Pilot Analysis: Non-Censored Linear Log Odds Model of Probability of Superiority for Gain Frame Trials Only

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 did not constrain the response scale to 50-100. Rather we allowed people to respond on a scale of 0-100. With this approach a censored model may not be necessary.

Since we are thinking of removing the loss frame trials, we will fit our models to data for trials 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_gain_frame = 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,
    company_value = companyValue,
    ground_truth = groundTruth,
    p_contract_new = pContractNew,
    p_contract_old = pContractOld,
    p_superiority = pSup,
    start_time = startTime,
    resp_time = respTime,
    trial_dur = trialDur,
    trial_idx = trialIdx
  ) %>%
  filter(trial_idx != "practice", trial_idx != "mock") # remove practice and mock trials
from responses dataframe, leave in full version

head(responses_df)
```

```
## # A tibble: 6 x 27
##   worker_id batch condition baseline contract_value exchange
##   <chr>     <int> <chr>        <dbl>          <dbl>    <dbl>
## 1 a3ee04d1     13 means_on…      0.5           2.25   0.0480
## 2 a3ee04d1     13 means_on…      0.5           2.25   0.0480
## 3 a3ee04d1     13 means_on…      0.5           2.25   0.0480
## 4 a3ee04d1     13 means_on…      0.5           2.25   0.0480
## 5 a3ee04d1     13 means_on…      0.5           2.25   0.0480
## 6 a3ee04d1     13 means_on…      0.5           2.25   0.0480
## # … with 21 more variables: start_gain_frame <chr>, total_bonus <dbl>,
## #   duration <dbl>, numeracy <int>, gender <chr>, age <chr>,
## #   education <chr>, chart_use <chr>, company_value <dbl>,
## #   ground_truth <dbl>, intervene <int>, outcome <chr>,
## #   p_contract_new <dbl>, p_contract_old <dbl>, p_superiority <int>,
## #   payoff <dbl>, resp_time <dbl>, start_time <dbl>, trial <chr>,
## #   trial_dur <dbl>, trial_idx <chr>
```

We need the data in a format where it is prepared for modeling. We will remove trials in the loss framing condition. Then we censor responses to the range 50.5% 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. 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 %>%
  # remove loss frame trials
  filter(ground_truth > 0.5) %>%
  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)
  )
```

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 narrow prior for the intercept parameter since we are reasonably sure that the inflection point of the model should be at a ground truth of 50% (i.e., 0 in logit units).

```
# 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(0, 0.02), class = Intercept),
                        prior(normal(0, 1), class = sigma)),
              sample_prior = "only",
              iter = 3000, warmup = 500, chains = 2, cores = 2)
```

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

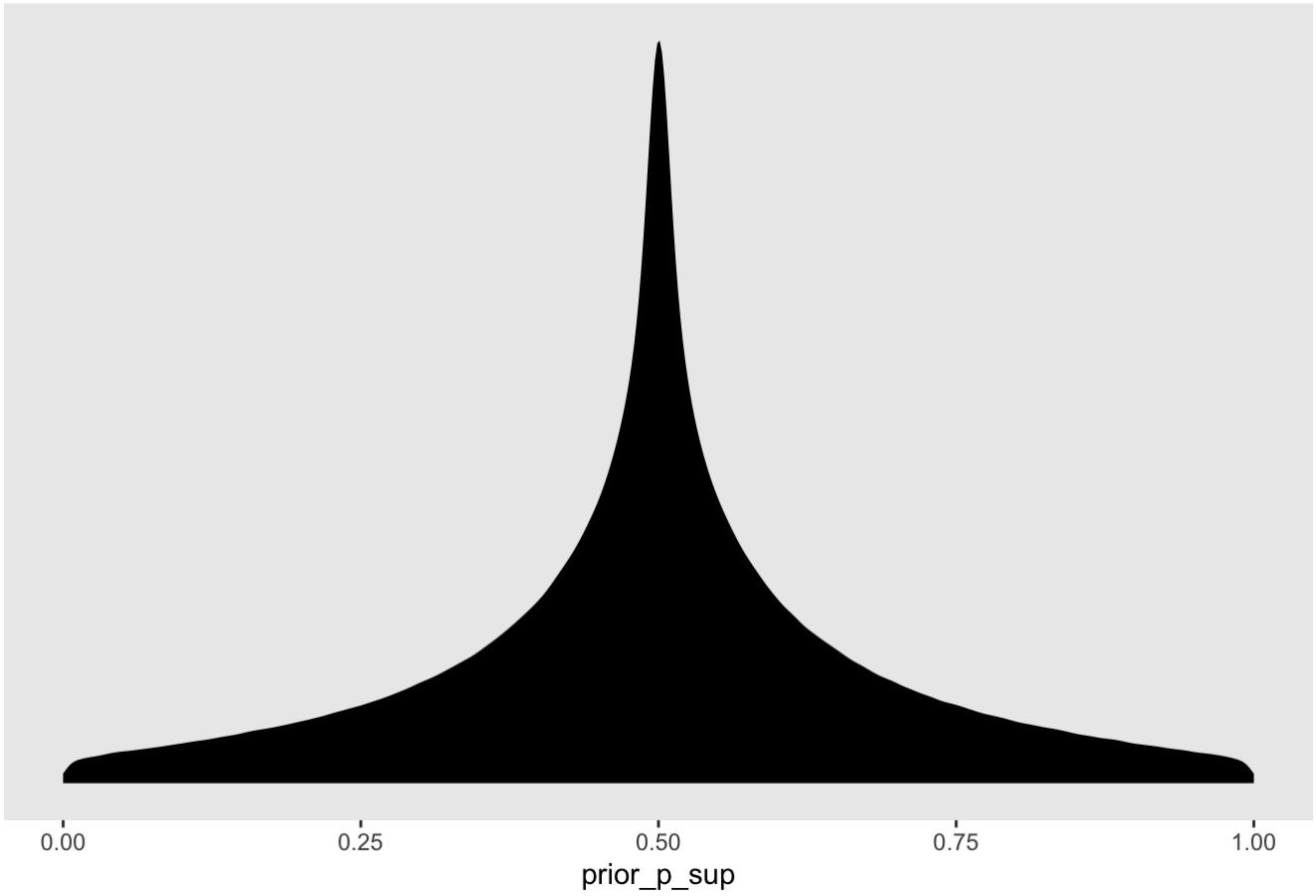
```
## Start sampling
```

```
## Warning: There were 1 divergent transitions after warmup. Increasing adapt_delta abov
e 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())
```



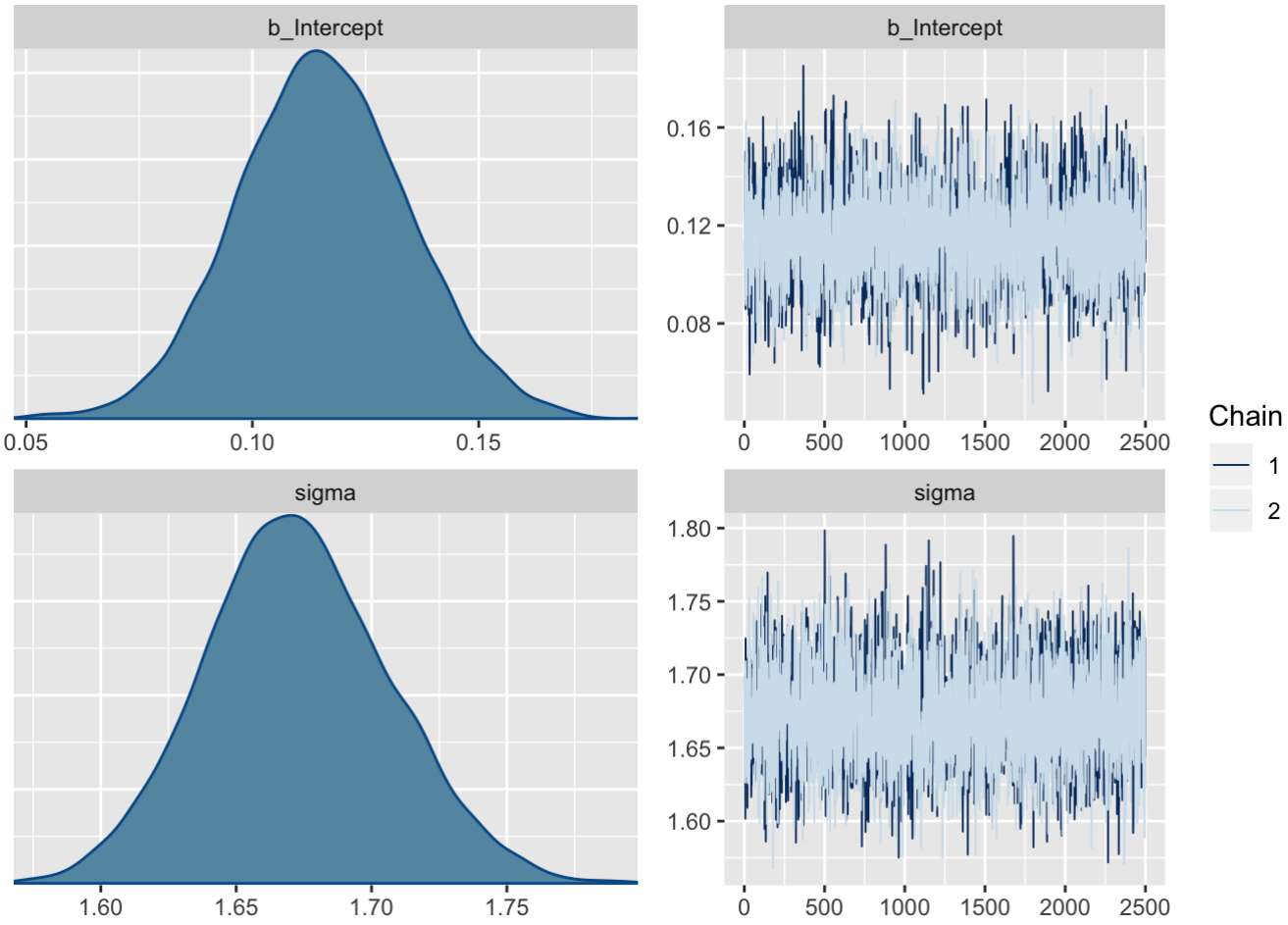
Now, let's fit the model to data.

```
# 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(0, 0.02), class = Intercept),
                        prior(normal(0, 1), class = sigma)),
              iter = 3000, warmup = 500, chains = 2, cores = 2,
              file = "model-fits/lo_mdl_noncens")
```

Check diagnostics:

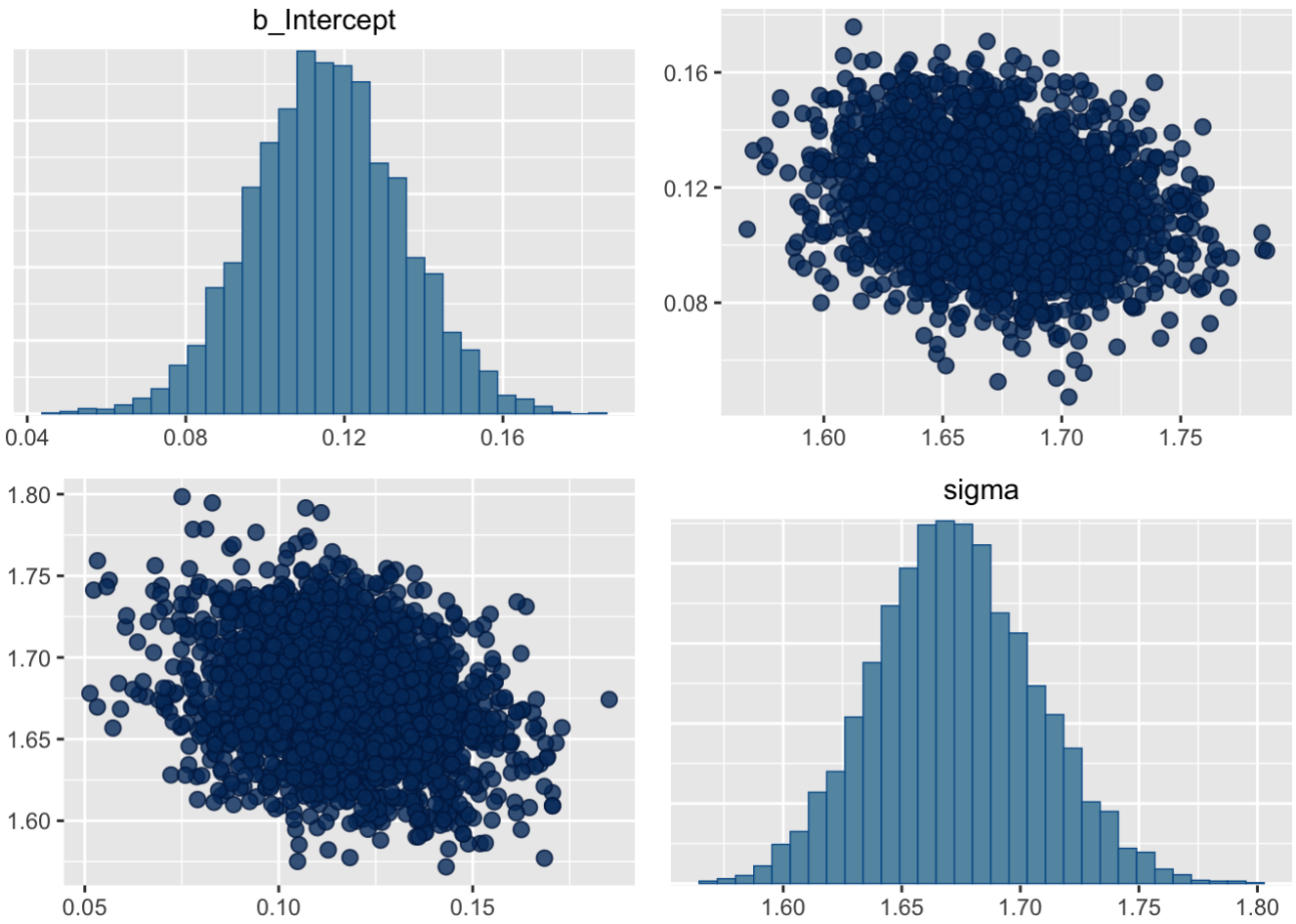
- Trace plots

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



- Pairs plot

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



- Summary

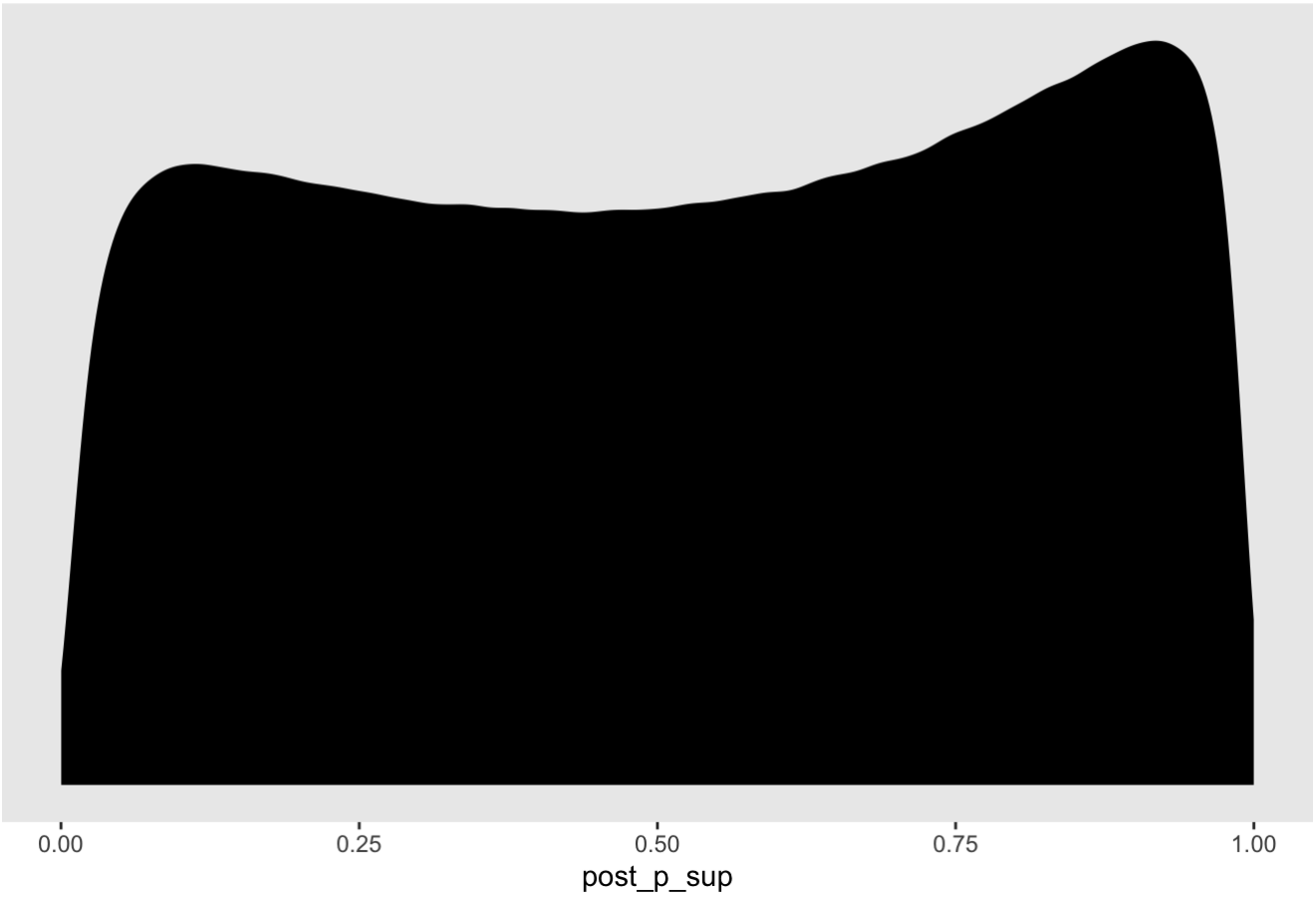
```
# model summary
print(m.lo_p_sup)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: lo_p_sup ~ 1
##    Data: model_df_llo (Number of observations: 1308)
## 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     0.12      0.02     0.08     0.15       3311 1.00
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## sigma     1.67      0.03     1.61     1.74       3309 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())
```

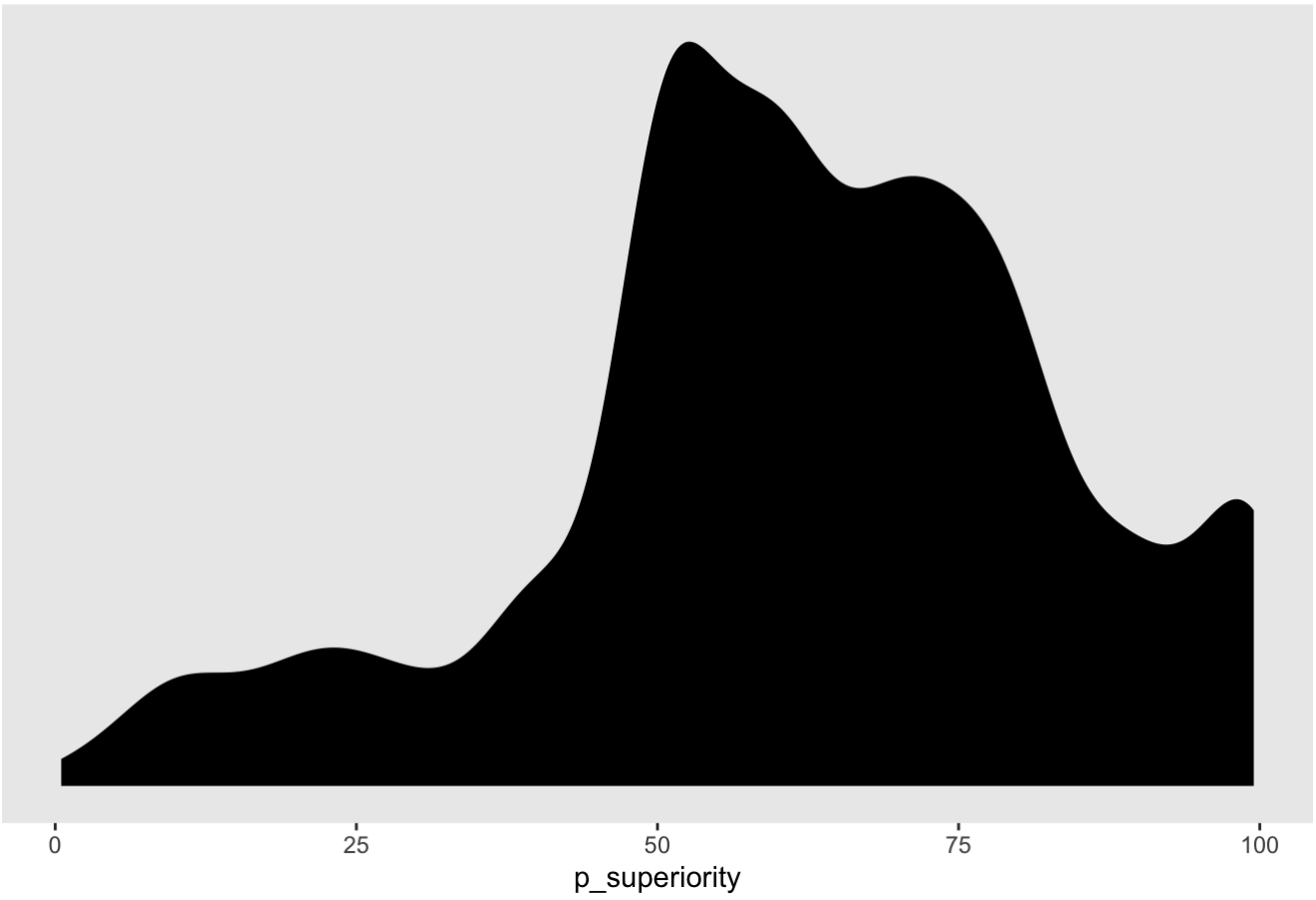
Posterior predictive distribution for probability of superiority



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 ~ 0 + intercept
+ lo_ground_truth)

# simple LLO model
prior.llo_p_sup <- brm(data = model_df_llo, family = "gaussian",
              lo_p_sup ~ 0 + intercept + lo_ground_truth, # non-centered intercept
              prior = c(prior(normal(1, 0.5), class = b),
                        prior(normal(0, 0.02), class = b, coef = 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
```

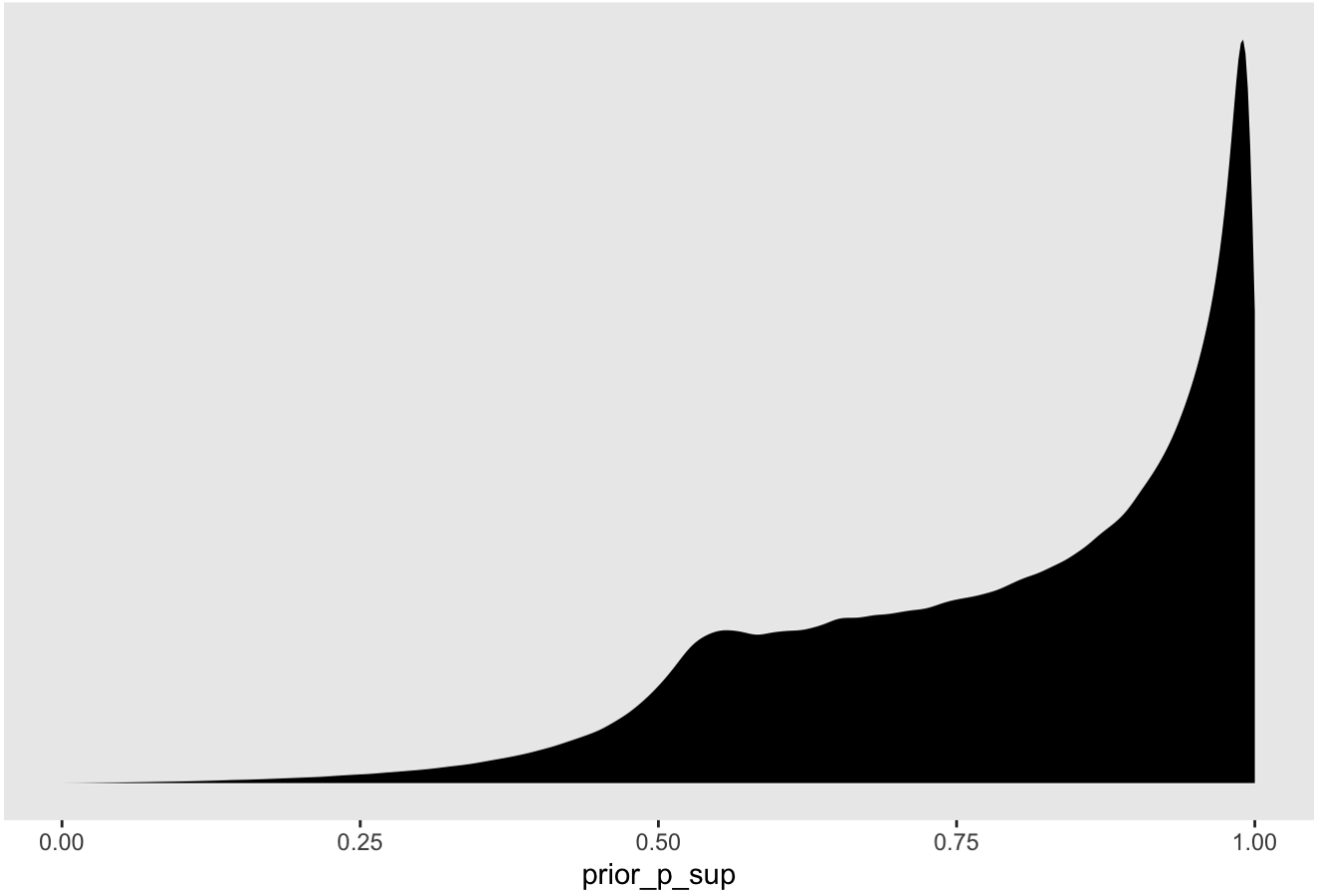
```
## Warning: There were 3 divergent transitions after warmup. Increasing adapt_delta abov
e 0.8 may help. See
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
```

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

Let's look at our prior predictive distribution. For this linear model, we should see a mode at 50% where our intercept is located and a larger mode near 100% due our skewed sampling of the ground truth.

```
# 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())
```

Prior predictive distribution for probability of superiority



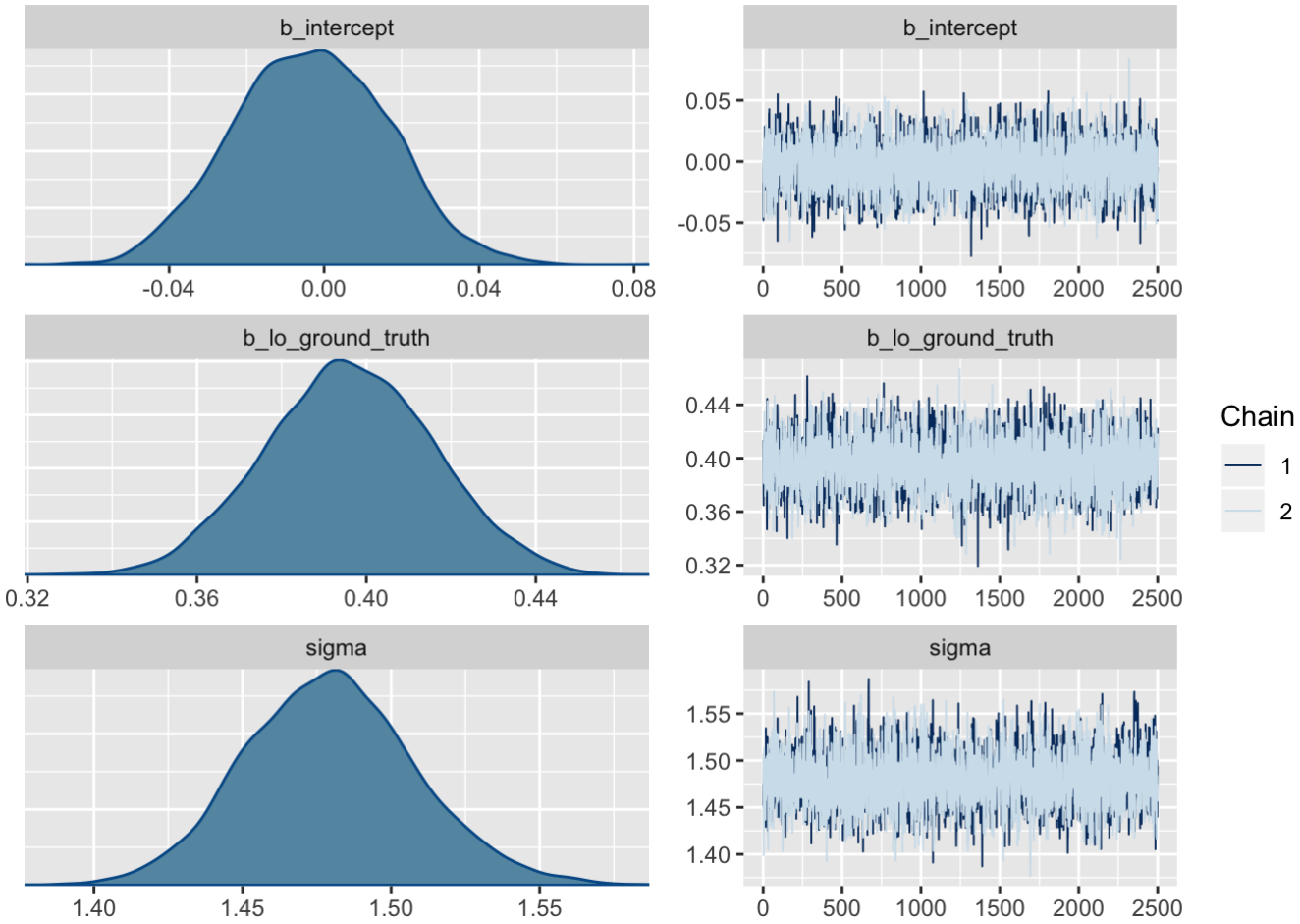
Now let's fit the model to data.

```
# simple LLO model
m.llo_p_sup <- brm(data = model_df_llo, family = "gaussian",
              lo_p_sup ~ 0 + intercept + lo_ground_truth,
              prior = c(prior(normal(1, 0.5), class = b),
                        prior(normal(0, 0.02), class = b, coef = intercept),
                        prior(normal(0, 0.5), class = sigma)),
              sample_prior = TRUE,
              iter = 3000, warmup = 500, chains = 2, cores = 2,
              file = "model-fits/llo_mdl_noncens")
```

Check diagnostics:

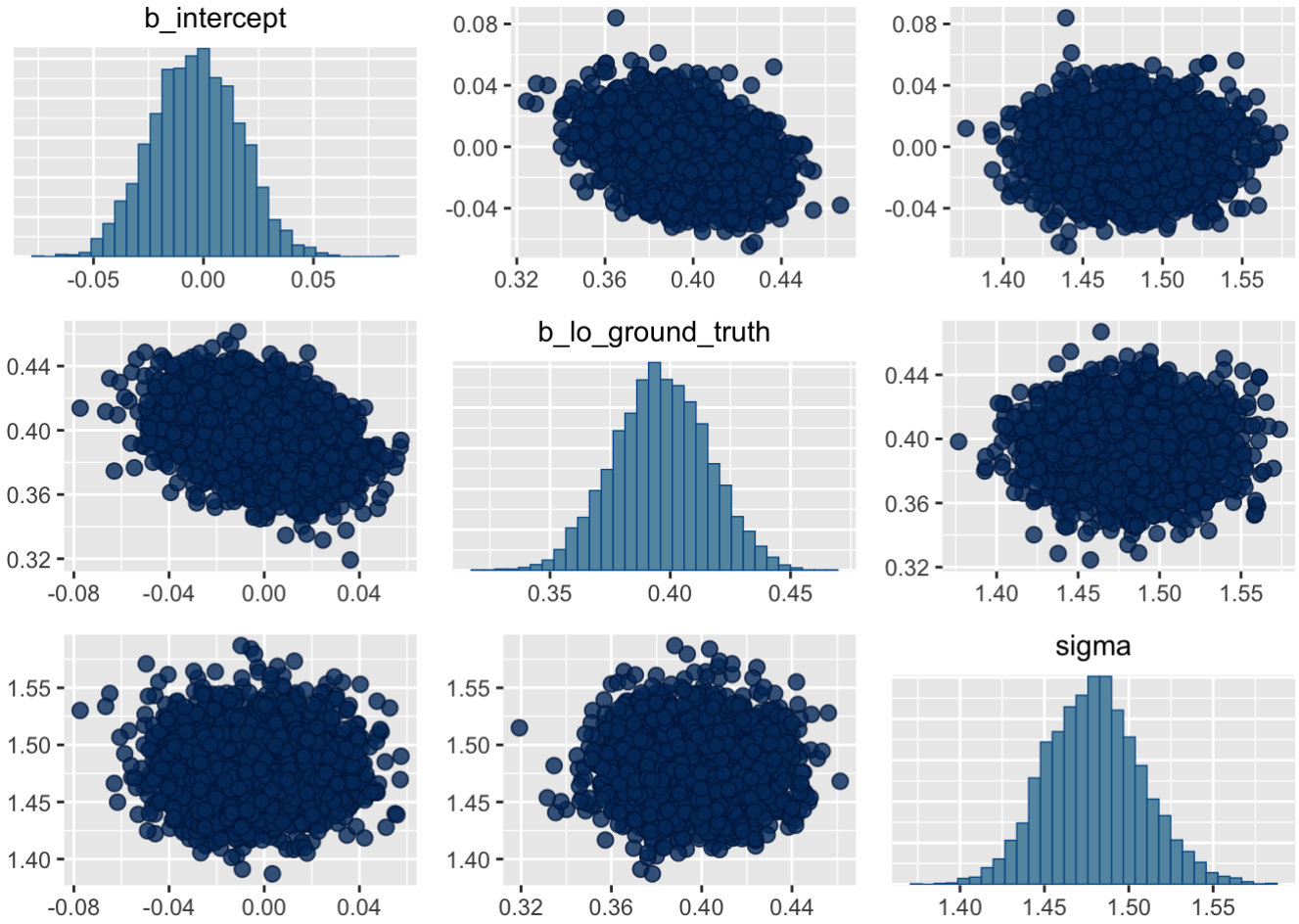
- Trace plots

```
# 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

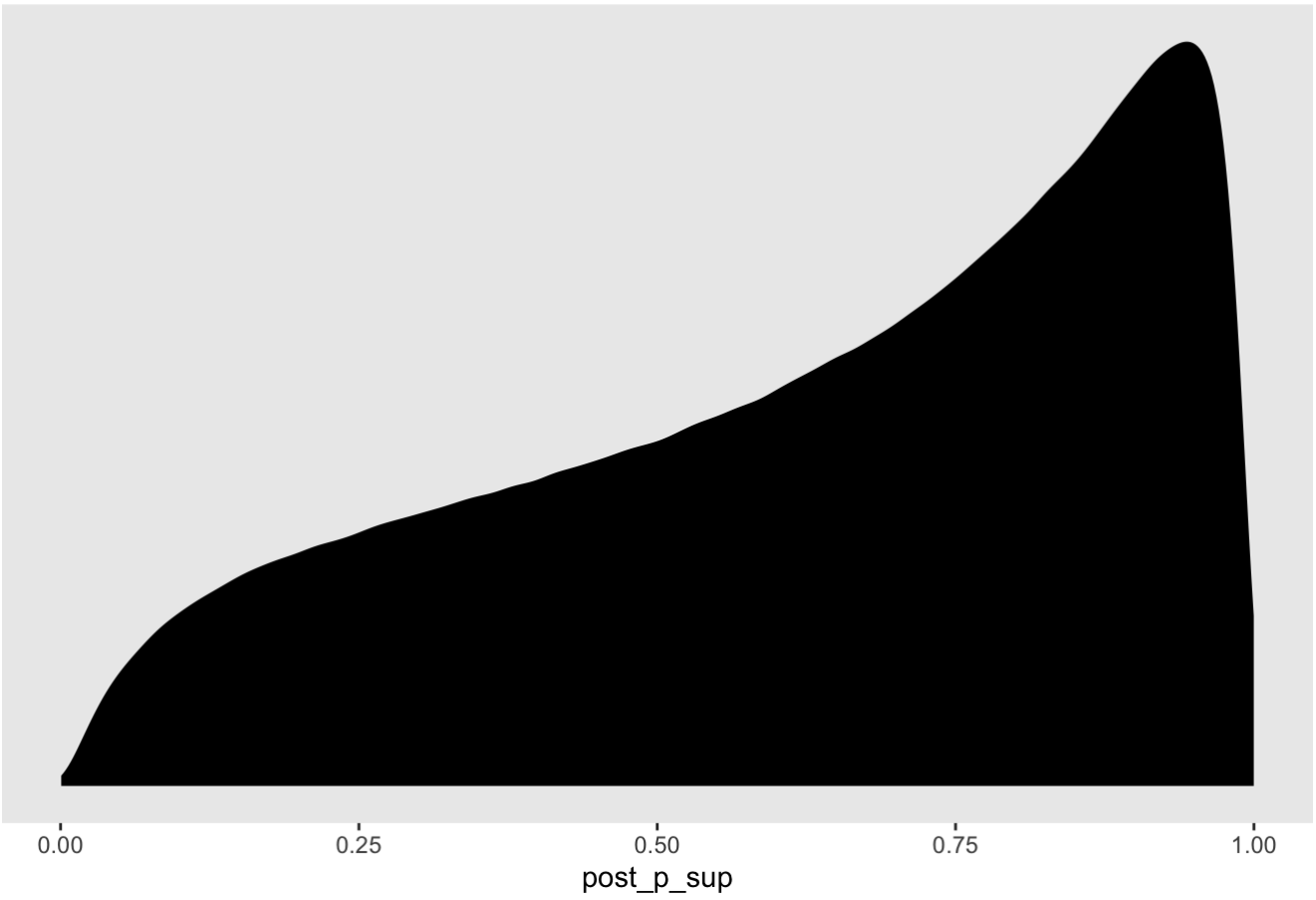
```
# model summary
print(m.llo_p_sup)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: lo_p_sup ~ 0 + intercept + lo_ground_truth
##    Data: model_df_llo (Number of observations: 1308)
## 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          -0.00      0.02    -0.04     0.04       4029 1.00
## lo_ground_truth     0.40      0.02     0.36     0.43       3736 1.00
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## sigma     1.48      0.03     1.43     1.54       3755 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())
```

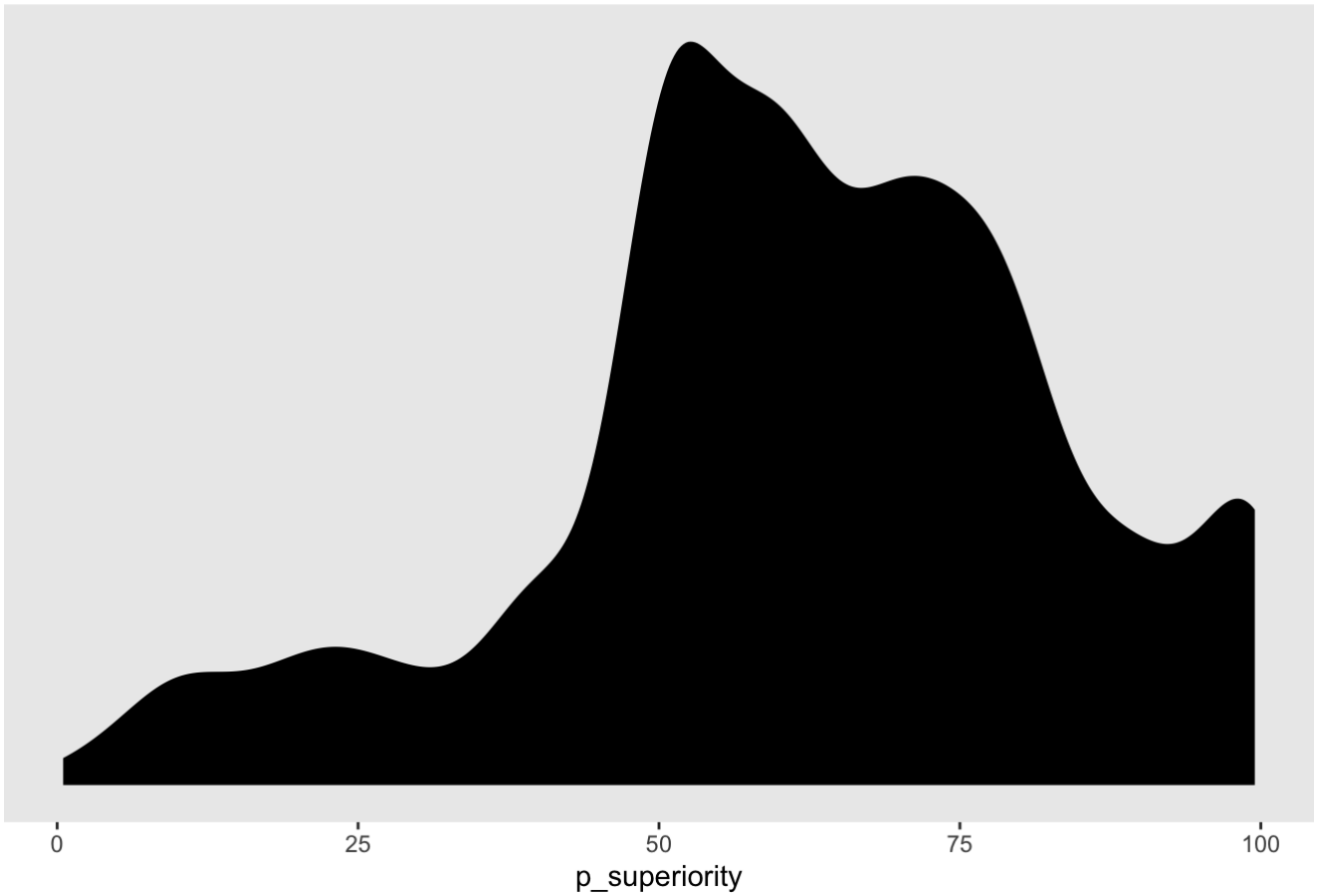
Posterior predictive distribution for probability of superiority



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 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 and intercepts per worker, the correlation between them, and residual variation (i.e., sigma) per worker. We'll constrain the variability of intercepts to a narrow range as before, and we'll also narrow the priors on the variability of slopes and 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 ~  0 + inter
cept + (lo_ground_truth|worker_id) + lo_ground_truth, sigma ~ (1|worker_id)))

# hierarchical LLO model
prior.wrkr.llo_p_sup <- brm(data = model_df_llo, family = "gaussian",
                            formula = bf(lo_p_sup ~  0 + intercept + (lo_ground_truth|wo
rker_id) + lo_ground_truth, # non-centered intercept
                                         sigma ~ (1|worker_id)),
                            prior = c(prior(normal(1, 0.5), class = b),
                                      prior(normal(0, 0.02), class = b, coef = intercep
t),
                                      prior(normal(0, 0.1), class = sd, group = worker_i
d),
                                      prior(normal(0, 0.02), class = sd, group = worker_
id, coef = Intercept),
                                      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)
```

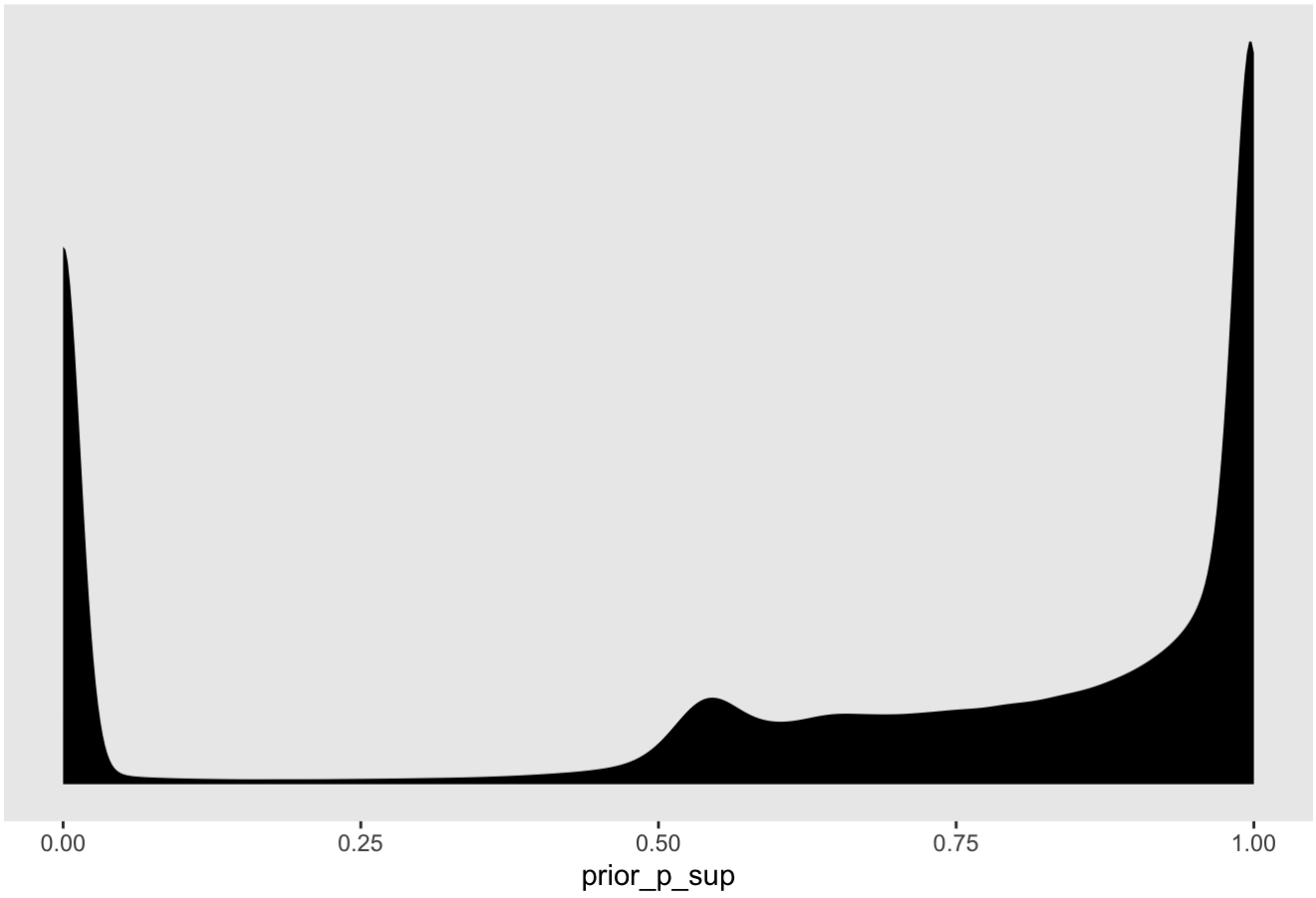
```
## Compiling the C++ model
```

```
## Start sampling
```

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. The large peak near zero is a sign that these priors priors on sigma may still be too wide, despite the fact that we narrowed them substantially from the previous model.

```
# 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())
```

Prior predictive distribution for probability of superiority



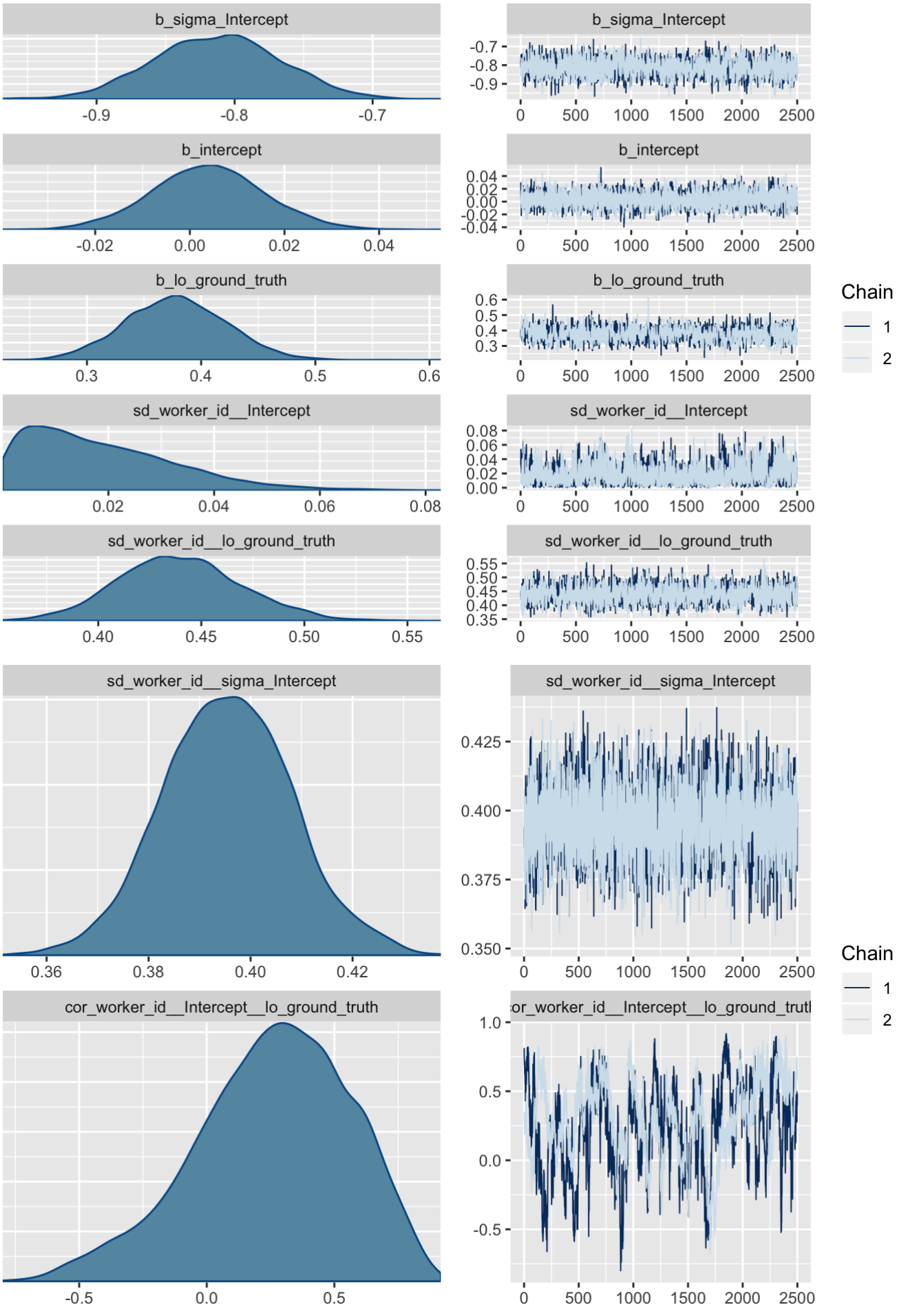
Now, let's fit the model to our data.

```
# hierarchical LLO model
m.wrkr.llo_p_sup <- brm(data = model_df_llo, family = "gaussian",
                        formula = bf(lo_p_sup ~  0 + intercept + (lo_ground_truth|worker
_id) + lo_ground_truth, # non-centered intercept
                                     sigma ~ (1|worker_id)),
                        prior = c(prior(normal(1, 0.5), class = b),
                                  prior(normal(0, 0.02), class = b, coef = intercept),
                                  prior(normal(0, 0.1), class = sd, group = worker_id),
                                  prior(normal(0, 0.02), class = sd, group = worker_id,
 coef = Intercept),
                                  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_noncens-wrkr")
```

Check diagnostics:

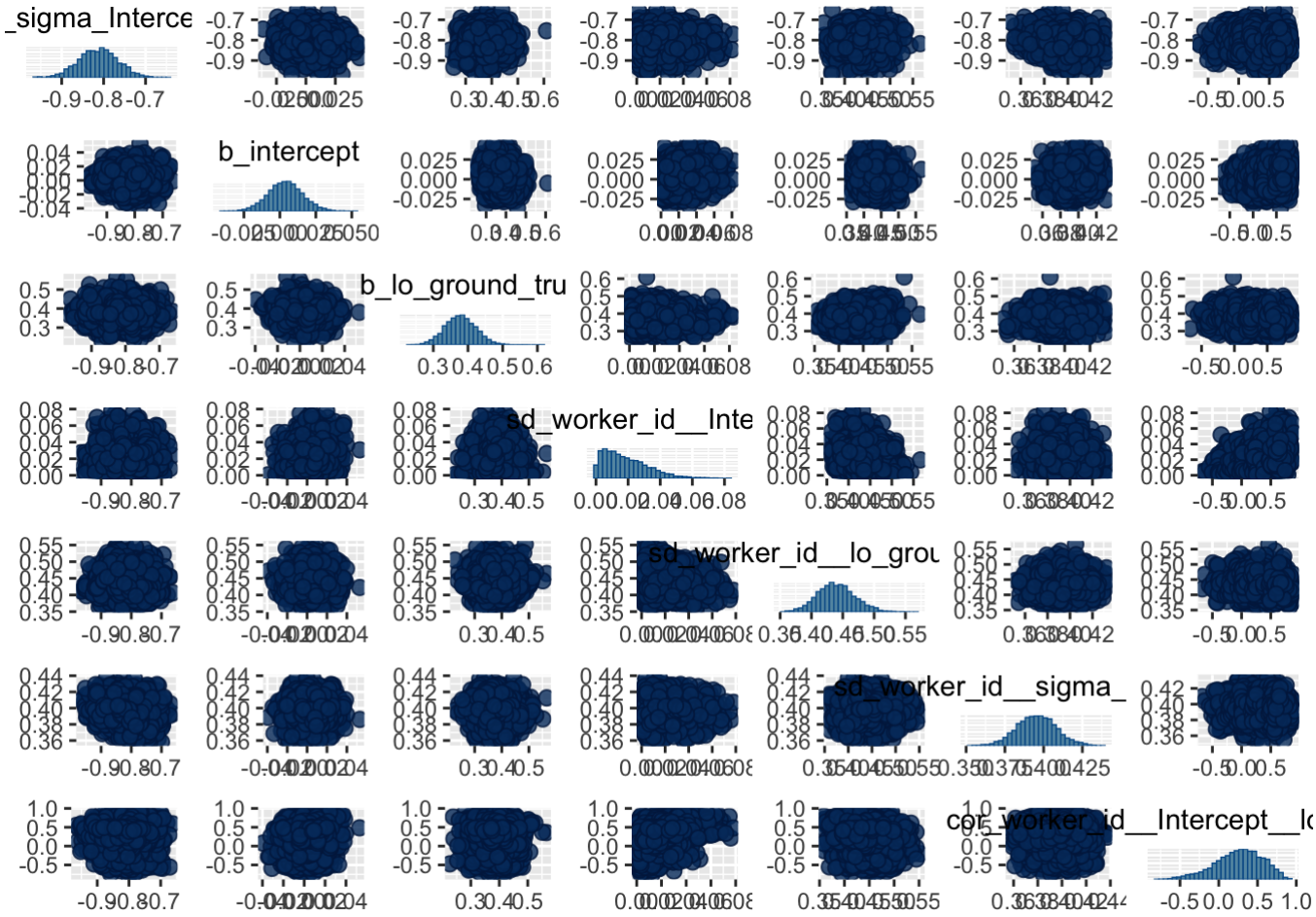
- Trace plots

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


b_sigma_Intercept
b_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
Chain
1
2

- Pairs plot

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



- Summary

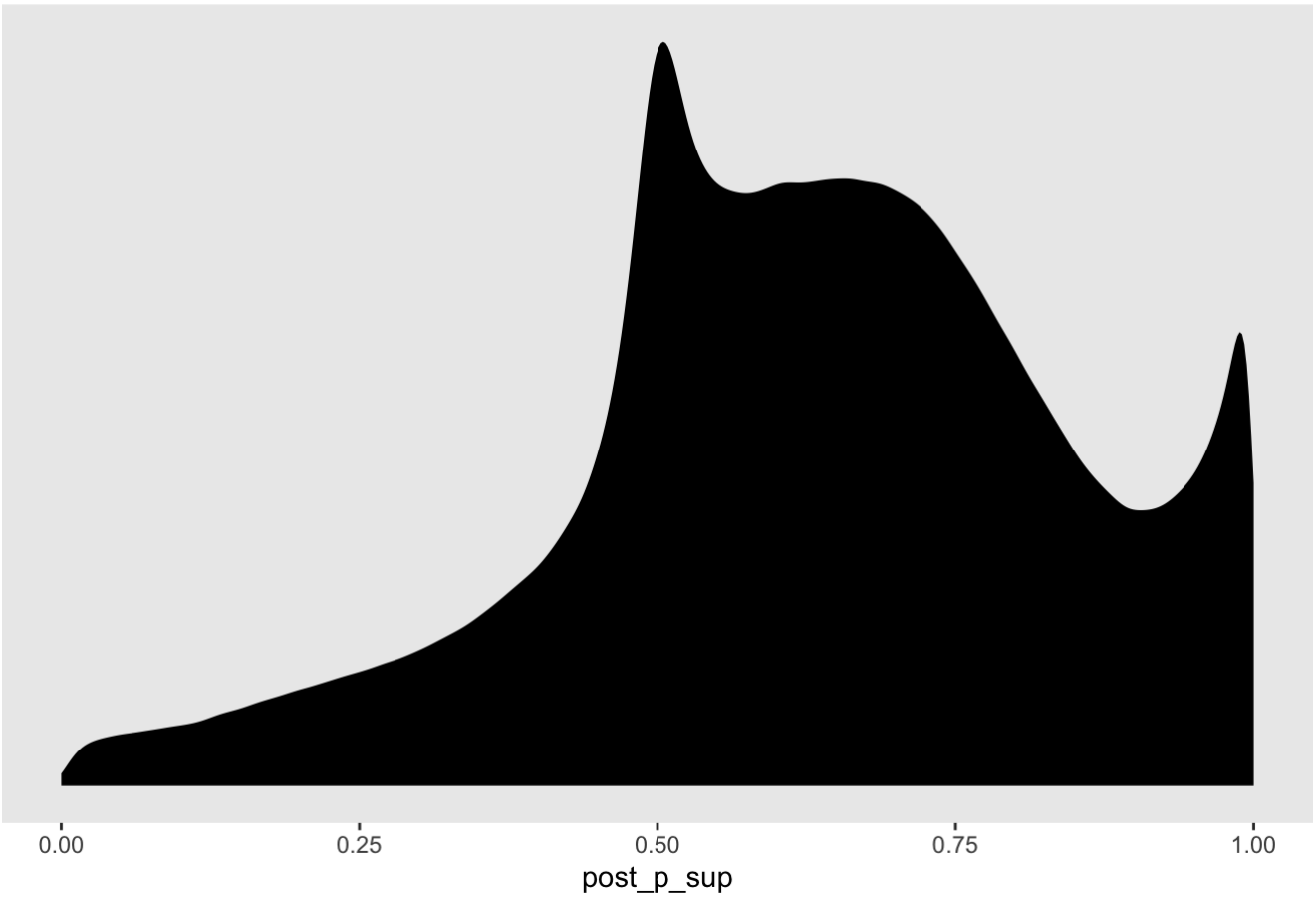
```
# model summary
print(m.wrkr.llo_p_sup)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = log
## Formula: lo_p_sup ~ 0 + intercept + (lo_ground_truth | worker_id) + lo_ground_truth
##          sigma ~ (1 | worker_id)
##    Data: model_df_llo (Number of observations: 1308)
## Samples: 2 chains, each with iter = 3000; warmup = 500; thin = 1;
##          total post-warmup samples = 5000
##
## Group-Level Effects:
## ~worker_id (Number of levels: 109)
##                               Estimate Est.Error l-95% CI u-95% CI
## sd(Intercept)                     0.02      0.01     0.00     0.05
## sd(lo_ground_truth)               0.44      0.03     0.38     0.50
## sd(sigma_Intercept)               0.40      0.01     0.37     0.42
## cor(Intercept,lo_ground_truth)    0.26      0.30    -0.41     0.77
##                               Eff.Sample Rhat
## sd(Intercept)                        331 1.00
## sd(lo_ground_truth)                 1909 1.00
## sd(sigma_Intercept)                 3996 1.00
## cor(Intercept,lo_ground_truth)        71 1.02
##
## Population-Level Effects:
##                 Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## sigma_Intercept    -0.81      0.05    -0.90    -0.72       1896 1.00
## intercept           0.00      0.01    -0.02     0.03       3084 1.00
## lo_ground_truth     0.38      0.04     0.29     0.46       1175 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())
```

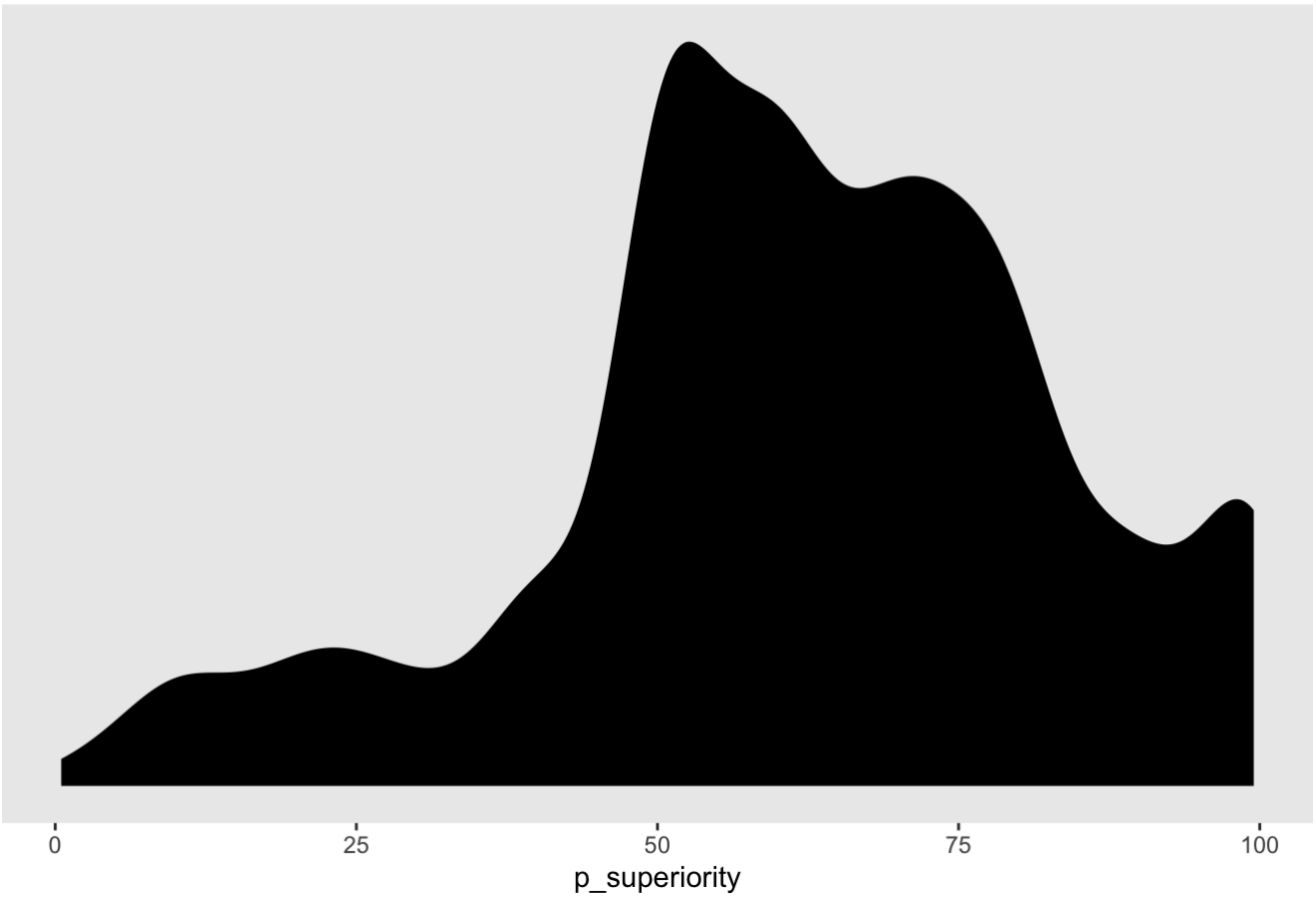
Posterior predictive distribution for probability of superiority



How do these predictions compare to the observed data?

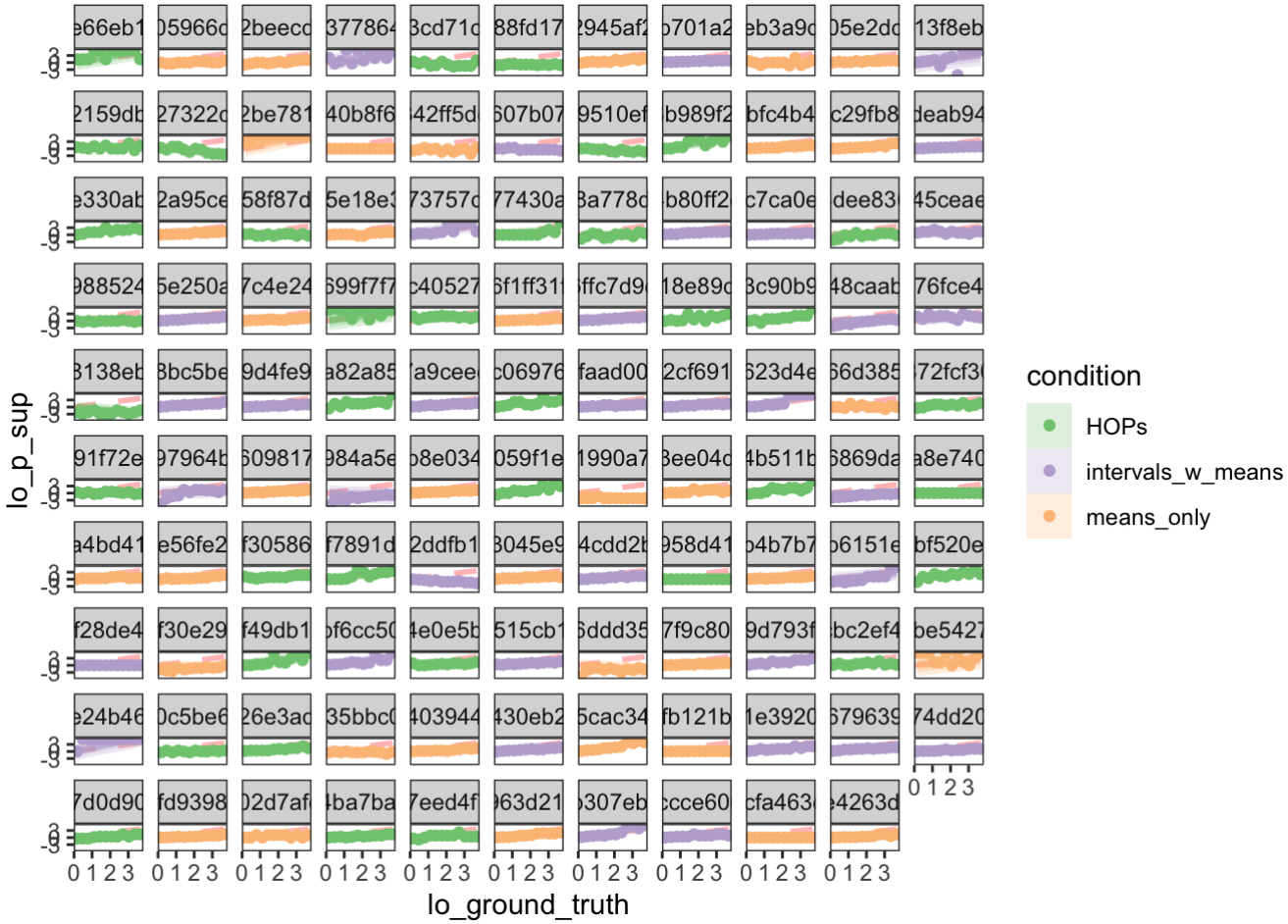
```
# data density
model_df_llo %>%
  ggplot(aes(x = p_superiority)) +
  geom_density(fill = "black", size = 0) +
  scale_y_continuous(NULL, breaks = NULL) +
  labs(subtitle = "Data distribution for probability of superiority") +
  theme(panel.grid = element_blank())
```

Data distribution for probability of superiority



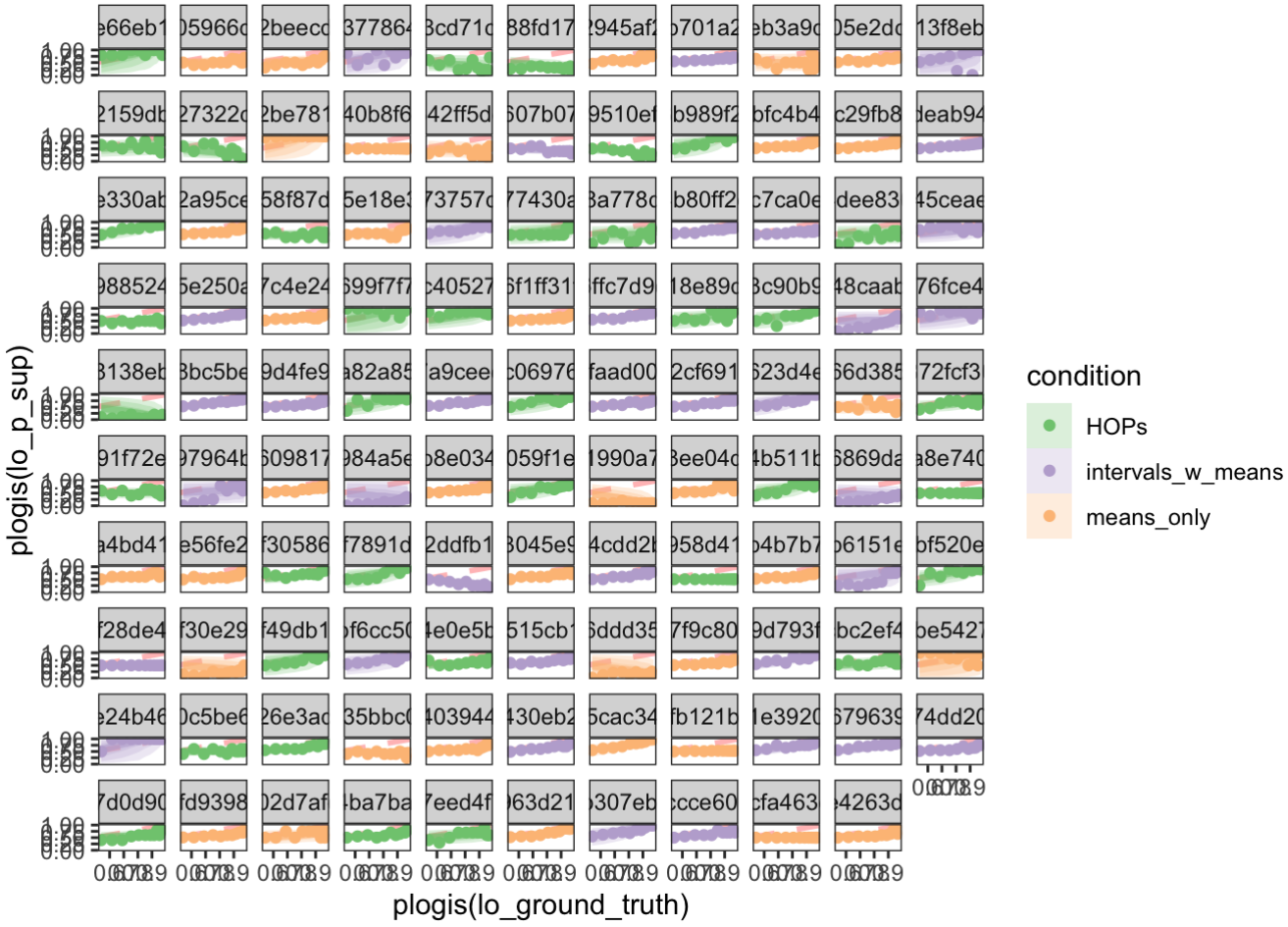
Let's look at posterior predictions per worker to get a more detailed sense of fit quality.

```
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 visualization condition as a predictor.

Add Fixed Effects per Visualization Condition

In the LLO framework, what 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.

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 visualization condition on residual variance. We keep this prior a little wider than the others for sigma since we want to allow effects to vary by visualization condition.

```
# get_prior(data = model_df_llo, family = "gaussian", formula = bf(lo_p_sup | cens(censo
red) ~  0 + intercept + (lo_ground_truth|worker_id) + lo_ground_truth:condition, sigma ~
(1|worker_id) + condition))

# hierarchical LLO model with fixed effects of visualization condition
prior.wrkr.vis.llo_p_sup <- brm(data = model_df_llo, family = "gaussian",
                            formula = bf(lo_p_sup ~  0 + intercept + (lo_ground_truth|wo
rker_id) + lo_ground_truth:condition,
                                         sigma ~ (1|worker_id) + condition),
                            prior = c(prior(normal(1, 0.5), class = b),
                                      prior(normal(0, 0.02), class = b, coef = intercep
t),
                                      prior(normal(0, 0.1), class = sd, group = worker_i
d),
                                      prior(normal(0, 0.02), 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)
```

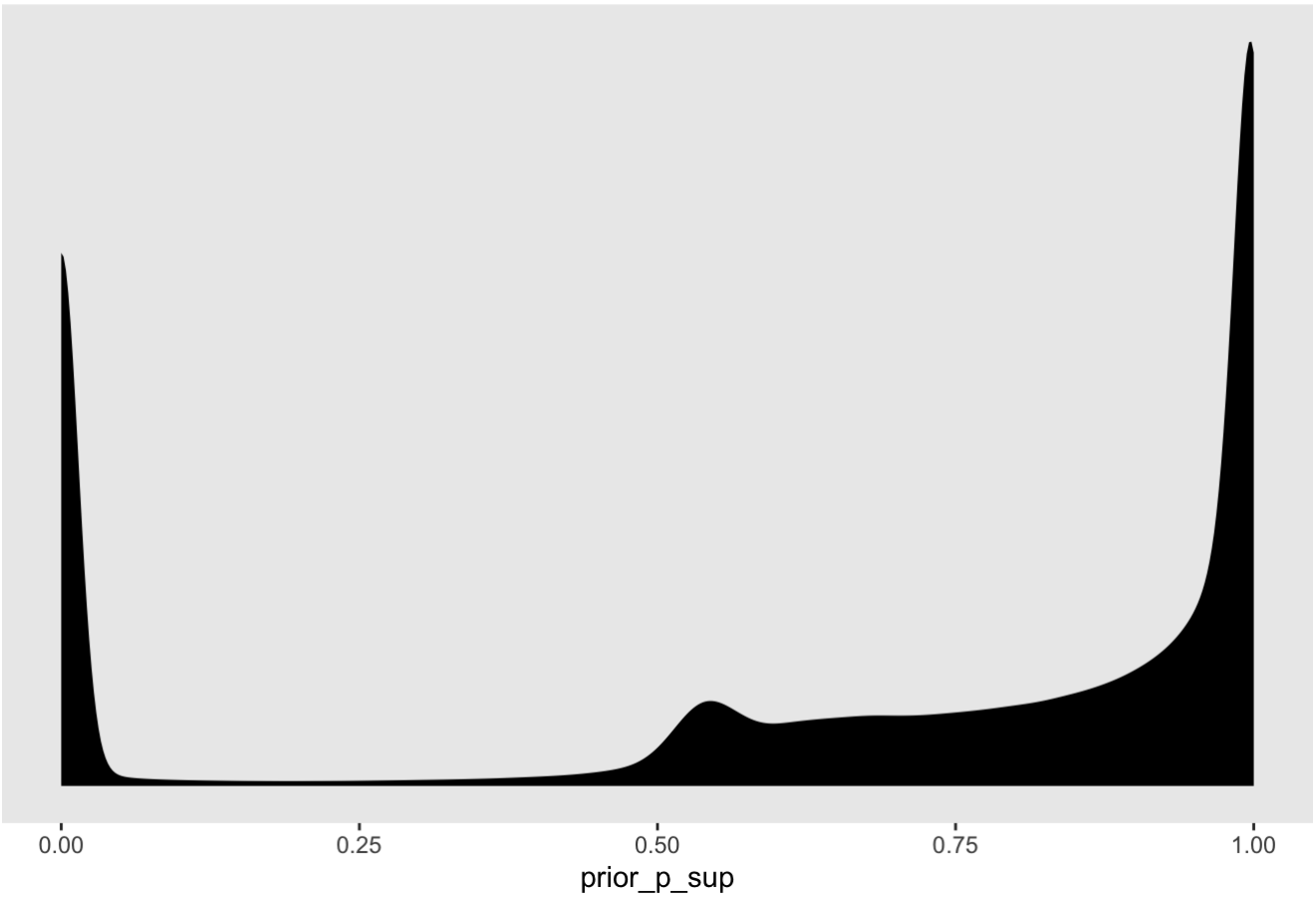
```
## Compiling the C++ model
```

```
## Start sampling
```

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, condition) %>%
  add_predicted_draws(prior.wrkr.vis.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



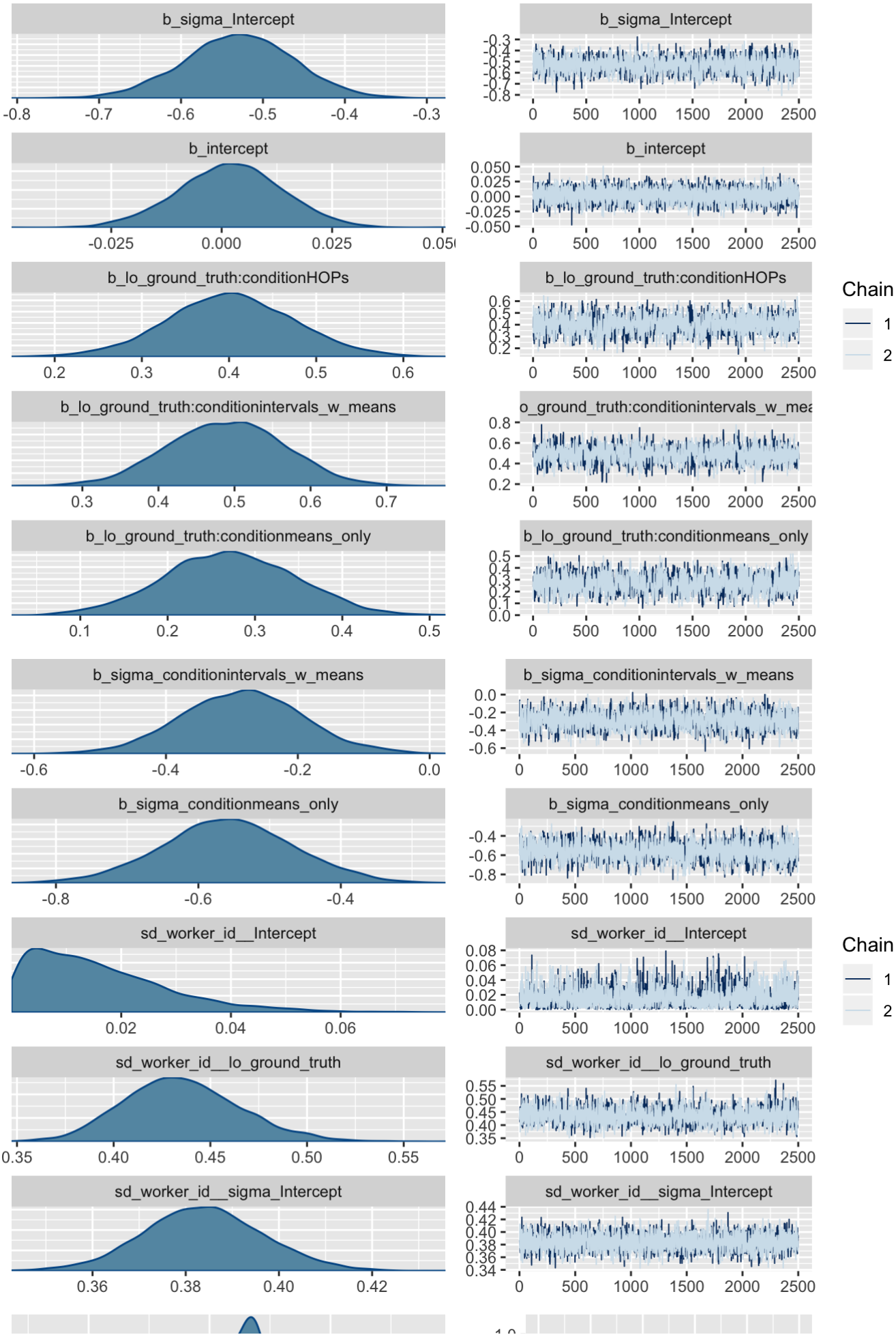
Now, let's fit the modek to our data.

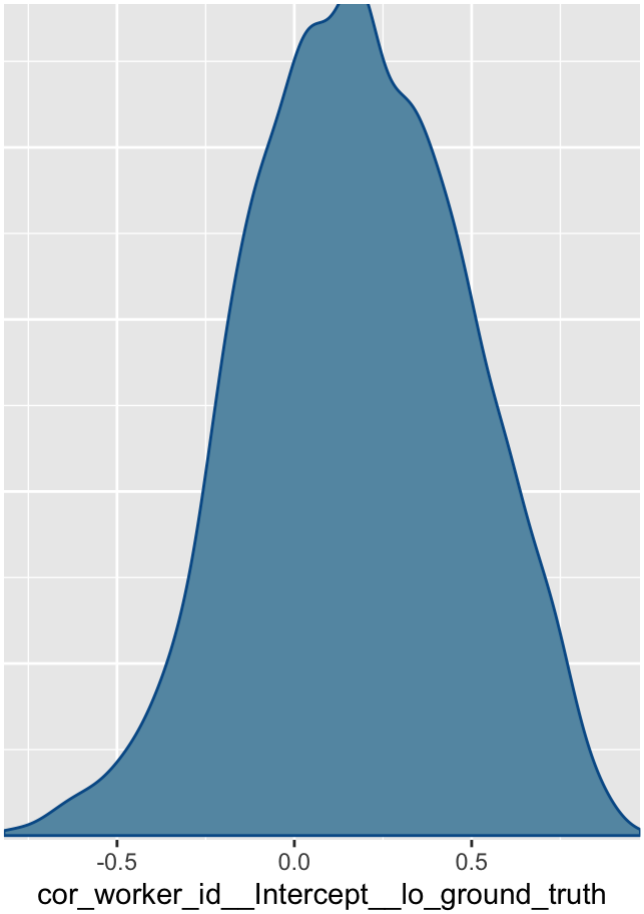
```
# hierarchical LLO model with fixed effects on slope and residual variance per visualiza
tion condition
m.wrkr.vis.llo_p_sup <- brm(data = model_df_llo, family = "gaussian",
                            formula = bf(lo_p_sup ~  0 + intercept + (lo_ground_truth|wo
rker_id) + lo_ground_truth:condition,
                                         sigma ~ (1|worker_id) + condition),
                            prior = c(prior(normal(1, 0.5), class = b),
                                      prior(normal(0, 0.02), class = b, coef = intercep
t),
                                      prior(normal(0, 0.1), class = sd, group = worker_i
d),
                                      prior(normal(0, 0.02), class = sd, group = worker_
id, coef = Intercept),
                                      prior(normal(0, 0.2), class = b, dpar = sigma),
                                      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")
```

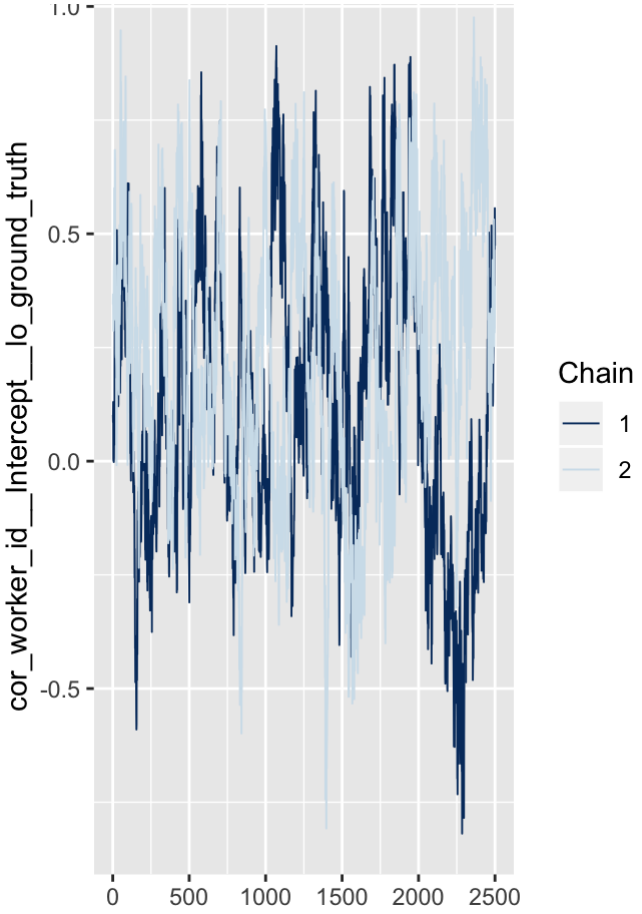
Check diagnostics:

- Trace plots

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

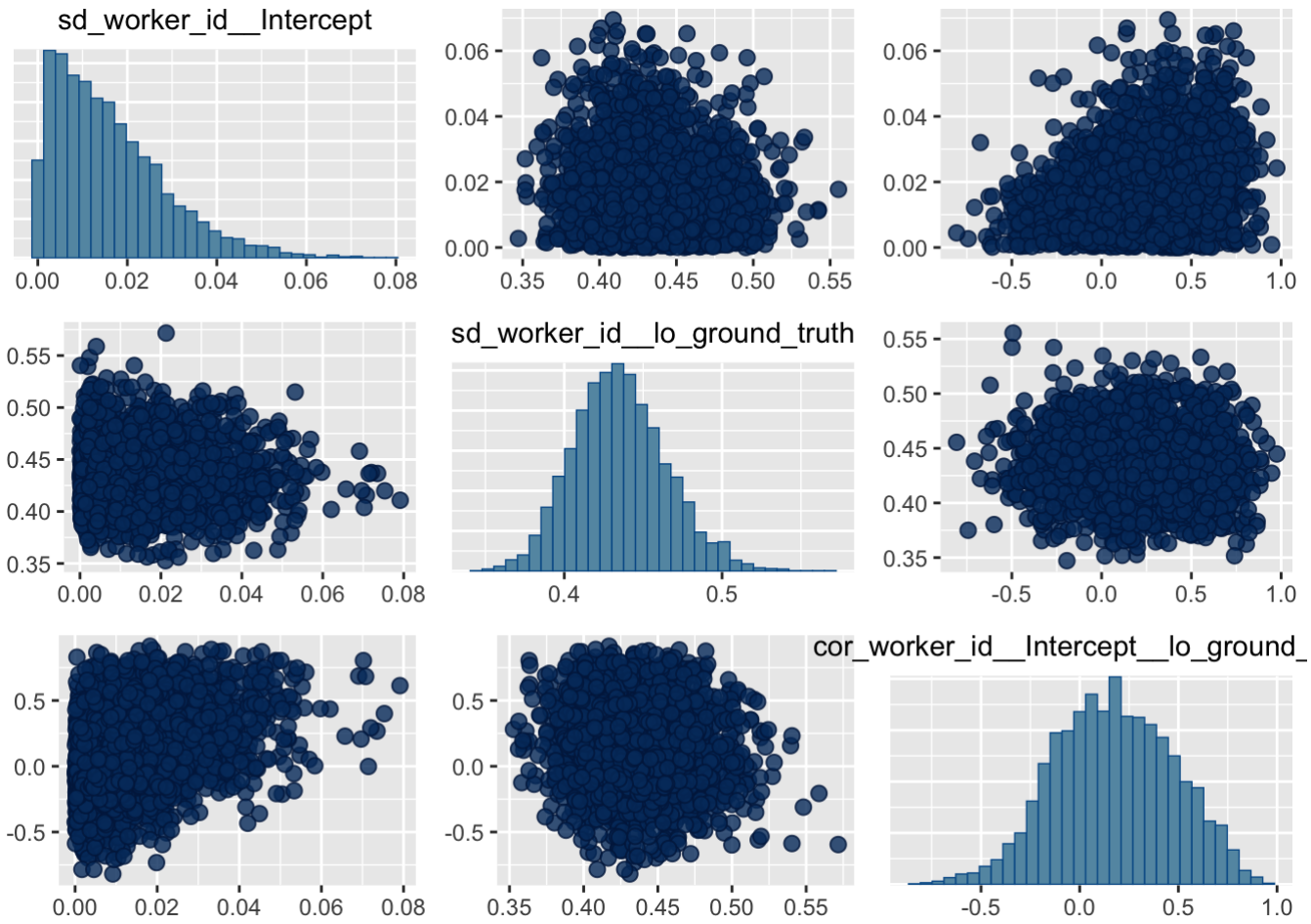

b_sigma_Intercept
b_intercept
b_lo_ground_truth:conditionHOPs
b_lo_ground_truth:conditionintervals_w_means
b_lo_ground_truth:conditionmeans_only
b_sigma_conditionintervals_w_means
b_sigma_conditionmeans_only
sd_worker_id__Intercept
sd_worker_id__lo_ground_truth
sd_worker_id__sigma_Intercept
Chain
1
2



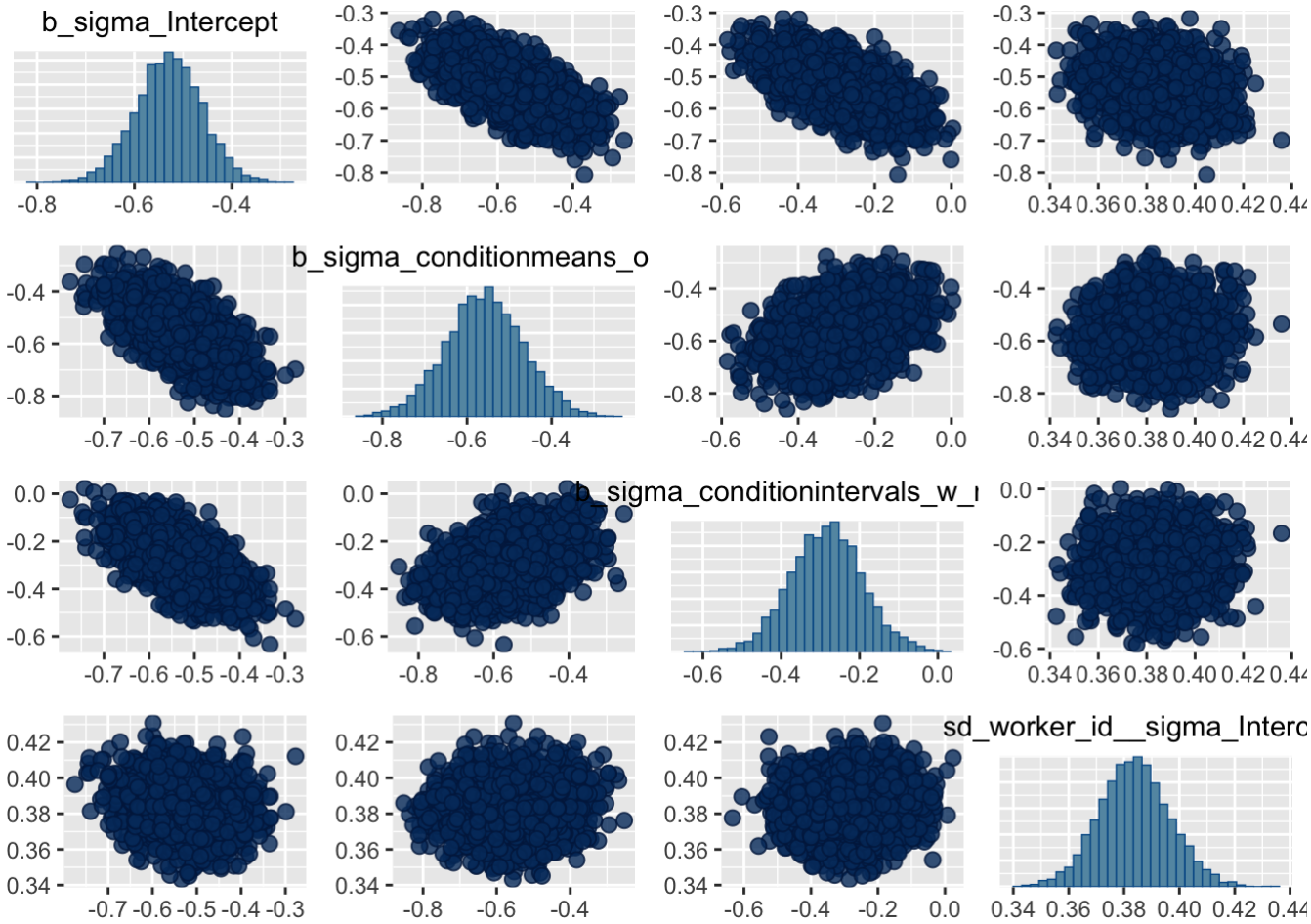


- Pairs plot

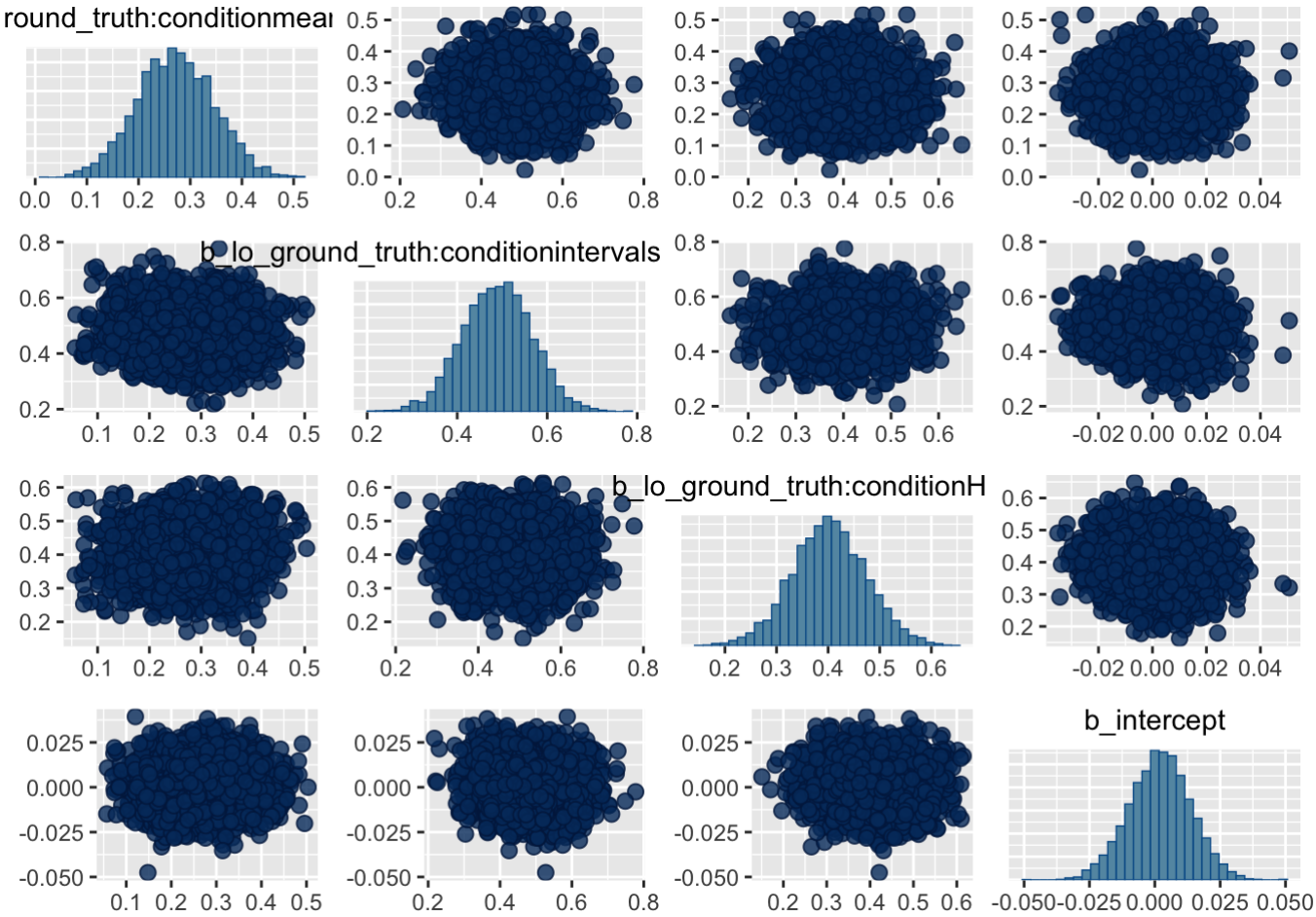
```
# pairs plot
pairs(m.wrkr.vis.llo_p_sup, exact_match = TRUE, pars = c("sd_worker_id__Intercept",
                                                         "sd_worker_id__lo_ground_truth"
,
                                                         "cor_worker_id__Intercept__lo_g
round_truth"))
```




```
# pairs plot
pairs(m.wrkr.vis.llo_p_sup, exact_match = TRUE, pars = c("b_sigma_Intercept",
                                                         "b_sigma_conditionmeans_only",
                                                         "b_sigma_conditionintervals_w_m
eans",
                                                         "sd_worker_id__sigma_Intercept"
))
```




```
# pairs plot
pairs(m.wrkr.vis.llo_p_sup, exact_match = TRUE, pars = c("b_lo_ground_truth:conditionmea
ns_only",
                                                         "b_lo_ground_truth:conditionint
ervals_w_means",
                                                         "b_lo_ground_truth:conditionHOP
s",
                                                         "b_intercept"))
```



- Summary

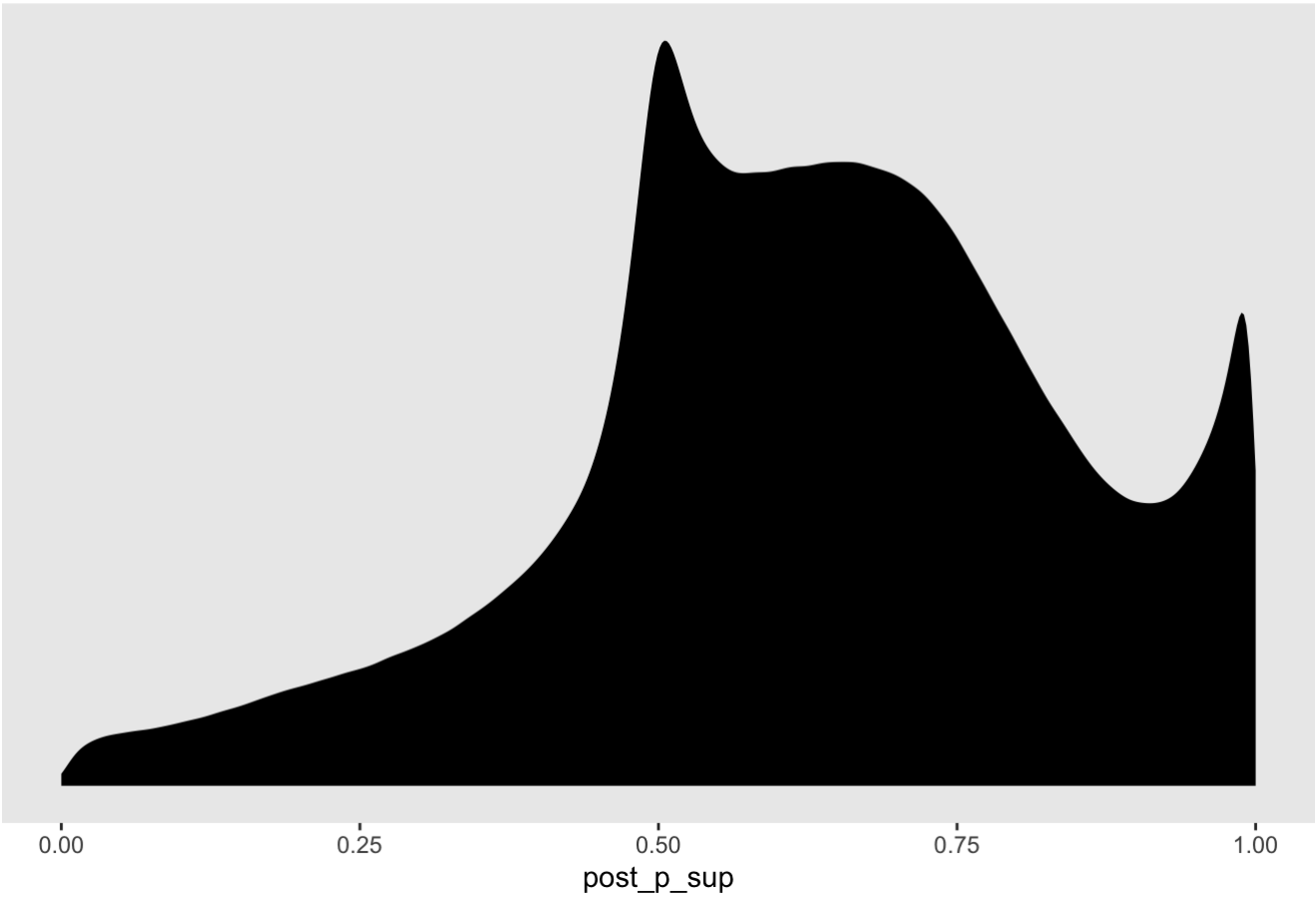
```
# model summary
print(m.wrkr.vis.llo_p_sup)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = log
## Formula: lo_p_sup ~ 0 + intercept + (lo_ground_truth | worker_id) + lo_ground_truth:c
ondition
##          sigma ~ (1 | worker_id) + condition
##    Data: model_df_llo (Number of observations: 1308)
## Samples: 2 chains, each with iter = 3000; warmup = 500; thin = 1;
##          total post-warmup samples = 5000
##
## Group-Level Effects:
## ~worker_id (Number of levels: 109)
##                                Estimate Est.Error l-95% CI u-95% CI
## sd(Intercept)                      0.02      0.01     0.00     0.05
## sd(lo_ground_truth)                0.44      0.03     0.38     0.50
## sd(sigma_Intercept)                0.38      0.01     0.36     0.41
## cor(Intercept,lo_ground_truth)     0.17      0.30    -0.42     0.74
##                                Eff.Sample Rhat
## sd(Intercept)                         526 1.00
## sd(lo_ground_truth)                  2069 1.00
## sd(sigma_Intercept)                  3119 1.00
## cor(Intercept,lo_ground_truth)         42 1.01
##
## Population-Level Effects:
##                                          Estimate Est.Error l-95% CI
## sigma_Intercept                             -0.53      0.07    -0.67
## intercept                                    0.00      0.01    -0.02
## lo_ground_truth:conditionHOPs                0.40      0.07     0.26
## lo_ground_truth:conditionintervals_w_means    0.49      0.08     0.34
## lo_ground_truth:conditionmeans_only           0.27      0.07     0.13
## sigma_conditionintervals_w_means             -0.28      0.09    -0.46
## sigma_conditionmeans_only                    -0.56      0.09    -0.74
##                                          u-95% CI Eff.Sample Rhat
## sigma_Intercept                             -0.40       1751 1.00
## intercept                                    0.02       4653 1.00
## lo_ground_truth:conditionHOPs                0.55       1463 1.00
## lo_ground_truth:conditionintervals_w_means    0.64       1237 1.00
## lo_ground_truth:conditionmeans_only           0.41       1120 1.00
## sigma_conditionintervals_w_means             -0.10       2101 1.00
## sigma_conditionmeans_only                    -0.38       1636 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())
```

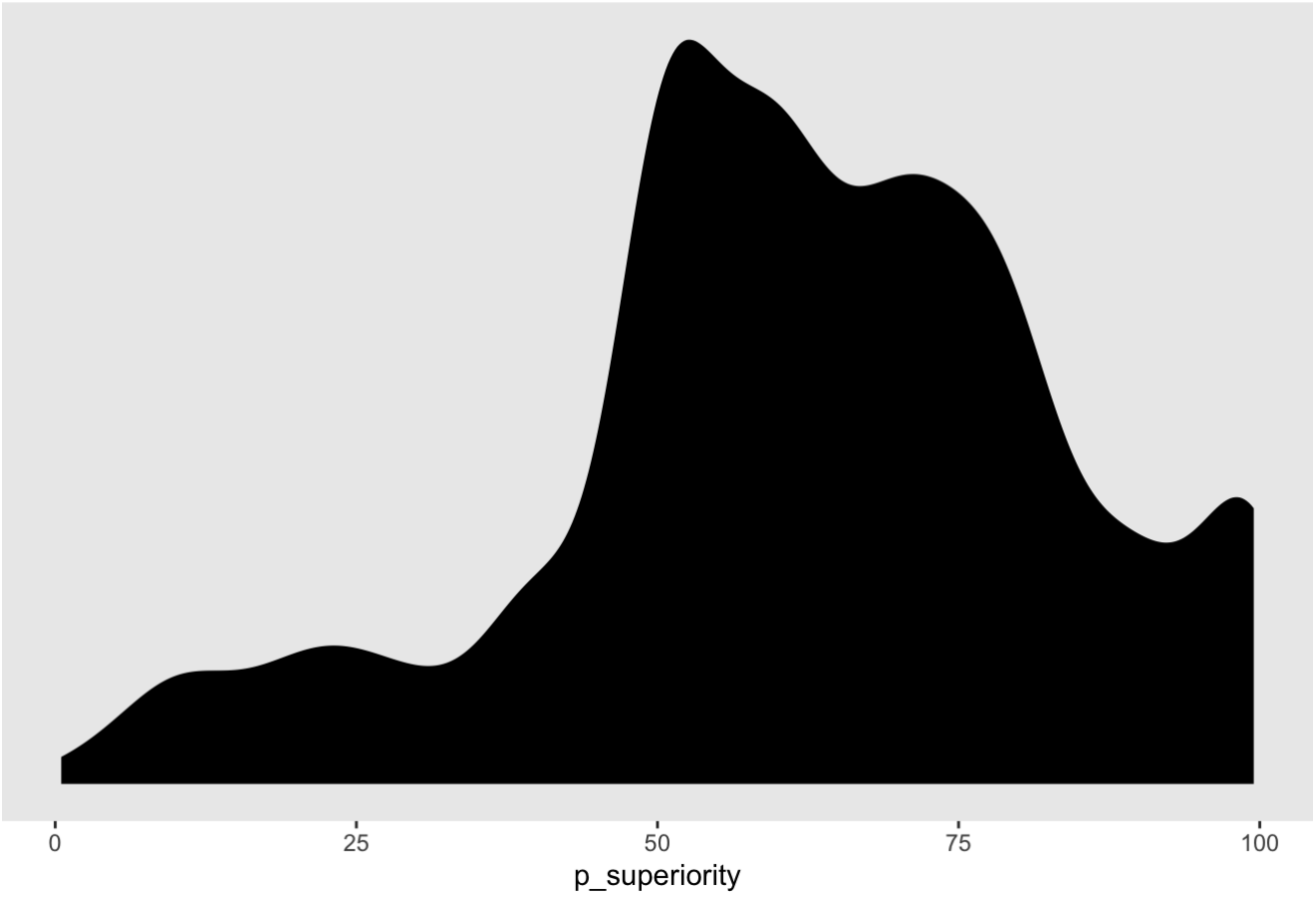
Posterior predictive distribution for probability of superiority



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())
```

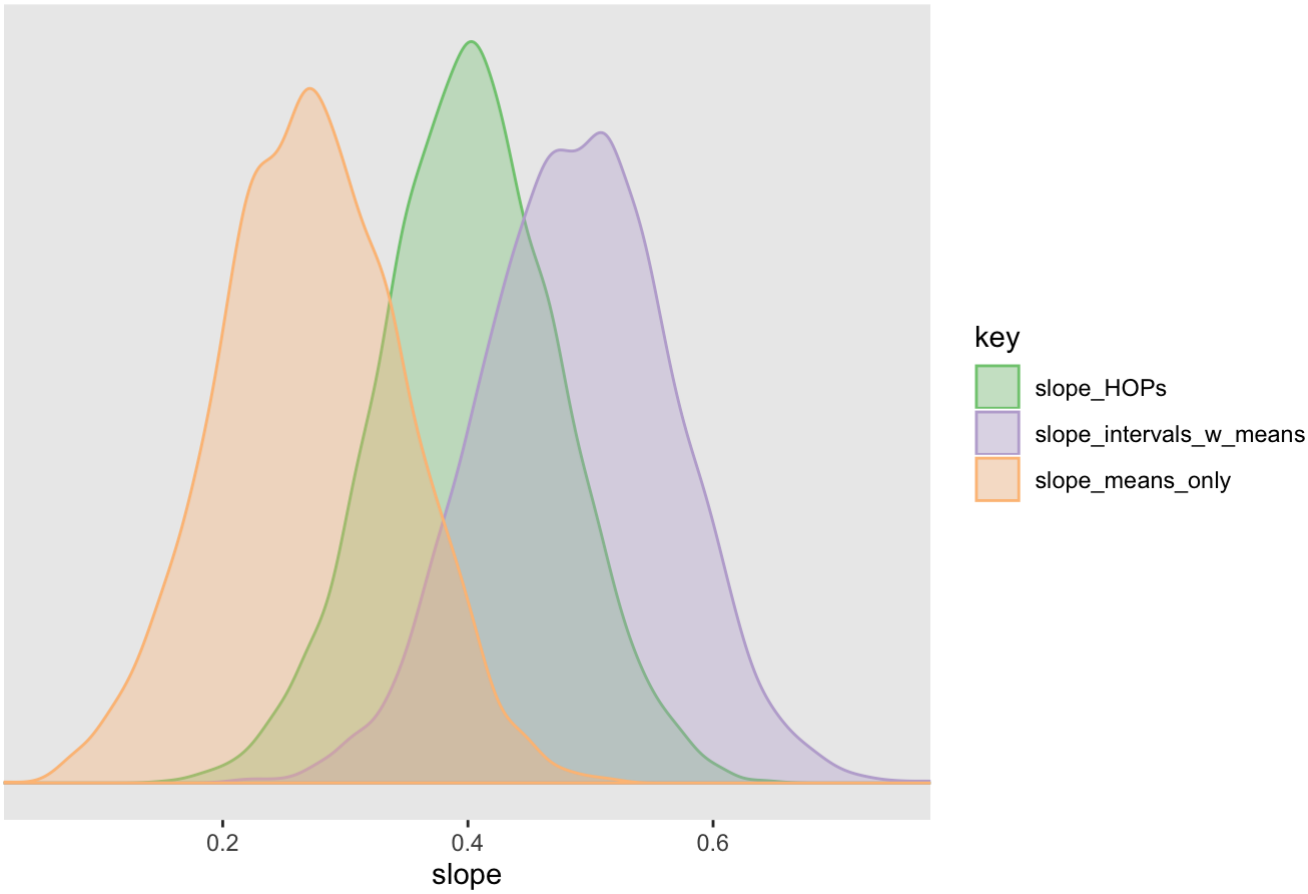
Data distribution for probability of superiority



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:conditionHOPs`,
            slope_intervals_w_means = `b_lo_ground_truth:conditionintervals_w_means`,
            slope_means_only = `b_lo_ground_truth:conditionmeans_only`) %>%
  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())
```

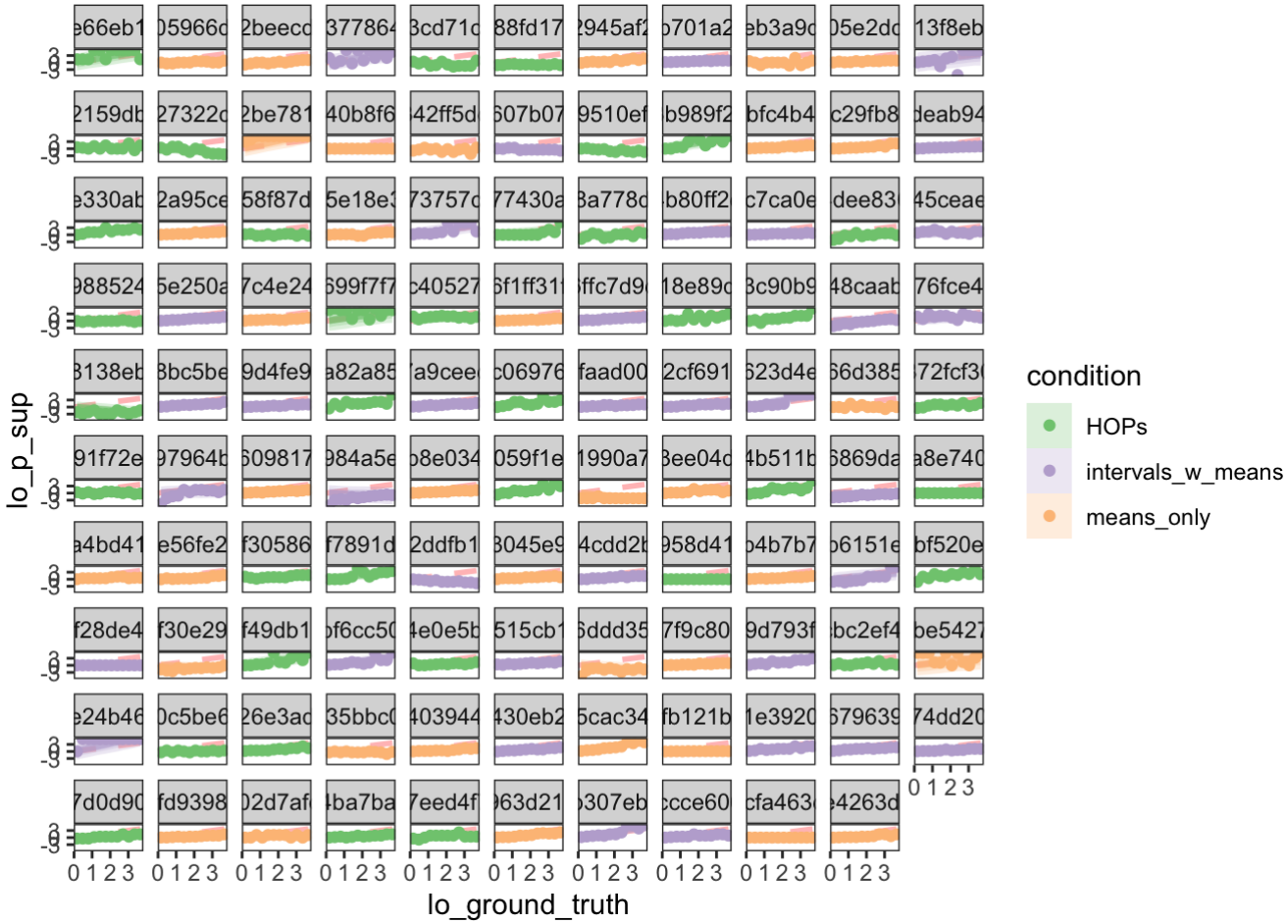
Posterior for slopes by visualization condition



Recall that a slope of 1 reflects zero bias. This suggests that users are biased toward responses of 50% in all conditions.

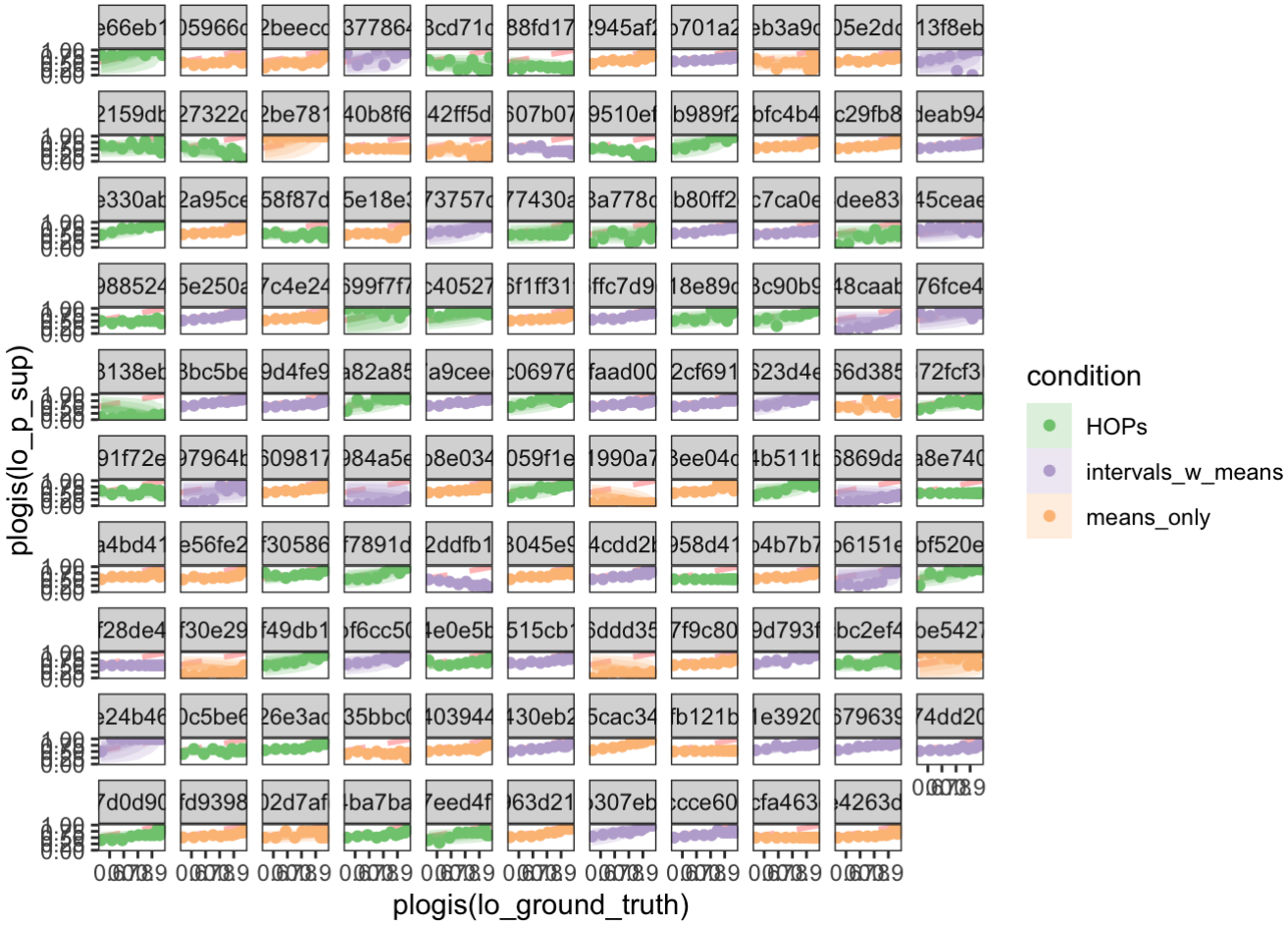
Let's take a look at predictions per worker and visualization condition to get a more granular sense of our model fit.

```
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)
```



This seems much better than the censored model.