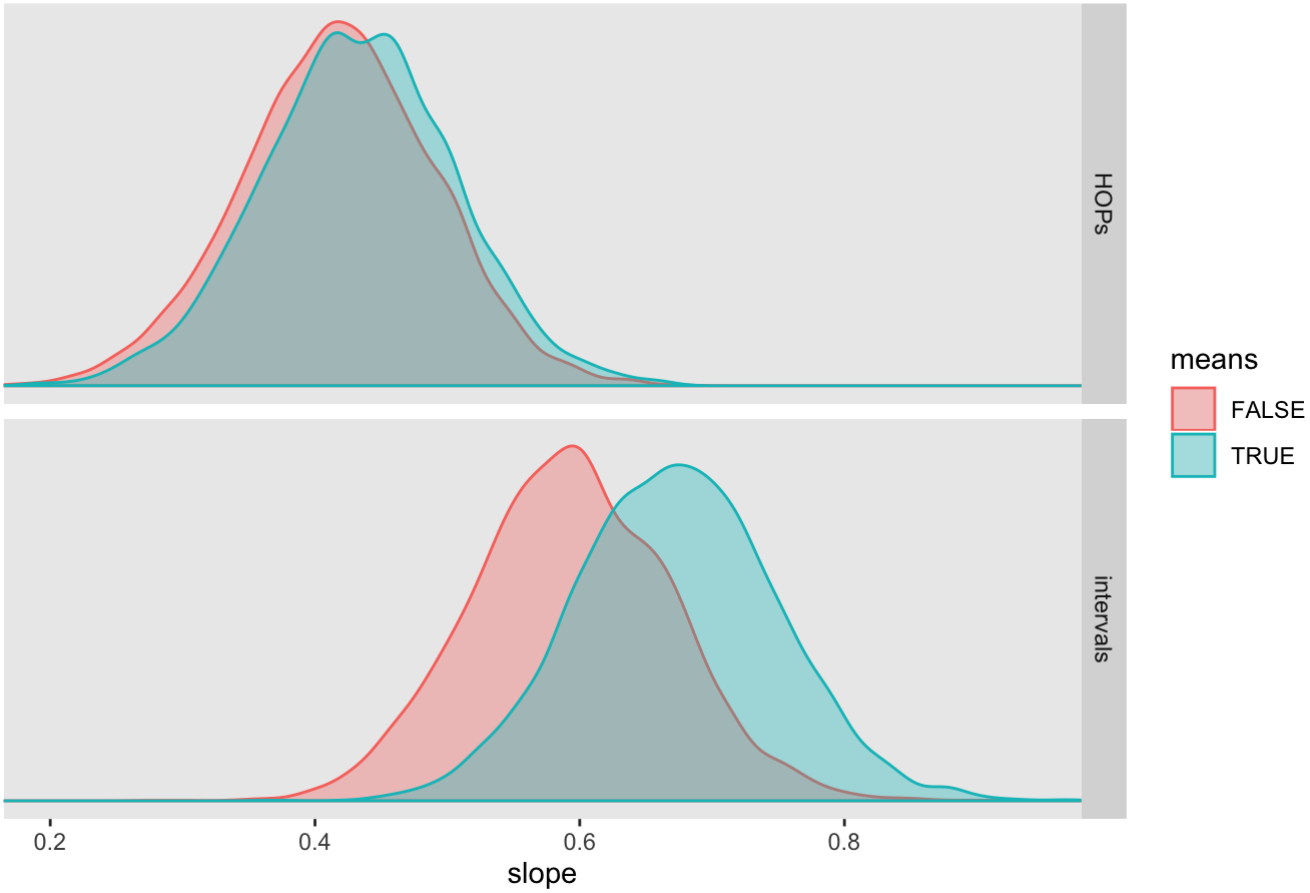


Recall that a slope of 1 on the logit scale reflects no bias. This suggests that users are biased toward responses of 50% on the probability scale in all conditions, but especially with HOPs. This is exactly the opposite of what we expected to see. Maybe this is because there are too many frames in HOPs for people to get a sense of the probability of superiority before they want to respond.

What if we break these marginal effects down into simple effects for the interaction of the presence/absence of the mean * visualization condition?

```
model_df_llo %>%
  group_by(means, 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.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 * visualization condition") +
  theme(panel.grid = element_blank()) +
  facet_grid(condition ~ .)
```

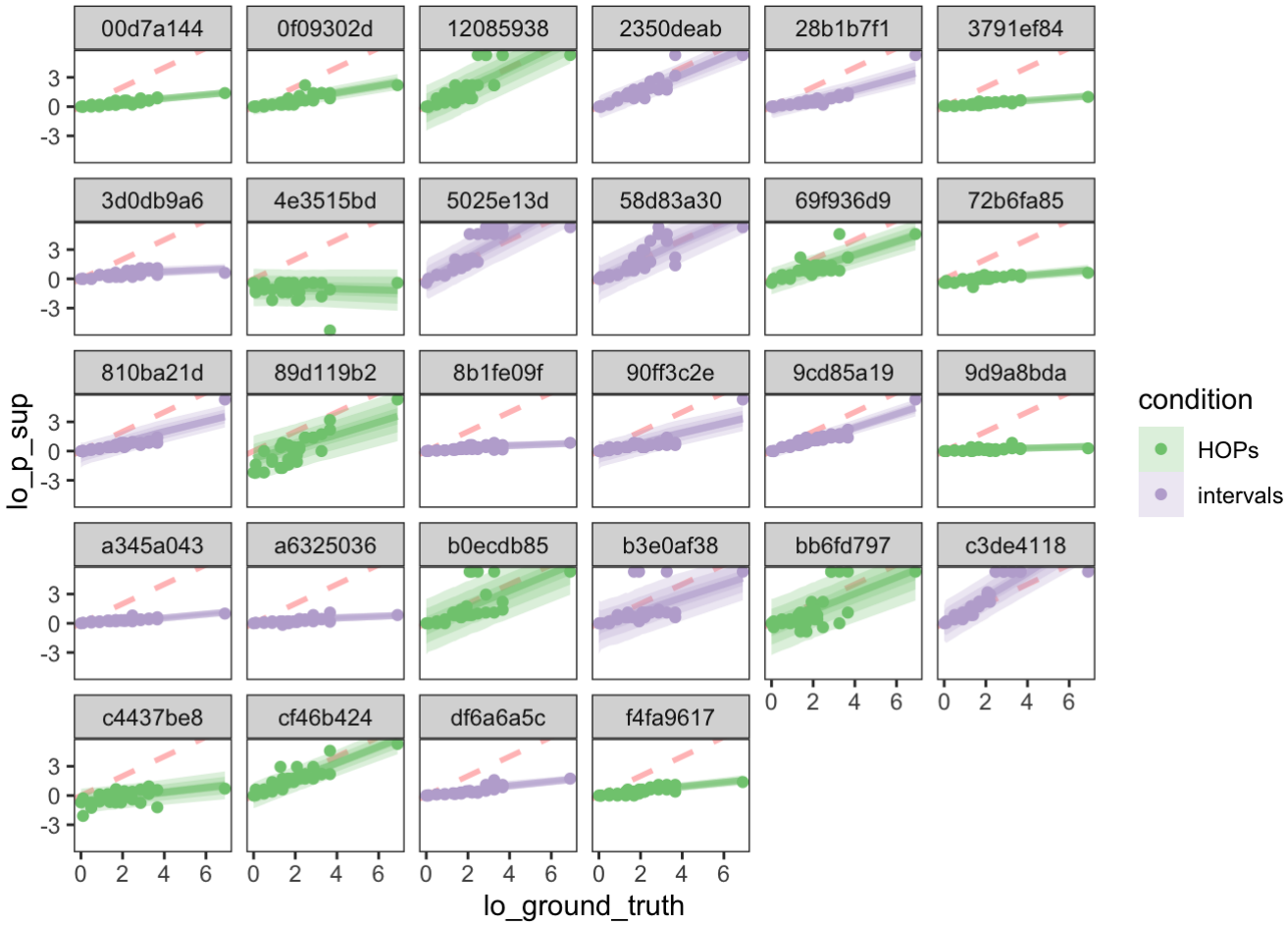
Posterior for slopes for means * visualization condition



Again, this is the opposite of what we expected to see.

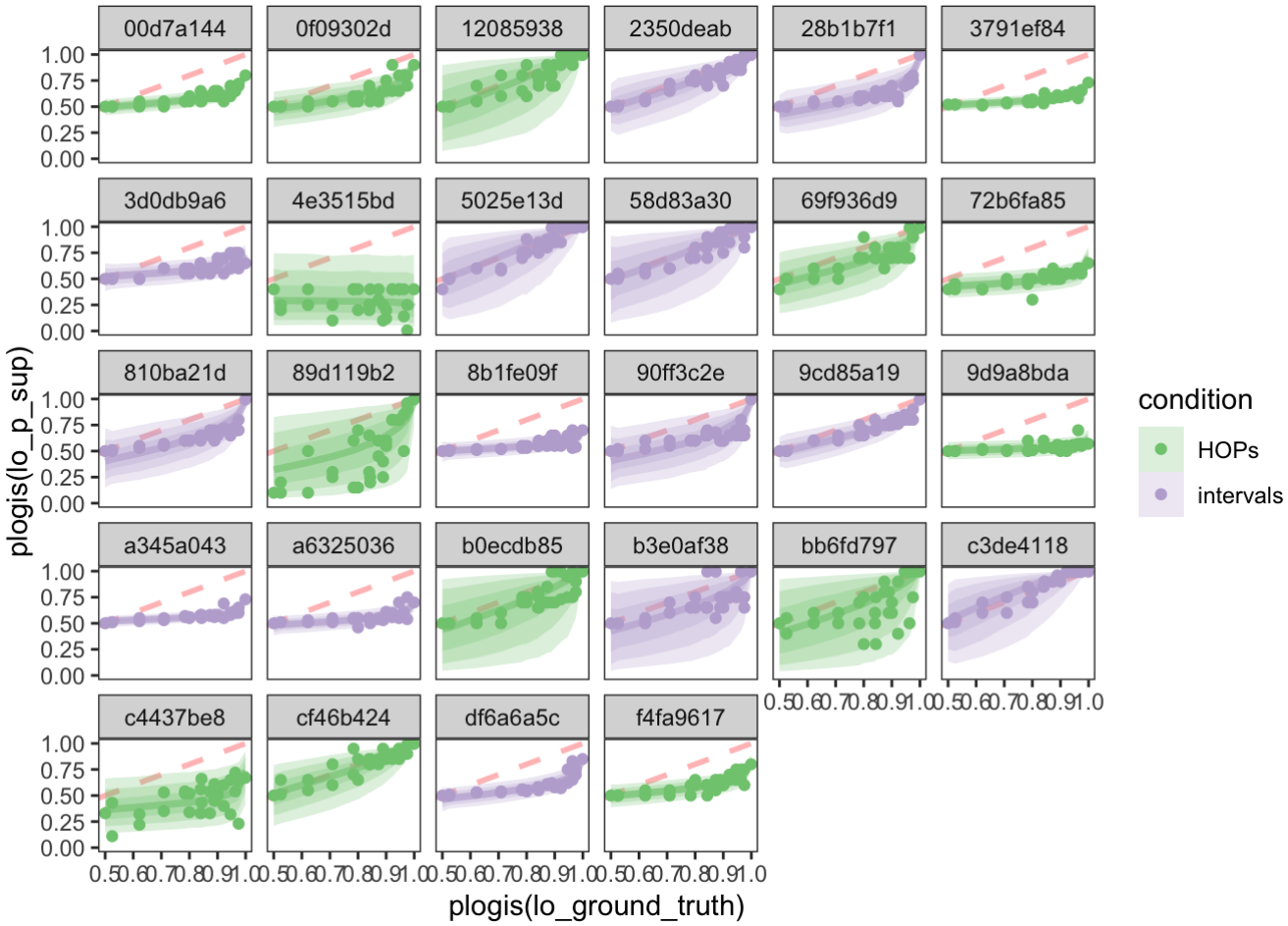
Let's take a look at predictions per worker and visualization condition to get a more granular sense of our model fit.

```
model_df_llo %>%
  group_by(lo_ground_truth, worker_id, means, condition) %>%
  add_predicted_draws(m.wrkr.means.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_llo) +
  scale_fill_brewer(type = "qual", palette = 1) +
  scale_color_brewer(type = "qual", palette = 1) +
  coord_cartesian(xlim = quantile(model_df_llo$lo_ground_truth, c(0, 1)),
                  ylim = quantile(model_df_llo$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_llo %>%
  group_by(lo_ground_truth, worker_id, means, condition) %>%
  add_predicted_draws(m.wrkr.means.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_llo) +
  scale_fill_brewer(type = "qual", palette = 1) +
  scale_color_brewer(type = "qual", palette = 1) +
  coord_cartesian(xlim = quantile(plogis(model_df_llo$lo_ground_truth), c(0, 1)),
                  ylim = quantile(plogis(model_df_llo$lo_p_sup), c(0, 1))) +
  theme_bw() +
  theme(panel.grid = element_blank()) +
  facet_wrap(~ worker_id)
```



Model Comparison

Let's check check which of these four hierarchical models, with and without the presence/absence of the mean, visualization condition, and their interaction as predictors, fits best insofar as the parameters contribute more to predictive validity than they contribute to overfitting. We'll determine this by comparing the models according to the widely applicable information criterion (WAIC). Lower values of WAIC indicate a better fitting model.

```
waic(m.wrkr.llo_p_sup, m.wrkr.means.llo_p_sup, m.wrkr.vis.llo_p_sup, m.wrkr.means.vis.ll
o_p_sup)
```

```
##                                                    WAIC     SE
## m.wrkr.llo_p_sup                                1204.55 102.91
## m.wrkr.means.llo_p_sup                          1153.76  98.64
## m.wrkr.vis.llo_p_sup                            1206.65 103.26
## m.wrkr.means.vis.llo_p_sup                      1134.68  97.53
## m.wrkr.llo_p_sup - m.wrkr.means.llo_p_sup         50.80  18.70
## m.wrkr.llo_p_sup - m.wrkr.vis.llo_p_sup           -2.10   1.69
## m.wrkr.llo_p_sup - m.wrkr.means.vis.llo_p_sup     69.87  21.72
## m.wrkr.means.llo_p_sup - m.wrkr.vis.llo_p_sup    -52.90  18.80
## m.wrkr.means.llo_p_sup - m.wrkr.means.vis.llo_p_sup  19.08  12.92
## m.wrkr.vis.llo_p_sup - m.wrkr.means.vis.llo_p_sup    71.97  22.03
```

It looks like the full model with the interaction between the presence/absence of the mean * visualization condition has the best balance of predictive validity vs overfitting.