Pilot Analysis: Framing Effects on Probability of Superiority Judgments

In this document, we explore the impact of gain vs loss framing on some of our best fitting models of probability. The first model is a linear log odds (LLO) model with predictors for visualization condition and worker. The second model is a mixture of the LLO process with a process where users make a random constant response. The second model fits better because it accounts for an inflated number of responses near 50% (the middle of the probability scale).

We do two things here:

1. We reproduce these models using only data from the first block of trials that each worker completed. This means that we only have data from half the domain of possible responses from each participant. We want to know whether the model fit is robust to this change in sampling. If so, we can make gain vs loss framing a between subjects manipulation in the next pilot in order to reduce the complexity of the task.
2. We check whether slopes in the LLO model depend on gain vs loss framing. This essentially answers the question of whether it is reasonable to assume that patterns of bias in probability of superiority judgments are symmetrical in the gain vs loss framing version of the task. We check this by comparing versions of the two models with and without predictors for problem framing.

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
  mutate( # mutate to jitter probability of superiority away from boundaries
    p_superiority = ifelse(p_superiority == 0, 0.25, p_superiority),           # avo
id responses equal to zero
    p_superiority = ifelse(p_superiority == 100, 99.75, p_superiority)        # avoi
d responses equal to one-hundred
  )

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 <dbl>,
## #   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. This means converting both probability of superiority judgments and the ground truth to a logit scale. It also means that we want baseline as a factor rather than a numeric value. Since we are looking at framing effects, we'll add gain/loss frame as another factor.

Since the response data seem so noisy we want to reduce the complexity of the task by making gain/loss framing a between subjects manipulation. Here we test the viability of this analysis by fitting models to only the first set of trials for each participant.

```
# create data frame for model
model_df_llo <- responses_df %>%
  mutate( # apply logit function to p_sup judgments and ground truth
    lo_p_sup = qlogis(p_superiority / 100),
    lo_ground_truth = qlogis(ground_truth),
    baseline = as.factor(baseline),
    frame = as.factor(if_else(ground_truth > 0.5, "gain", "loss"))
  ) %>%
  # filter to first block of trials only
  filter((start_gain_frame == "True" & ground_truth > 0.5) | start_gain_frame == "Fal
se" & ground_truth < 0.5)
```

Linear Log Odds (LLO) Model with and without Framing Effects

LLO Model with Fixed Effect of Visualization and Random Effect of Worker

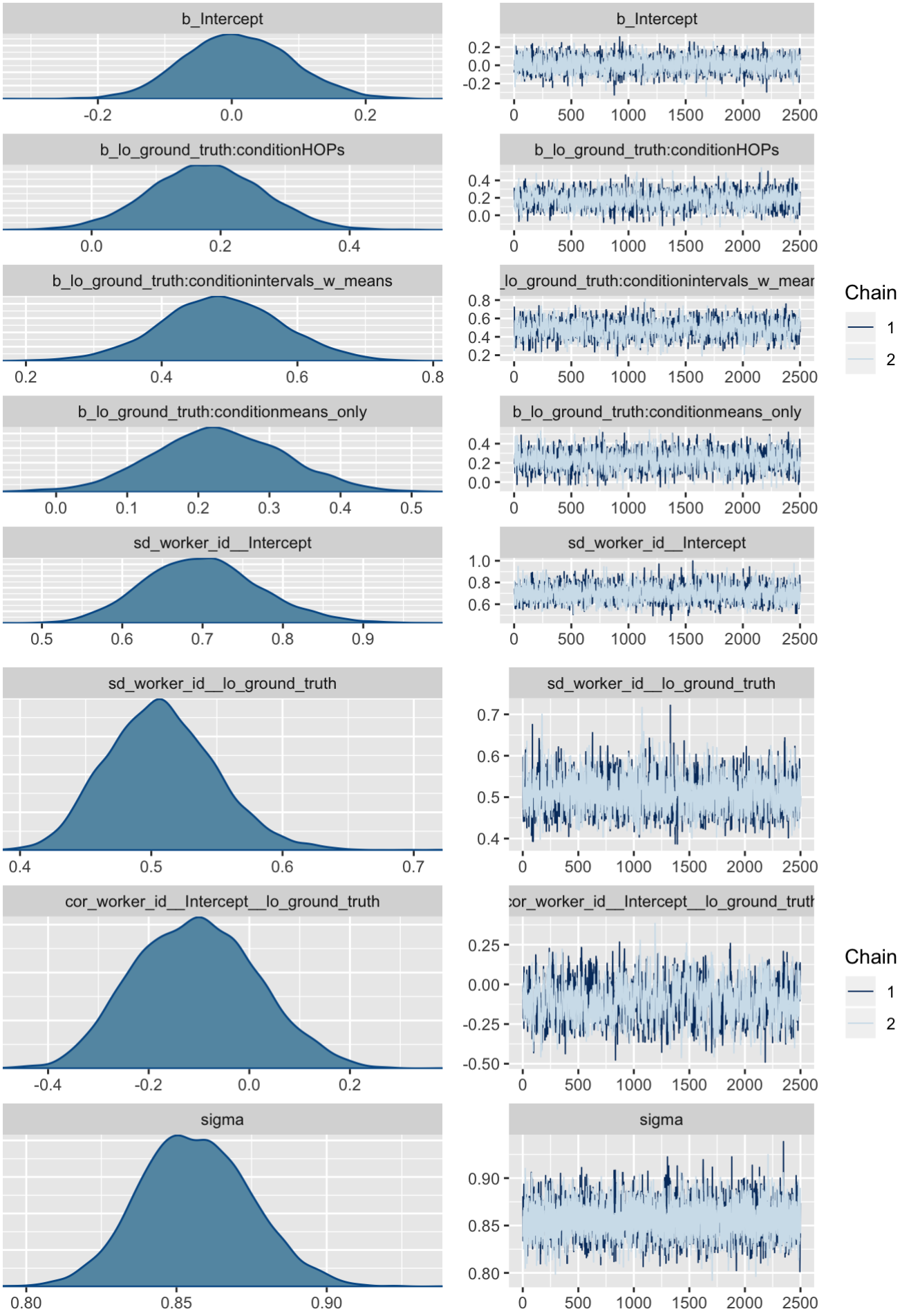
This is a hierarchical linear log odds (LLO) model of probability of superiority judgments which accounts for the effect of visualization and for individual differences. This is the best fitting version of the LLO model when fit to the full data set. Let's see how it works with only the first block of trials for each participant.

```
# update the llo model of p_sup responses to include an interaction
m.vis.wrkr.llo_p_sup <- brm(data = model_df_llo, family = gaussian,
                            formula = lo_p_sup ~ (1 + lo_ground_truth|worker_id) + lo
_ground_truth:condition,
                            prior = c(prior(normal(0, 1), class = Intercept),
                                      prior(normal(0, 1), class = b),
                                      prior(normal(0, 1), class = sigma)),
                            iter = 3000, warmup = 500, chains = 2, cores = 2,
                            file = "model-fits/llo_mdl_vis_wrkr-first_block")
```

Check diagnostics:

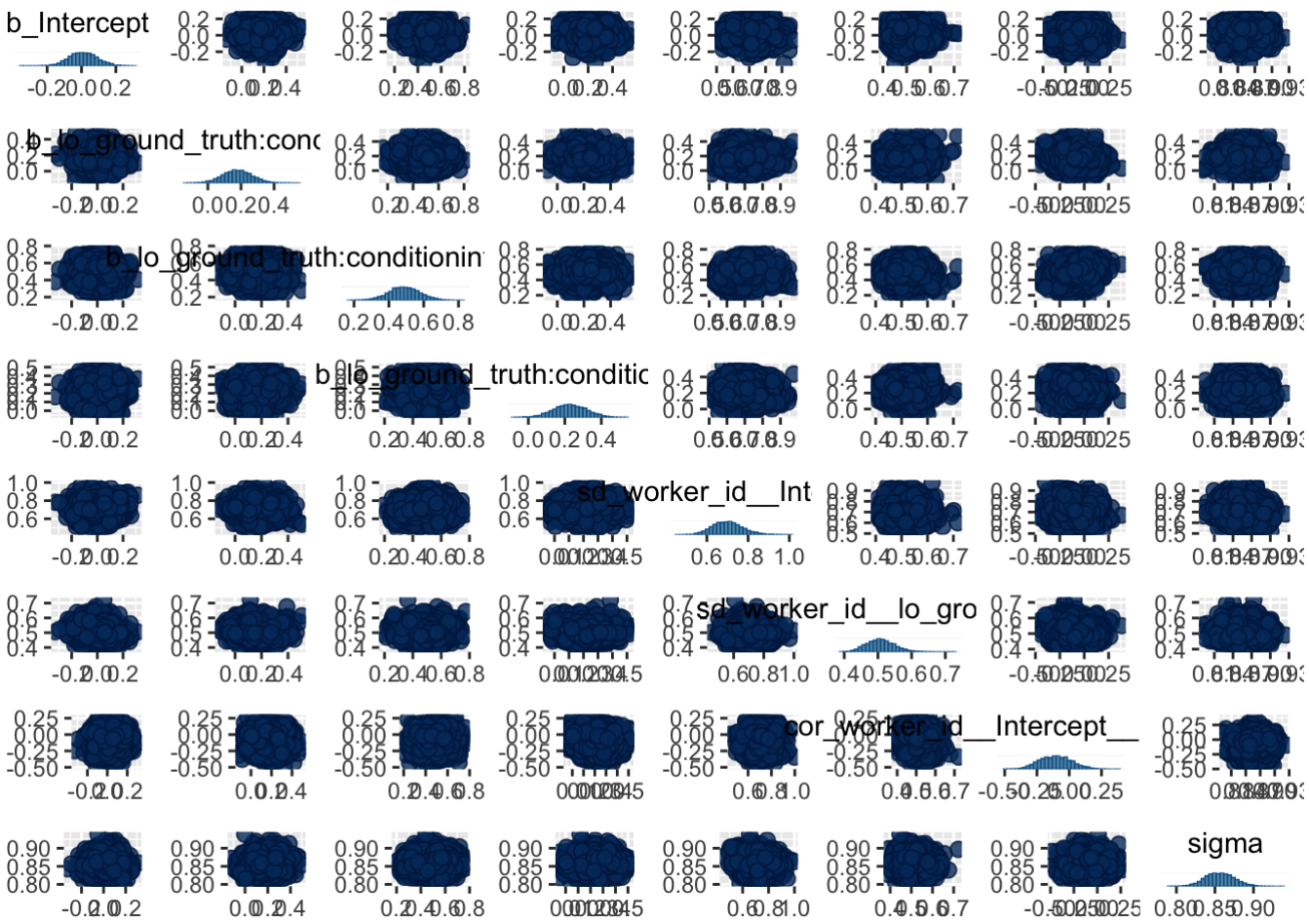
- Trace plots

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


b_Intercept
b_lo_ground_truth:conditionHOPs
b_lo_ground_truth:conditionintervals_w_means
b_lo_ground_truth:conditionmeans_only
sd_worker_id__Intercept
sd_worker_id__lo_ground_truth
cor_worker_id__Intercept__lo_ground_truth
sigma
Chain
1
2

- Pairs plot

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



- Summary

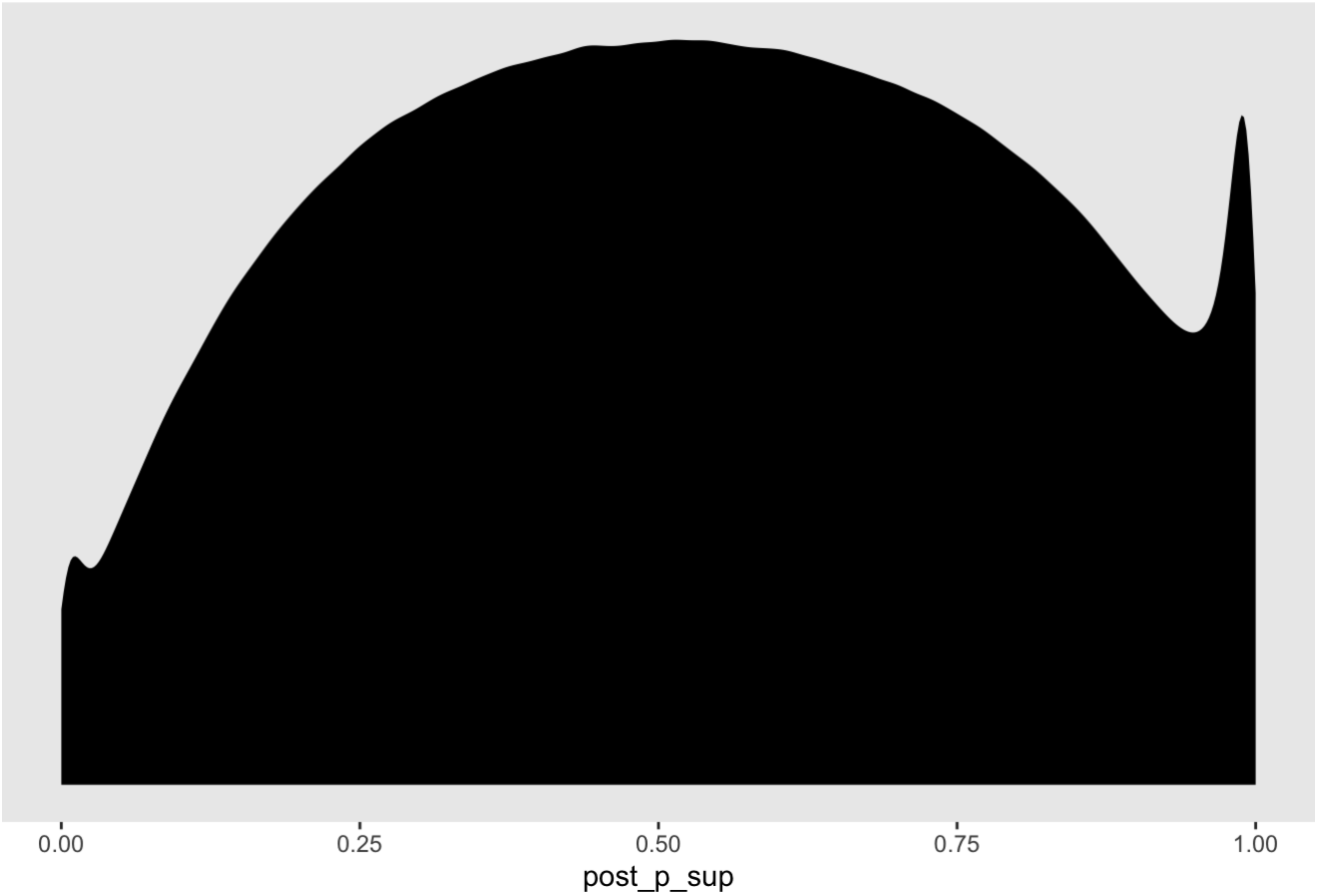
```
# model summary
print(m.vis.wrkr.llo_p_sup)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: lo_p_sup ~ (1 + lo_ground_truth | worker_id) + lo_ground_truth: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)                      0.70      0.07     0.57     0.85
## sd(lo_ground_truth)                0.51      0.04     0.43     0.59
## cor(Intercept,lo_ground_truth)    -0.11      0.12    -0.33     0.14
##                                Eff.Sample Rhat
## sd(Intercept)                        2608 1.00
## sd(lo_ground_truth)                  2211 1.00
## cor(Intercept,lo_ground_truth)        874 1.00
##
## Population-Level Effects:
##                                          Estimate Est.Error l-95% CI
## Intercept                                    0.01      0.08    -0.14
## lo_ground_truth:conditionHOPs                0.18      0.09     0.01
## lo_ground_truth:conditionintervals_w_means    0.49      0.09     0.31
## lo_ground_truth:conditionmeans_only           0.23      0.09     0.05
##                                          u-95% CI Eff.Sample Rhat
## Intercept                                    0.17       3005 1.00
## lo_ground_truth:conditionHOPs                0.35       1708 1.00
## lo_ground_truth:conditionintervals_w_means    0.66       1786 1.00
## lo_ground_truth:conditionmeans_only           0.41       1876 1.00
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## sigma     0.86      0.02     0.82     0.89       4540 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 out a posterior predictive distribution for probability of superiority.

```
# posterior predictive check
model_df_llo %>%
  select(lo_ground_truth, condition, worker_id) %>%
  add_predicted_draws(m.vis.wrkr.llo_p_sup, prediction = "lo_p_sup", seed = 1234) %>%
  mutate(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())
```

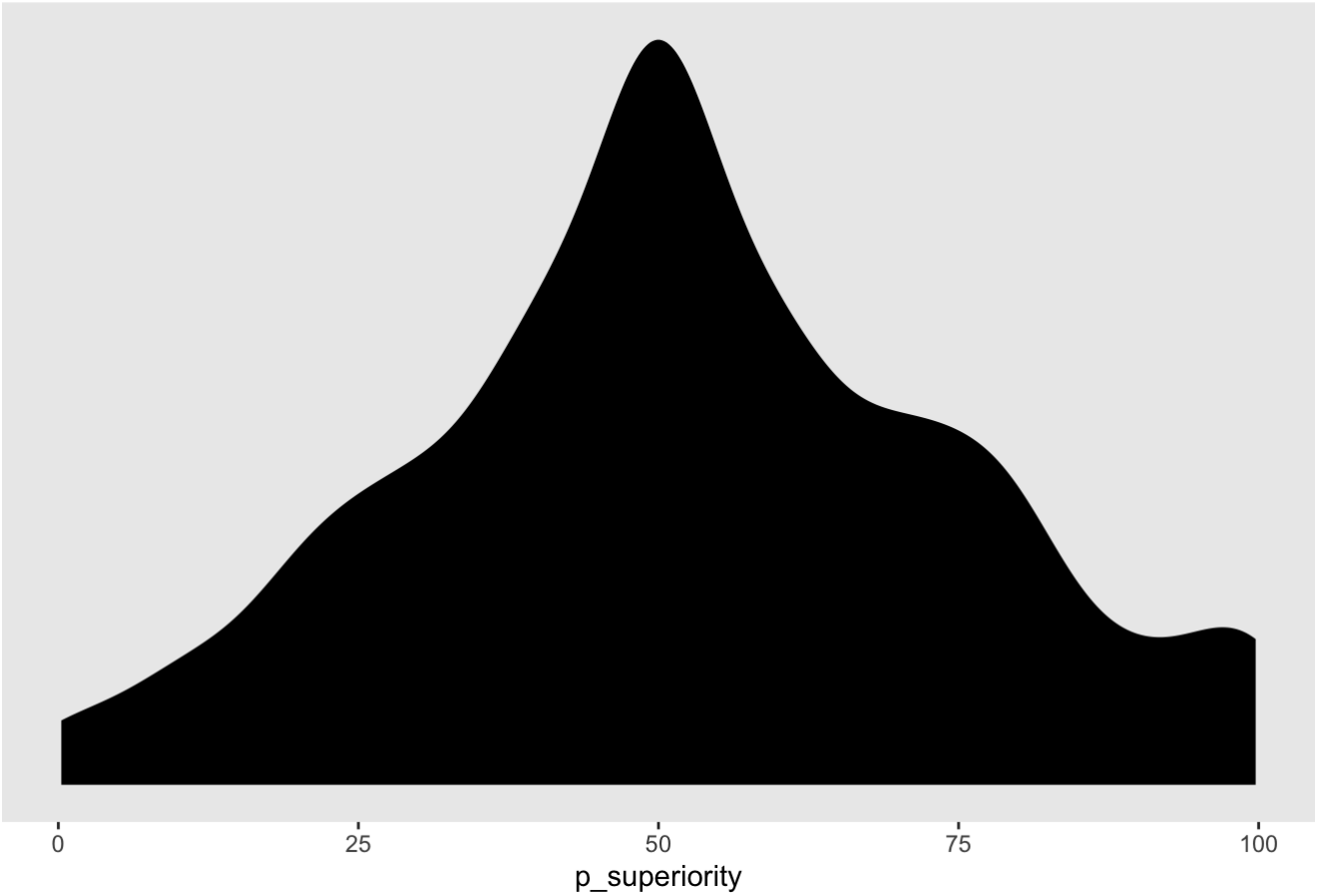
Posterior predictive distribution for probability of superiority



How does this compare to the empirical distribution of probability of superiority responses?

```
# posterior predictive check
model_df_llo %>%
  ggplot(aes(x = p_superiority)) +
  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())
```

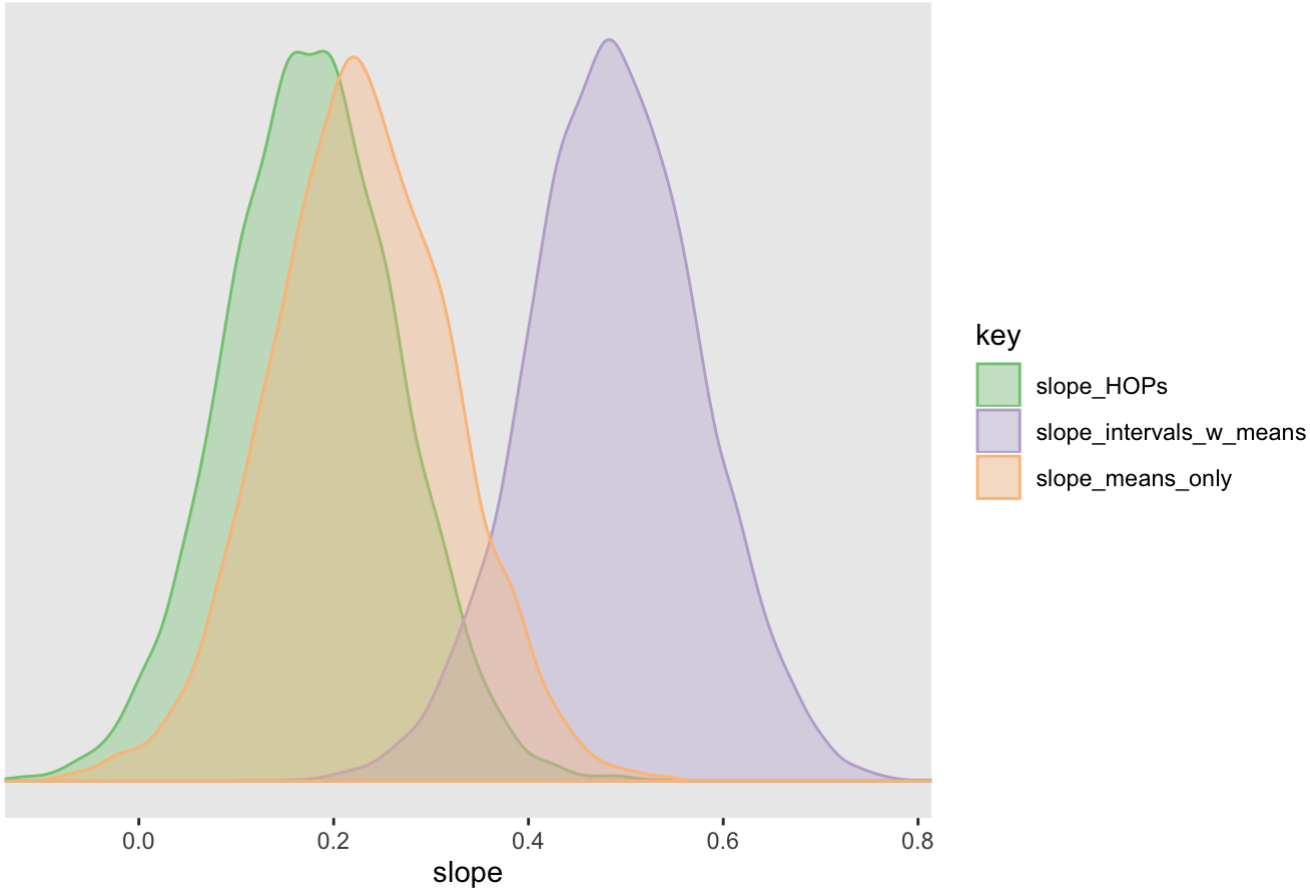
Posterior predictive distribution for probability of superiority



What do the posterior for the effect of each visualization condition look like?

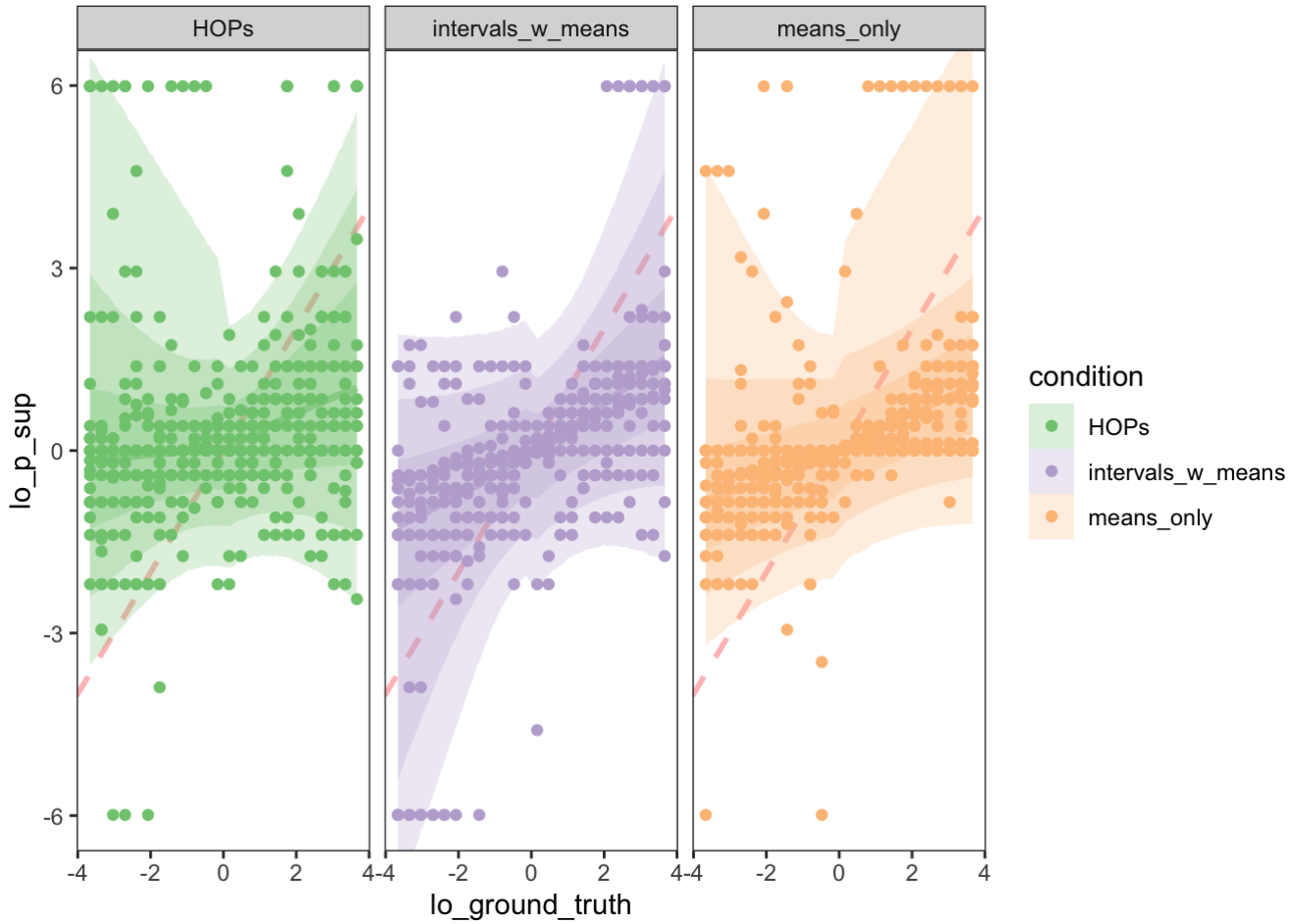
```
# use posterior samples to define distributions for the slope in each visualization c
ondition
posterior_samples(m.vis.wrkr.llo_p_sup) %>%
  # transmute(slope_HOPs = `b_conditionHOPs:lo_ground_truth`,
  #           slope_intervals_w_means = `b_conditionintervals_w_means:lo_ground_truth
`,
  #           slope_means_only = `b_conditionmeans_only:lo_ground_truth`) %>%
  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



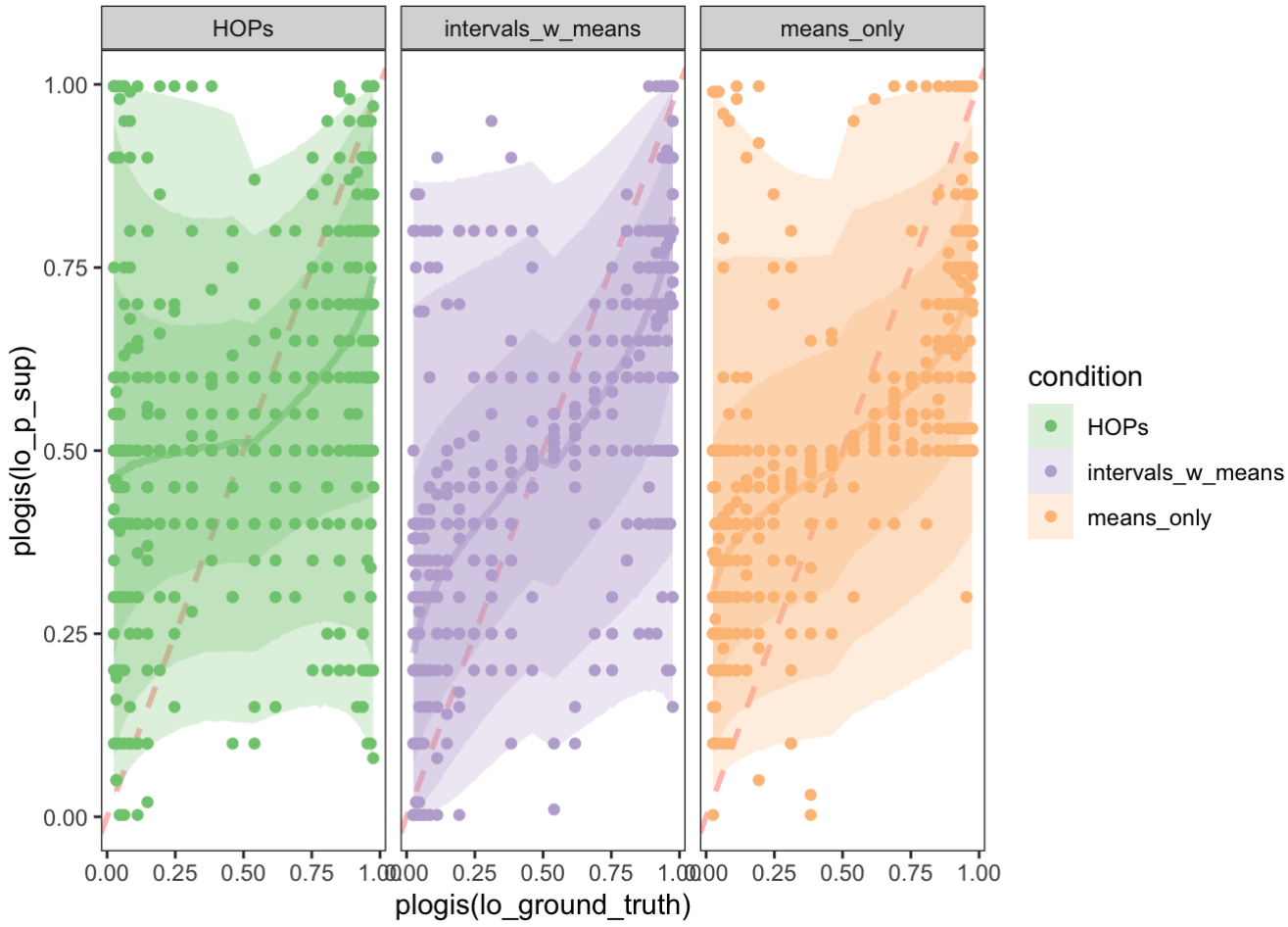
Let's take a look at some of the estimated linear models per visualization condition.

```
# this time we'll adopt functions from the tidybayes package to make plotting posteri
or predictions easier
model_df_llo %>%
  group_by(condition, worker_id) %>%
  data_grid(lo_ground_truth = seq_range(lo_ground_truth, n = 51)) %>%
  add_predicted_draws(m.vis.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_grid(. ~ condition)
```



What does this look like in probability units?

```
# this time we'll adopt functions from the tidybayes package to make plotting posteri
or predictions easier
model_df_llo %>%
  group_by(condition, worker_id) %>%
  data_grid(lo_ground_truth = seq_range(lo_ground_truth, n = 51)) %>%
  add_predicted_draws(m.vis.wrkr.llo_p_sup) %>%
  ggplot(aes(x = plogis(lo_ground_truth), y = plogis(lo_p_sup), color = condition, fi
ll = 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 = .2
5) +
  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_grid(. ~ condition)
```



This looks pretty good. However, there are areas of high posterior density where there are no observations, especially in the means_only condition. This is why we moved to a mixture model.

The asymmetry in these predictions is due to the random effect of worker. Because each worker only sees one frame, predictions are asymmetrical about the inflection point where ground truth is 50%. Also, there is obviously some confusion about how to use the response scale. Hopefully changes to the elicitation interface will clean things up in the next pilot.

Adding a Predictor for the Effect of Framing

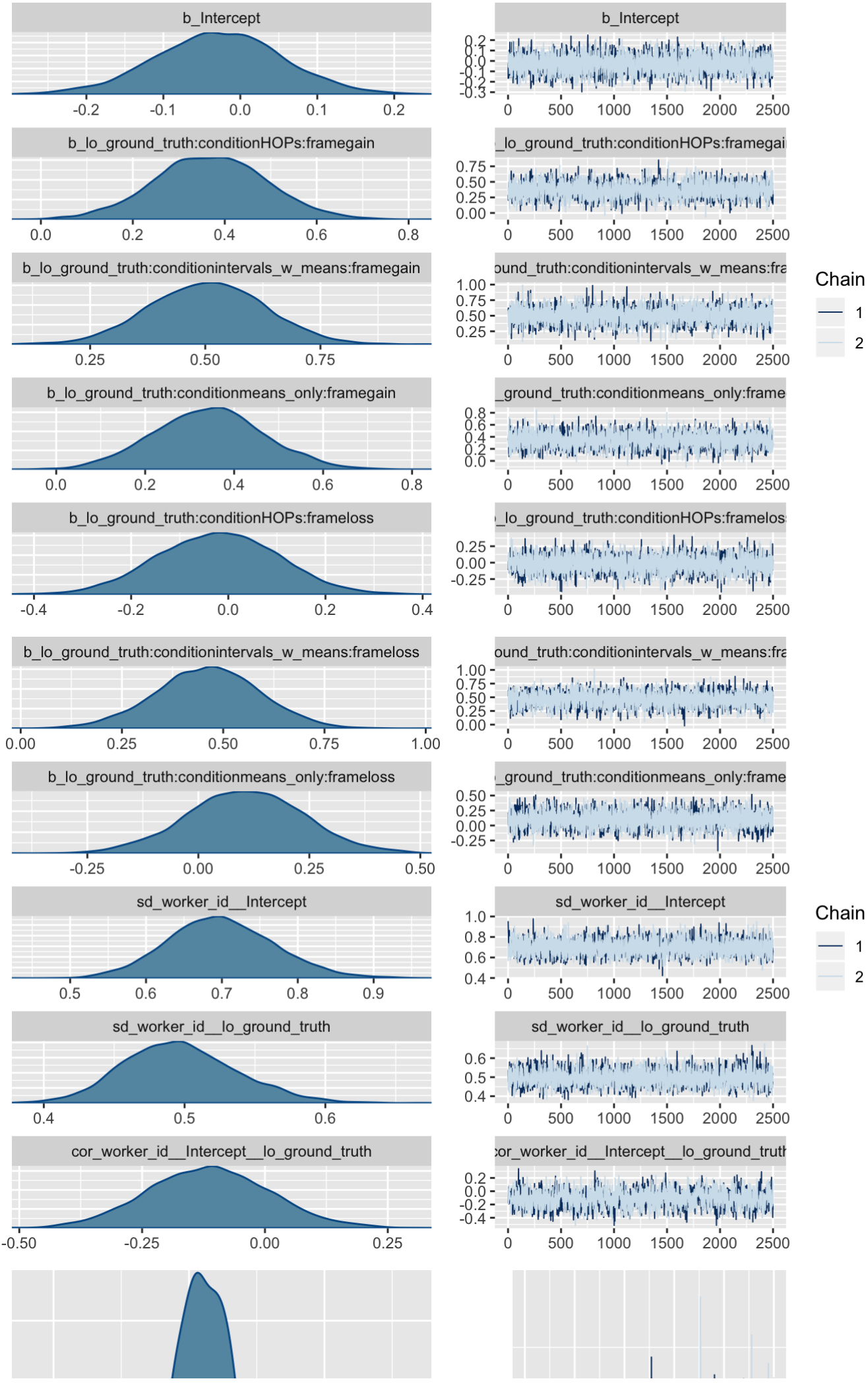
Let's look at the effect of framing. This is the same model as before, but now we've added a predictor to see if slopes are different in the gain vs loss framing conditions. Recall that slope effects are represented by interactions with the ground truth probability of superiority.

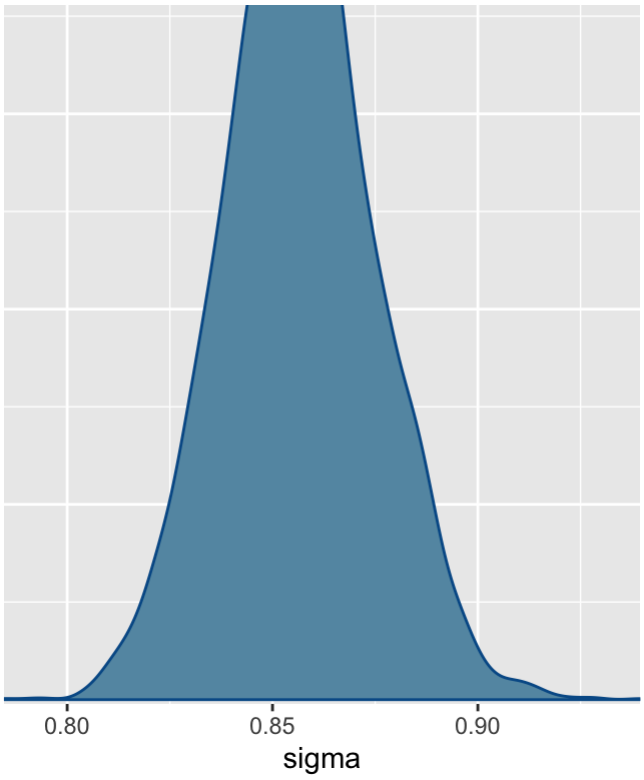
```
# update the llo model of p_sup responses to include an interaction
m.vis.frame.wrkr.llo_p_sup <- brm(data = model_df_llo, family = gaussian,
                                  formula = lo_p_sup ~ (1 + lo_ground_truth|worker_i
d) + lo_ground_truth:condition:frame,
                                  prior = c(prior(normal(0, 1), class = Intercept),
                                            prior(normal(0, 1), class = b),
                                            prior(normal(0, 1), class = sigma)),
                                  iter = 3000, warmup = 500, chains = 2, cores = 2,
                                  file = "model-fits/llo_mdl_vis_frame_wrkr-first_blo
ck")
```

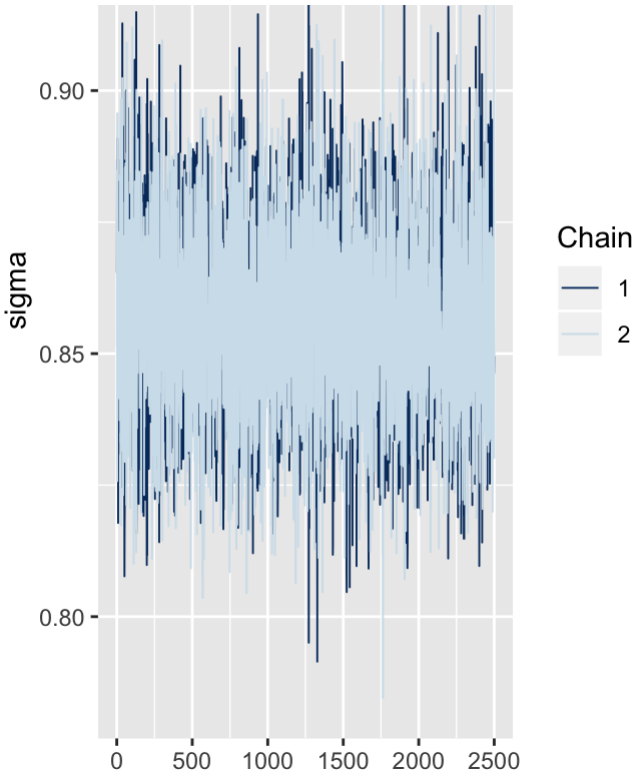
Check diagnostics:

- Trace plots

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

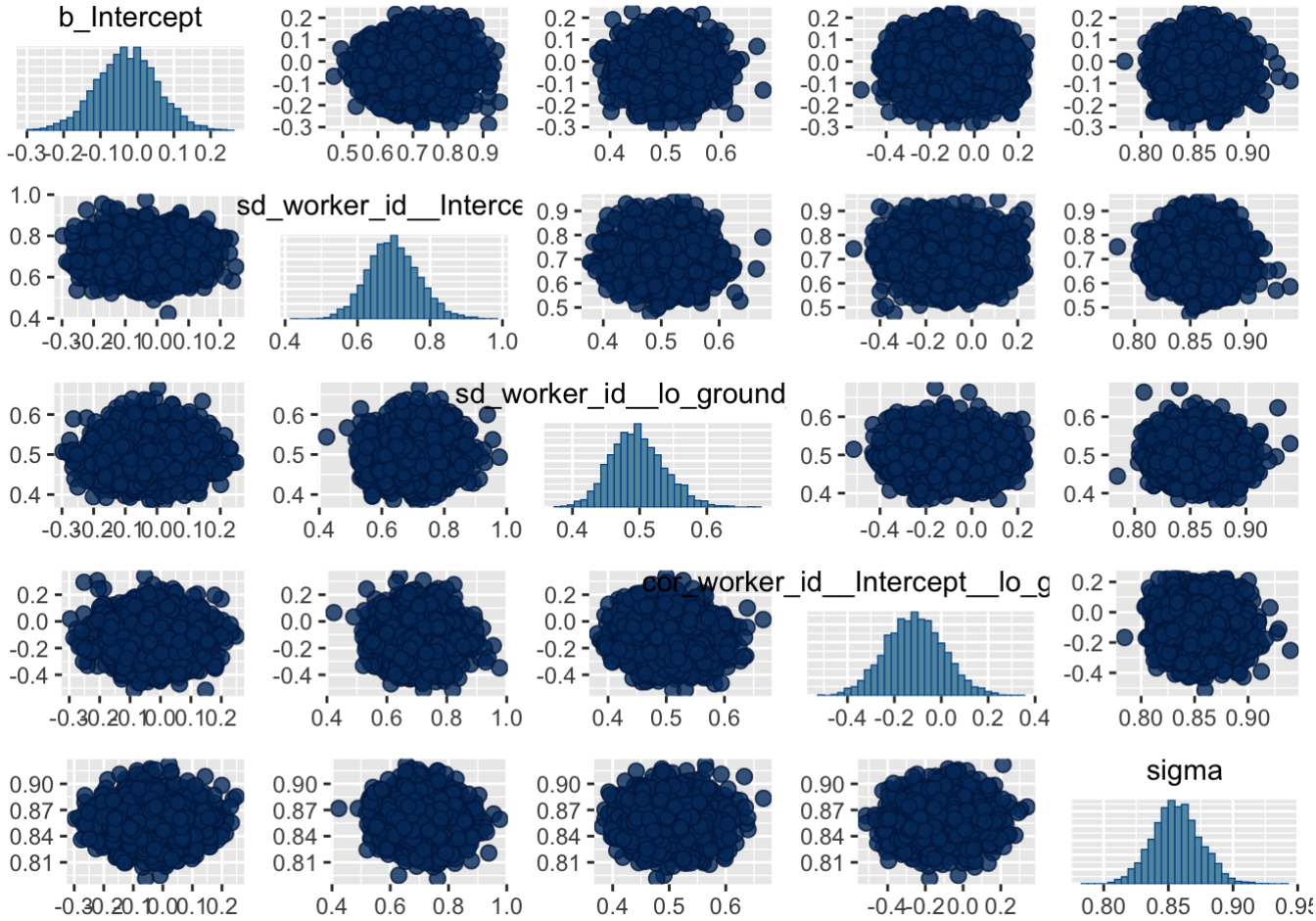

b_Intercept
b_lo_ground_truth:conditionHOPs:framegain
b_lo_ground_truth:conditionintervals_w_means:framegain
b_lo_ground_truth:conditionmeans_only:framegain
b_lo_ground_truth:conditionHOPs:frameloss
b_lo_ground_truth:conditionintervals_w_means:frameloss
b_lo_ground_truth:conditionmeans_only:frameloss
sd_worker_id__Intercept
sd_worker_id__lo_ground_truth
cor_worker_id__Intercept__lo_ground_truth
Chain
1
2



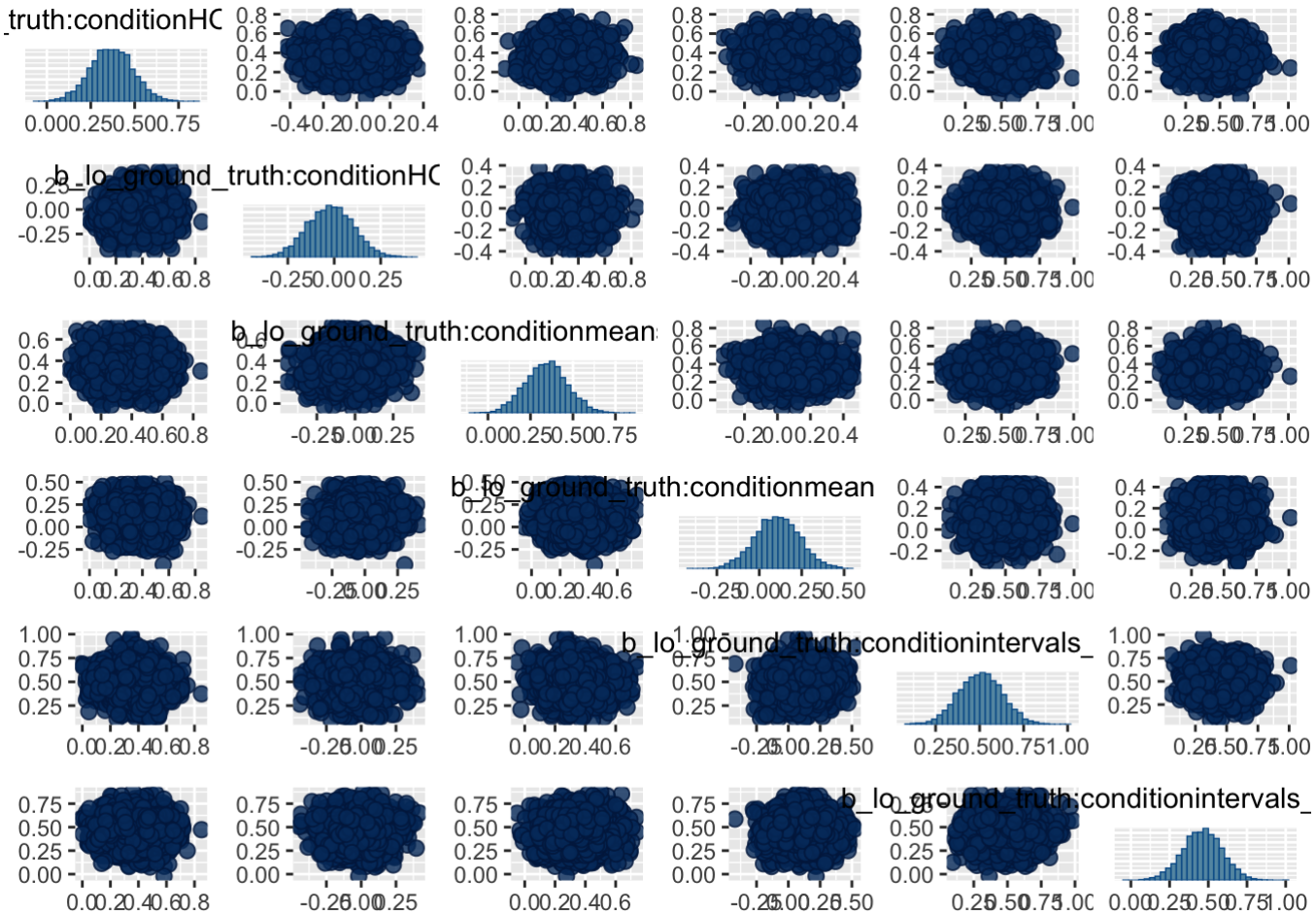


- Pairs plot

```
# pairs plot (too many things to view at once, so we've grouped them)
# hyperparameters
pairs(m.vis.frame.wrkr.llo_p_sup, pars = c("b_Intercept",
                                "sd_worker_id__Intercept",
                                "sd_worker_id__lo_ground_truth",
                                "cor_worker_id__Intercept__lo_ground_truth",
                                "sigma"))
```




```
# pairs plot (too many things to view at once, so we've grouped them)
# slope effects
pairs(m.vis.frame.wrkr.llo_p_sup, pars = c("b_lo_ground_truth:conditionHOPs:framegai
n",
                                           "b_lo_ground_truth:conditionHOPs:framelos
s",
                                           "b_lo_ground_truth:conditionmeans_only:fra
megain",
                                           "b_lo_ground_truth:conditionmeans_only:fra
meloss",
                                           "b_lo_ground_truth:conditionintervals_w_me
ans:framegain",
                                           "b_lo_ground_truth:conditionintervals_w_me
ans:frameloss"))
```



- Summary

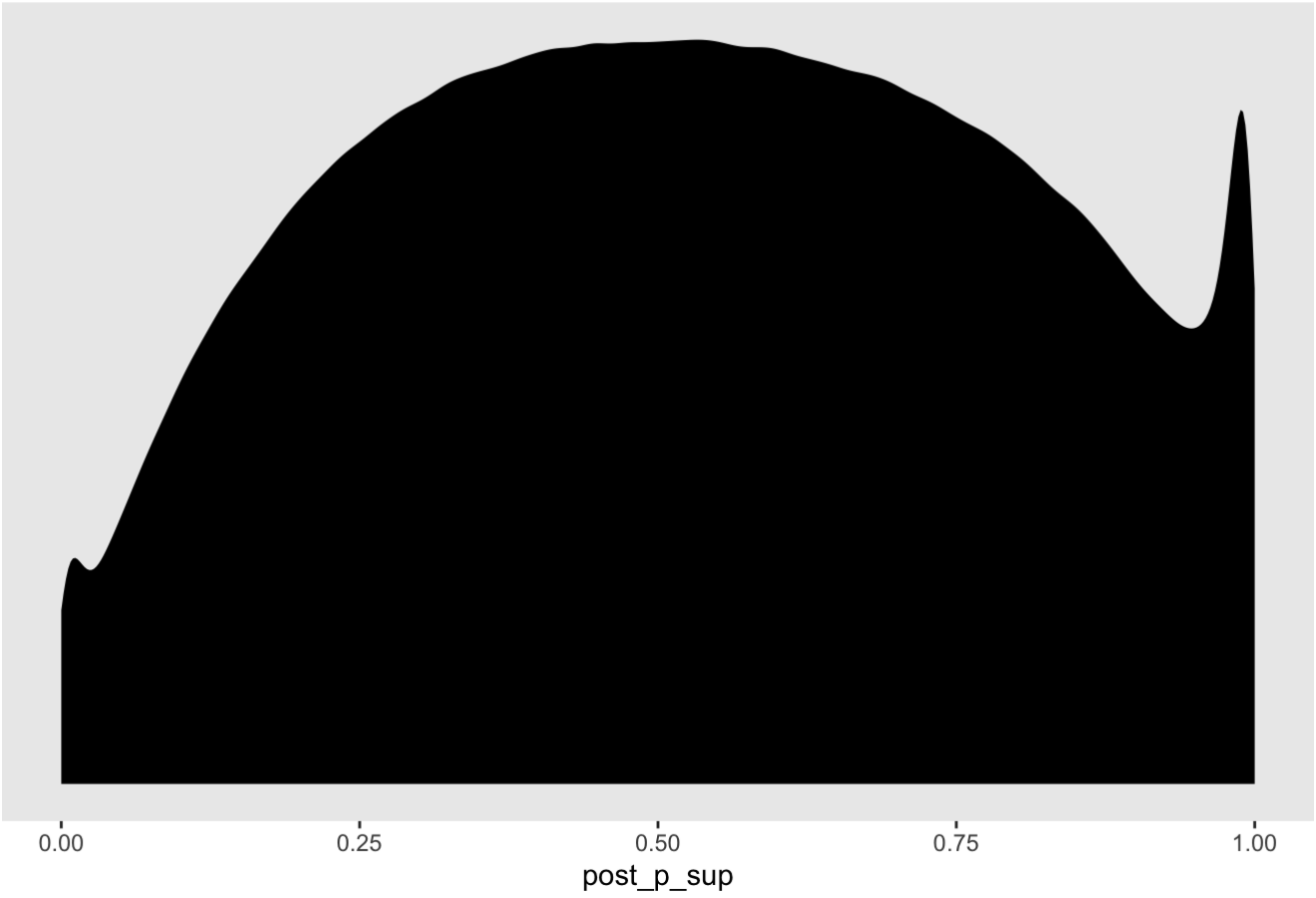
```
# model summary
print(m.vis.frame.wrkr.llo_p_sup)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: lo_p_sup ~ (1 + lo_ground_truth | worker_id) + lo_ground_truth:condition:
frame
##    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)                     0.70      0.07     0.57     0.84
## sd(lo_ground_truth)               0.50      0.04     0.42     0.58
## cor(Intercept,lo_ground_truth)   -0.11      0.12    -0.35     0.13
##                               Eff.Sample Rhat
## sd(Intercept)                       2927 1.00
## sd(lo_ground_truth)                 2785 1.00
## cor(Intercept,lo_ground_truth)       899 1.00
##
## Population-Level Effects:
##                                                   Estimate Est.Error
## Intercept                                            -0.03      0.08
## lo_ground_truth:conditionHOPs:framegain               0.37      0.12
## lo_ground_truth:conditionintervals_w_means:framegain  0.50      0.12
## lo_ground_truth:conditionmeans_only:framegain         0.34      0.12
## lo_ground_truth:conditionHOPs:frameloss              -0.02      0.12
## lo_ground_truth:conditionintervals_w_means:frameloss  0.46      0.13
## lo_ground_truth:conditionmeans_only:frameloss         0.11      0.13
##                                                   l-95% CI u-95% CI
## Intercept                                            -0.20     0.13
## lo_ground_truth:conditionHOPs:framegain               0.13     0.61
## lo_ground_truth:conditionintervals_w_means:framegain  0.26     0.74
## lo_ground_truth:conditionmeans_only:framegain         0.10     0.59
## lo_ground_truth:conditionHOPs:frameloss              -0.26     0.22
## lo_ground_truth:conditionintervals_w_means:frameloss  0.21     0.71
## lo_ground_truth:conditionmeans_only:frameloss        -0.15     0.37
##                                                   Eff.Sample Rhat
## Intercept                                               4130 1.00
## lo_ground_truth:conditionHOPs:framegain                 2114 1.00
## lo_ground_truth:conditionintervals_w_means:framegain    2381 1.00
## lo_ground_truth:conditionmeans_only:framegain           2518 1.00
## lo_ground_truth:conditionHOPs:frameloss                 2812 1.00
## lo_ground_truth:conditionintervals_w_means:frameloss    3106 1.00
## lo_ground_truth:conditionmeans_only:frameloss           2898 1.00
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## sigma     0.86      0.02     0.82     0.89       5791 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 out a posterior predictive distribution for probability of superiority.

```
# posterior predictive check
model_df_llo %>%
  select(lo_ground_truth, condition, frame, worker_id) %>%
  add_predicted_draws(m.vis.frame.wrkr.llo_p_sup, prediction = "lo_p_sup", seed = 123
4) %>%
  mutate(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())
```

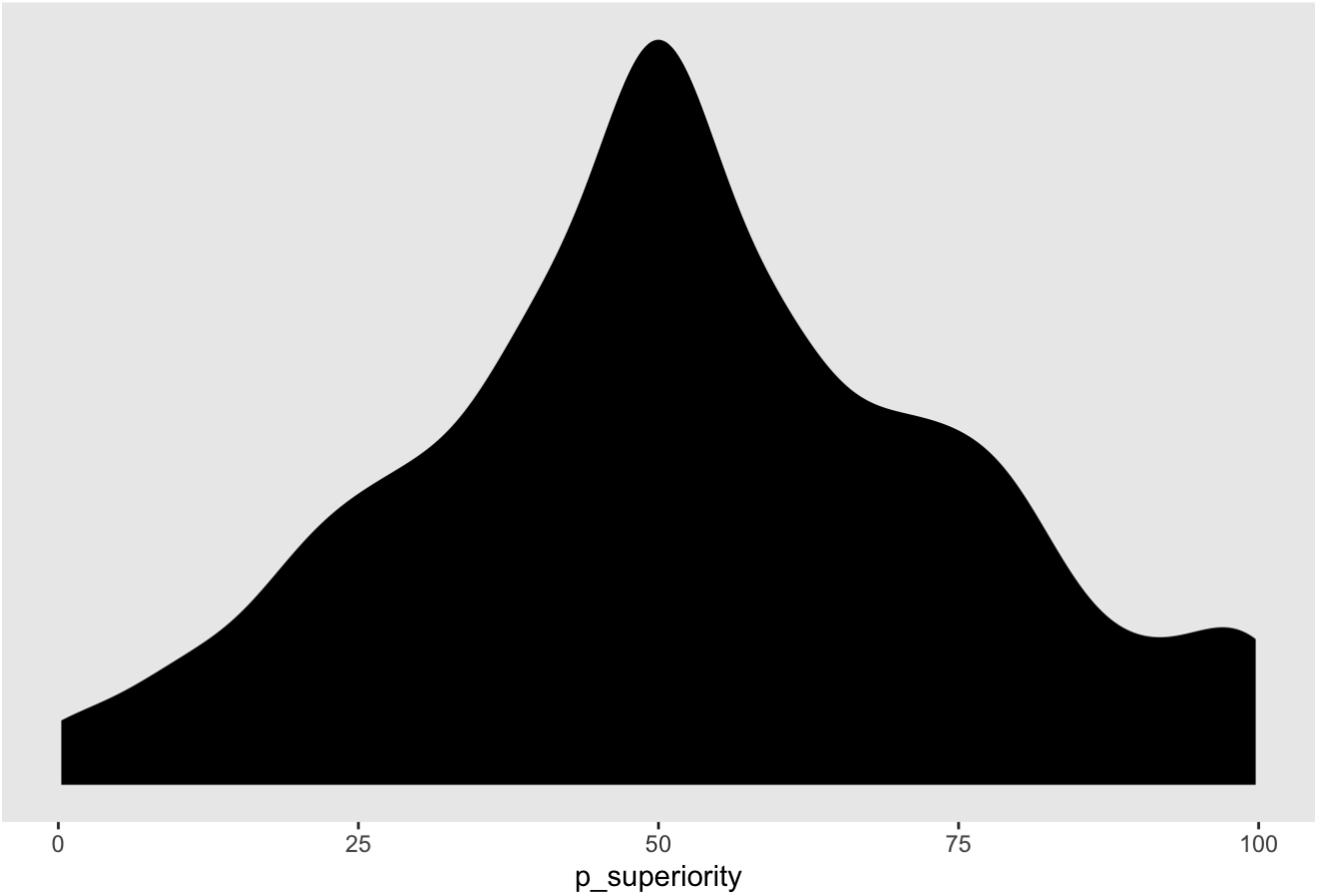
Posterior predictive distribution for probability of superiority



How does this compare to the empirical distribution of probability of superiority responses?

```
# posterior predictive check
model_df_llo %>%
  ggplot(aes(x = p_superiority)) +
  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())
```

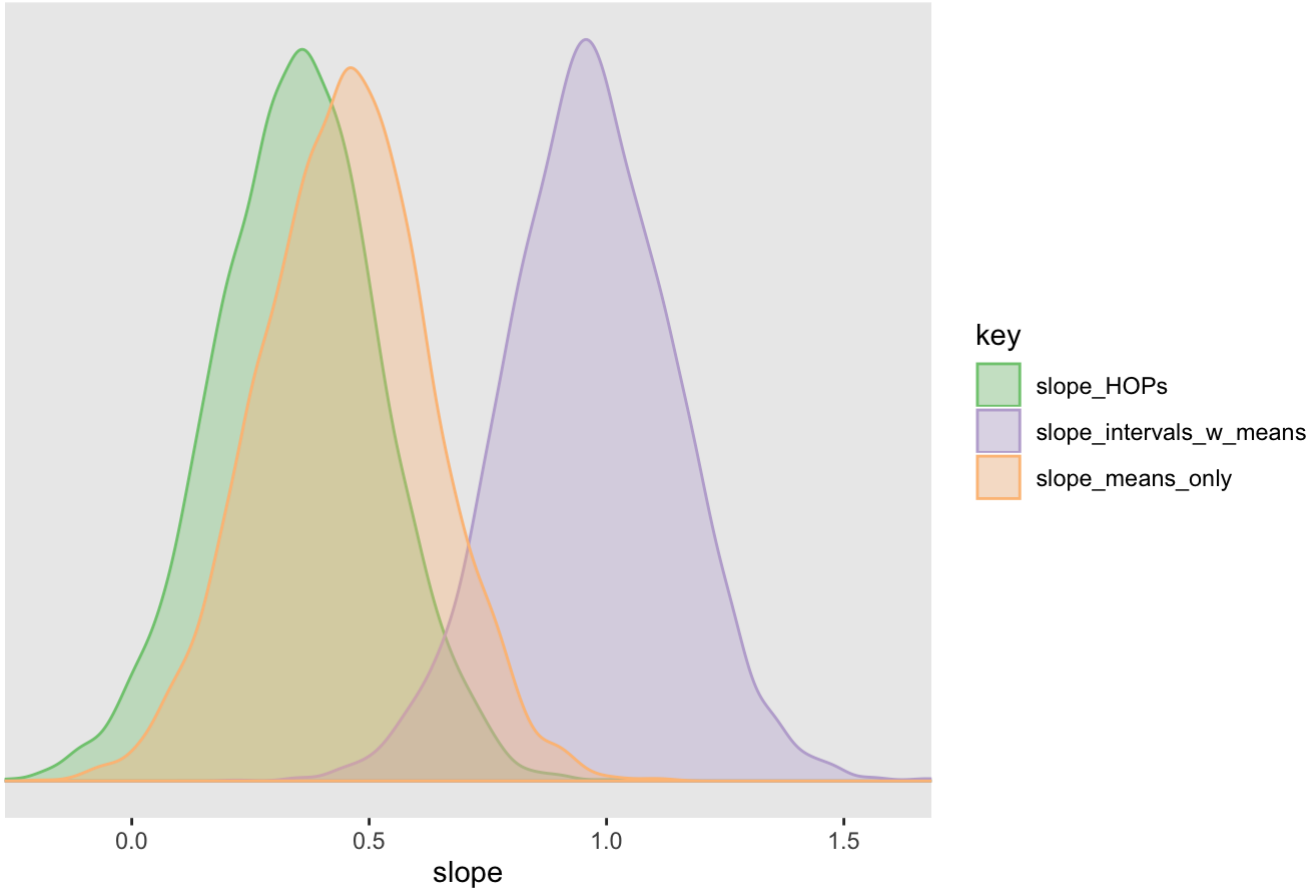
Posterior predictive distribution for probability of superiority



What do the posterior for the effect of each visualization condition look like?

```
# use posterior samples to define distributions for the slope in each visualization c
ondition
posterior_samples(m.vis.frame.wrkr.llo_p_sup) %>%
  transmute(slope_HOPs = `b_lo_ground_truth:conditionHOPs:framegain` + `b_lo_ground_t
ruth:conditionHOPs:frameloss`,
            slope_intervals_w_means = `b_lo_ground_truth:conditionintervals_w_means:f
ramegain` + `b_lo_ground_truth:conditionintervals_w_means:frameloss`,
            slope_means_only = `b_lo_ground_truth:conditionmeans_only:framegain` + `b
_lo_ground_truth:conditionmeans_only:frameloss`) %>%
  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())
```

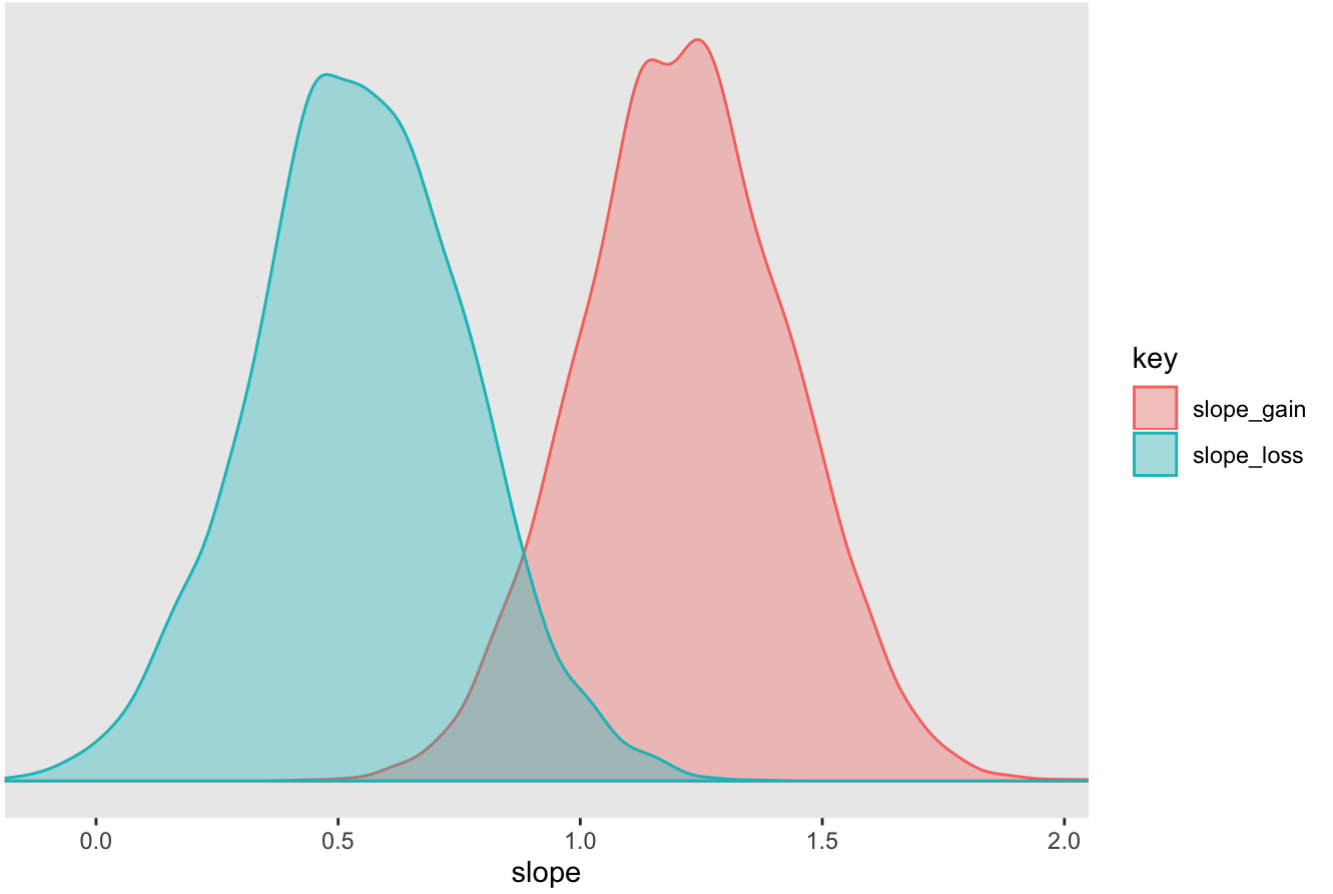
Posterior for slopes by visualization condition



What does the posterior for the effect of framing look like?

```
# use posterior samples to define distributions for the slope in the gain vs loss fra
ming conditions
posterior_samples(m.vis.frame.wrkr.llo_p_sup) %>%
  transmute(slope_gain = `b_lo_ground_truth:conditionHOPs:framegain` + `b_lo_ground_t
ruth:conditionintervals_w_means:framegain` +  `b_lo_ground_truth:conditionmeans_only:
framegain`,
            slope_loss = `b_lo_ground_truth:conditionHOPs:frameloss` + `b_lo_ground_t
ruth:conditionintervals_w_means:frameloss` + `b_lo_ground_truth:conditionmeans_only:f
rameloss`) %>%
  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 gain vs loss framing") +
  theme(panel.grid = element_blank())
```

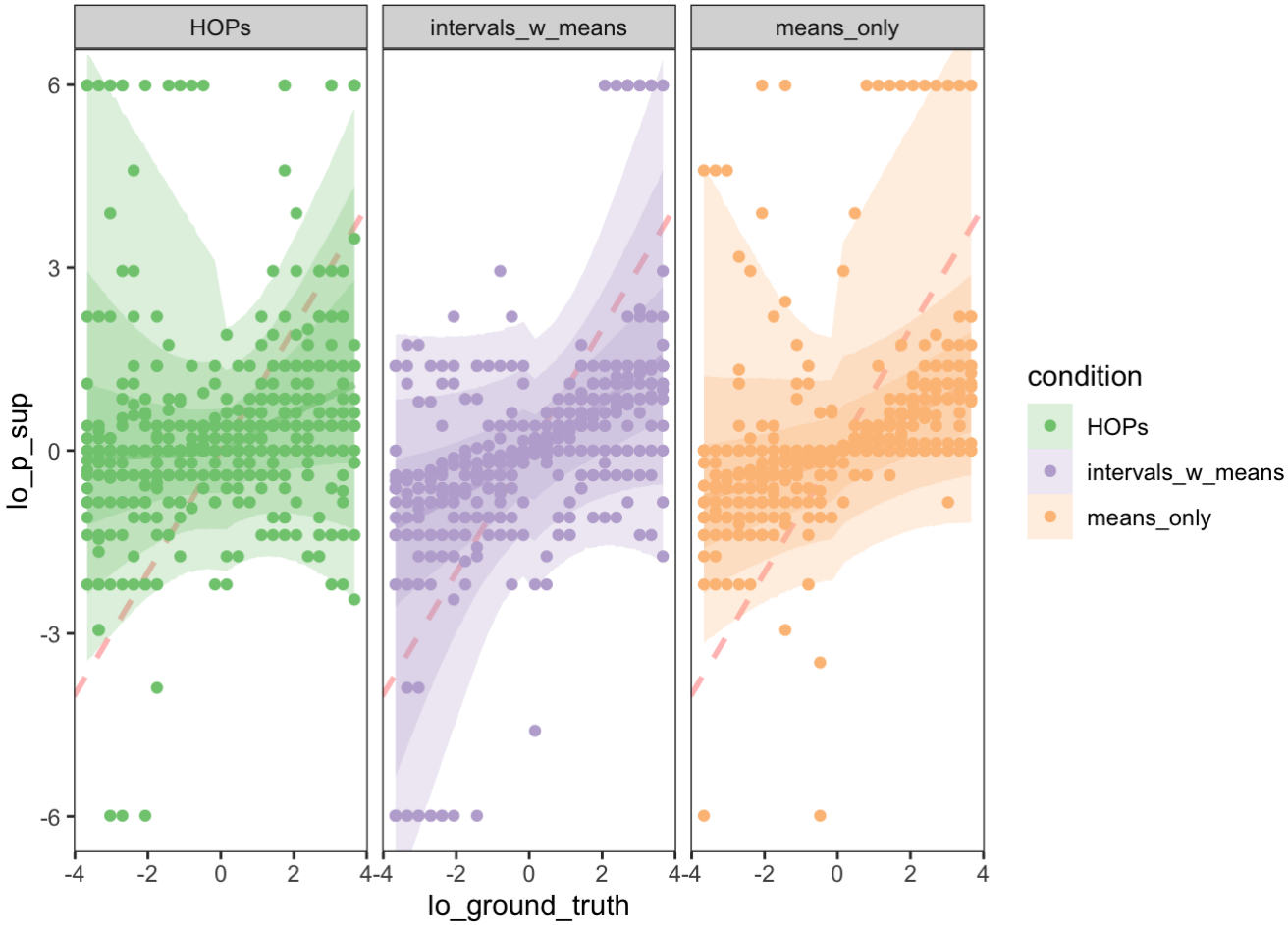
Posterior for slopes for gain vs loss framing



The chart above suggests that probability of superiority judgments are biased in a qualitatively different way (i.e., slopes on opposite sides of 1.0) depending on gain vs loss framing. However, this is a noisy estimate.

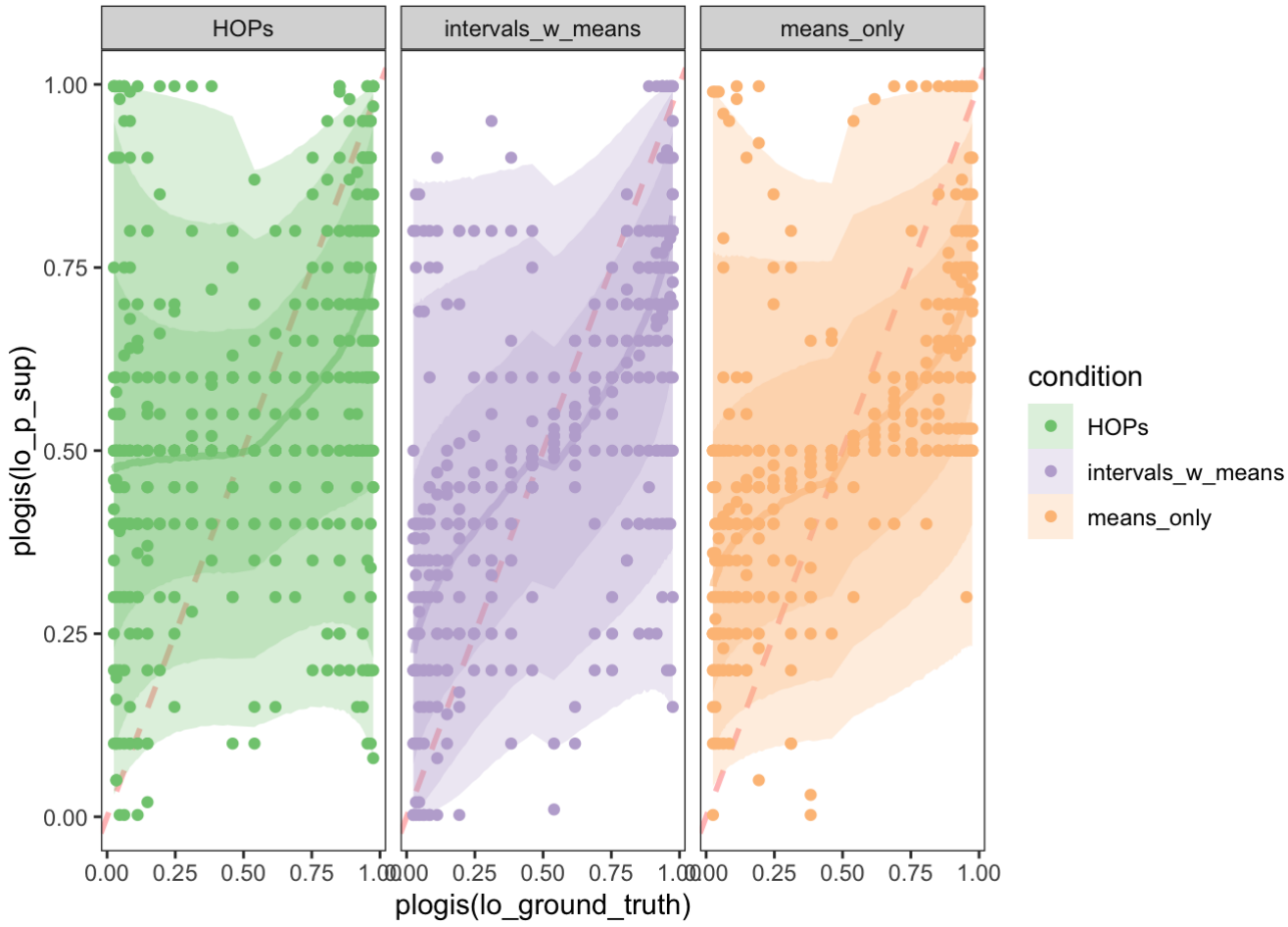
Let's take a look at some of the estimated linear models per visualization condition.

```
model_df_llo %>%
  group_by(condition, frame, worker_id) %>%
  data_grid(lo_ground_truth = seq_range(lo_ground_truth, n = 51)) %>%
  add_predicted_draws(m.vis.frame.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_grid(. ~ condition)
```



What does this look like in probability units?

```
model_df_llo %>%
  group_by(condition, frame, worker_id) %>%
  data_grid(lo_ground_truth = seq_range(lo_ground_truth, n = 51)) %>%
  add_predicted_draws(m.vis.frame.wrkr.llo_p_sup) %>%
  ggplot(aes(x = plogis(lo_ground_truth), y = plogis(lo_p_sup), color = condition, fi
ll = 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 = .2
5) +
  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_grid(. ~ condition)
```



Note the asymmetry above and below a ground truth of 50%, especially in the HOPs condition.

Again, there are areas of high posterior density where there are no observations. In the next section, we switch to a mixture model which fixes this.

Model Comparison

Let's check whether adding parameters to account for framing is worth it insofar as the added parameters contribute more to predictive validity than they contribute to overfitting. We'll determine this by comparing the models with and without parameters for frame according to the widely applicable information criterion (WAIC). Lower values of WAIC indicate a better fitting model.

```
waic(m.vis.wrkr.llo_p_sup, m.vis.frame.wrkr.llo_p_sup)
```

```
##                                                    WAIC     SE
## m.vis.wrkr.llo_p_sup                            3501.46 144.57
## m.vis.frame.wrkr.llo_p_sup                      3500.61 143.98
## m.vis.wrkr.llo_p_sup - m.vis.frame.wrkr.llo_p_sup    0.85   2.92
```

The fact that WAIC values for the two models are approximately equal suggests that we don't get much improvement in model fit from adding the parameters for gain vs loss framing to this model. This makes sense as the problem framing is more likely to impact decisions than probability judgments.

Mixture of the LLO Model and a Random Constant Response

Our Best Model of Probability of Superiority So Far

This is an adaptation of the LLO model with predictors for visualization and worker which incorporates a random constant response process to account for the inflation of responses near 50%. We use a multivariate normal prior for the rate of random constant responses in each visualization condition. This allows the model to learn both the mean rate for each visualization condition `mu_theta2` (in log odds units) and the shared covariance matrix for the rates in each condition `Sigma_theta2`.

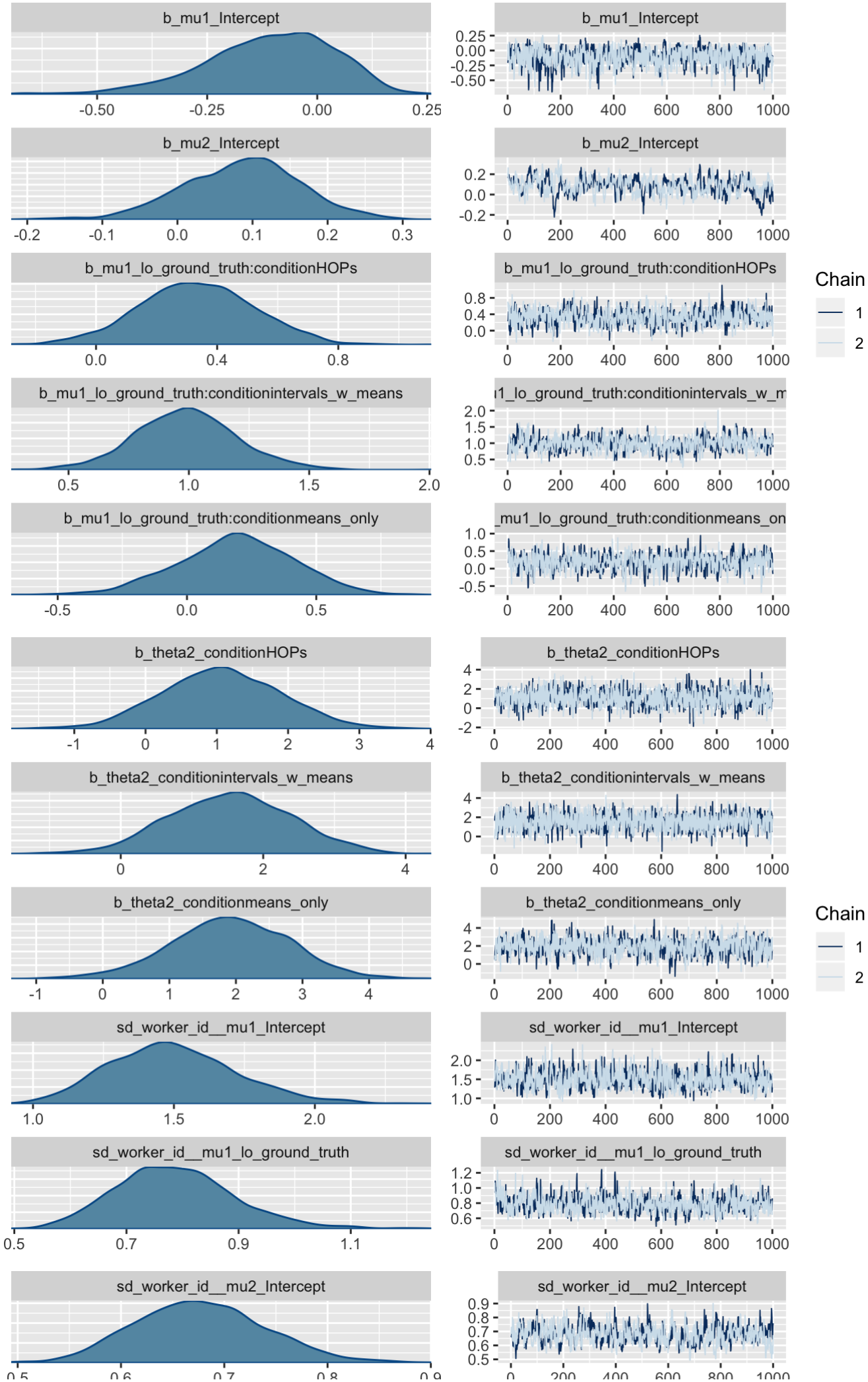
Again, let's see how this works with the data for only the first block of trials.

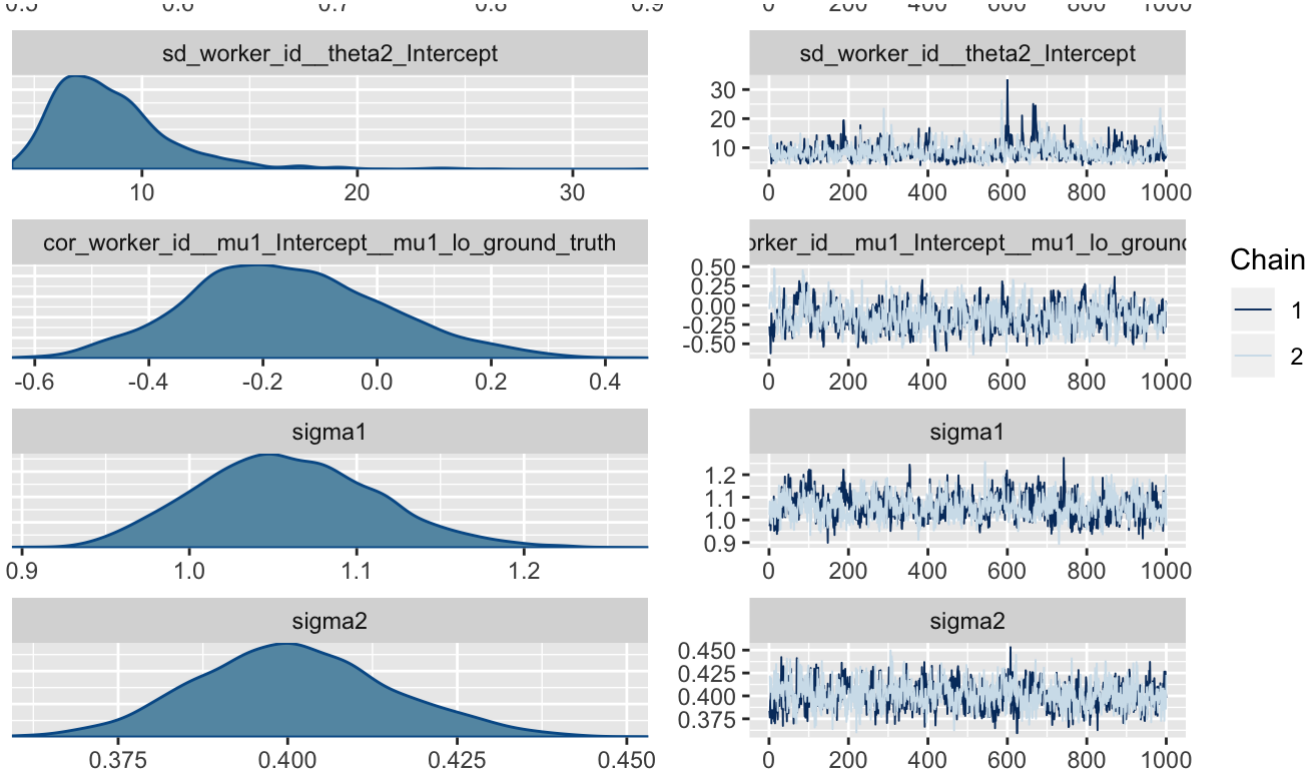
```
# define stanvars for multi_normal prior on condition effects
stanvars <- stanvar(rep(1, 3), "mu_theta2", scode = "  vector[3] mu_theta2;") +
  stanvar(diag(3), "Sigma_theta2", scode = "  matrix[3, 3] Sigma_theta2;")

# fit the model
m.vis.wrkr.llo_mix <- brm(
   bf(lo_p_sup ~ 1,
    mu1 ~ (1 + lo_ground_truth|worker_id) + lo_ground_truth:condition, # our most rec
ent llo model
    mu2 ~ (1|worker_id), # random constant response per worker (to account for people
who always answer the same, often but not always 50%)
    theta2 ~ (1|worker_id) + 0 + condition # the proportion of responses that are con
stant
  ),
  data = model_df_llo,
  family = mixture(gaussian, gaussian, order = 'mu'),
  prior = c(
    prior(normal(0, 1), class = Intercept, dpar = mu1),
    prior(normal(0, 1), class = Intercept, dpar = mu2),
    prior("multi_normal(mu_theta2, Sigma_theta2)", class = b, dpar = theta2)
  ),
  stanvars = stanvars,
  inits = 1, chains = 2, cores = 2,
  control = list(adapt_delta = 0.999, max_treedepth=15),
  file = "model-fits/llo_mix_mdl_vis_wrkr-first_block"
)
```

Check diagnostics:

- Trace plots

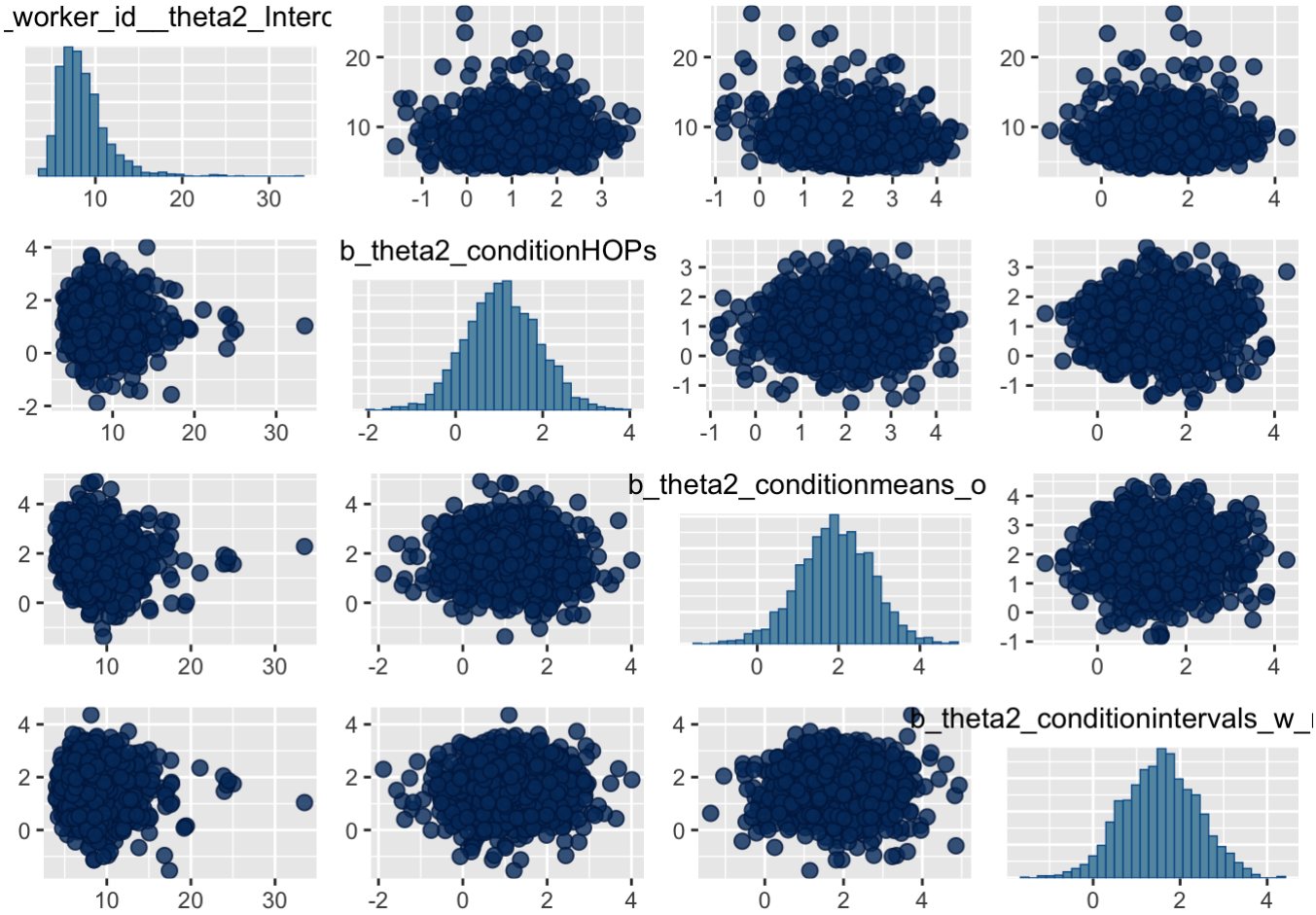
```
# trace plots
plot(m.vis.wrkr.llo_mix)
```


b_mu1_Intercept
b_mu2_Intercept
b_mu1_lo_ground_truth:conditionHOPs
b_mu1_lo_ground_truth:conditionintervals_w_means
b_mu1_lo_ground_truth:conditionmeans_only
b_theta2_conditionHOPs
b_theta2_conditionintervals_w_means
b_theta2_conditionmeans_only
sd_worker_id__mu1_Intercept
sd_worker_id__mu1_lo_ground_truth
sd_worker_id__mu2_Intercept
Chain
1
2

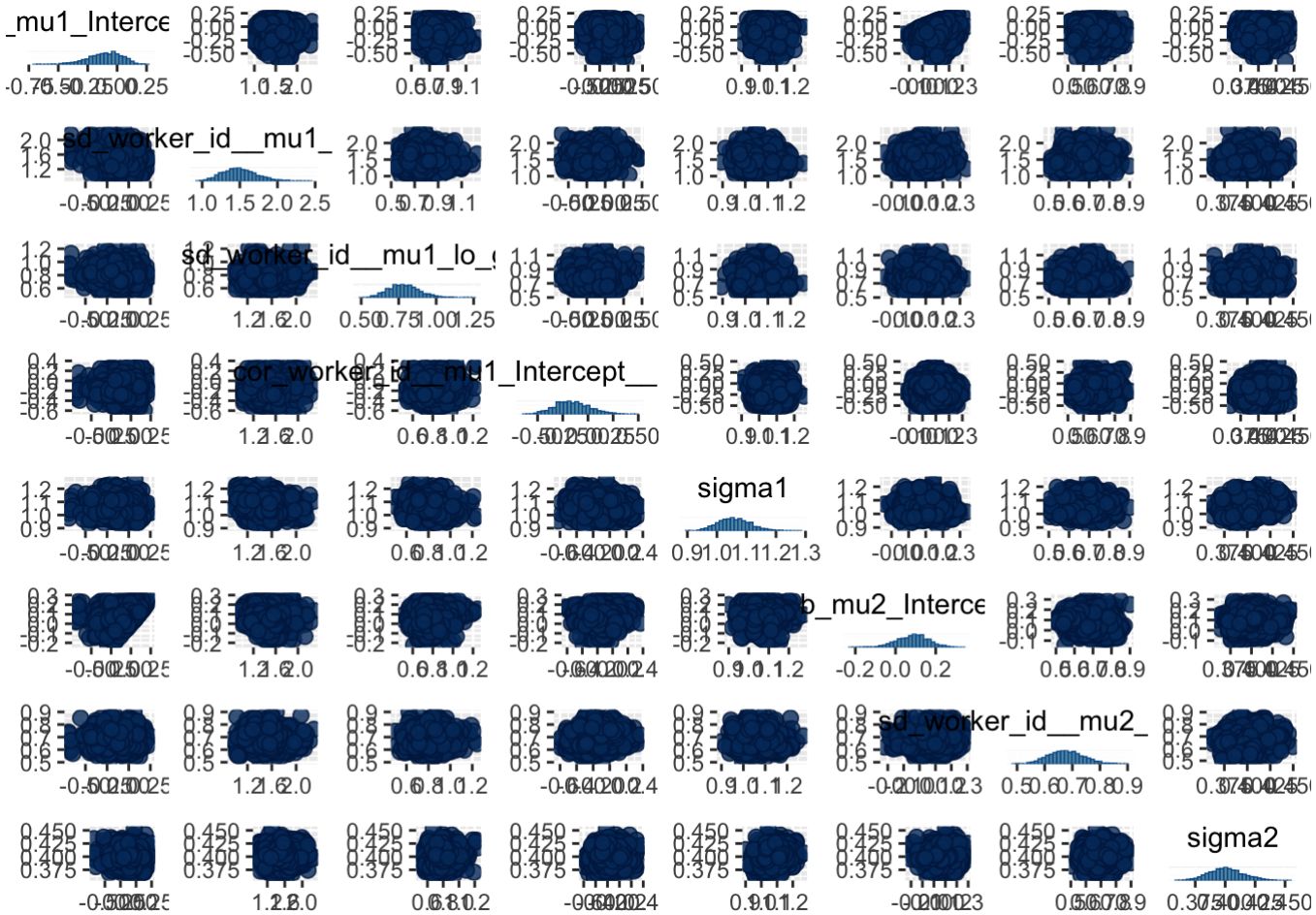


- Pairs plot

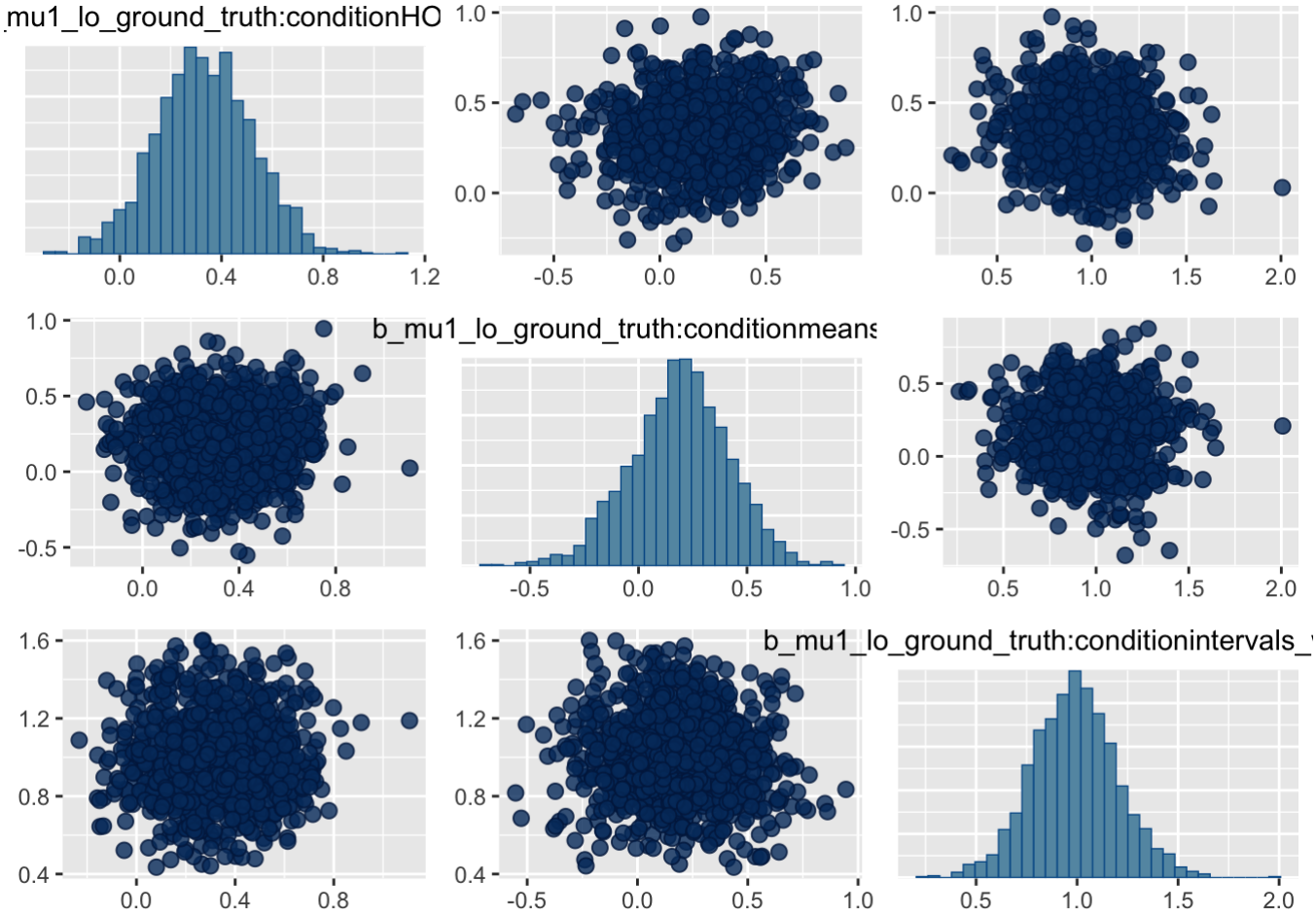
```
# pairs plot (too many things to view at once, so we've grouped them)
# mixture proportions
pairs(m.vis.wrkr.llo_mix, pars = c("sd_worker_id__theta2_Intercept",
                                   "b_theta2_conditionHOPs",
                                   "b_theta2_conditionmeans_only",
                                   "b_theta2_conditionintervals_w_means"))
```




```
# pairs plot (too many things to view at once, so we've grouped them)
# hyperparameters
pairs(m.vis.wrkr.llo_mix, pars = c("b_mu1_Intercept",
                                   "sd_worker_id__mu1_Intercept",
                                   "sd_worker_id__mu1_lo_ground_truth",
                                   "cor_worker_id__mu1_Intercept__mu1_lo_ground_trut
h",
                                   "sigma1",
                                   "b_mu2_Intercept",
                                   "sd_worker_id__mu2_Intercept",
                                   "sigma2"))
```




```
# pairs plot (too many things to view at once, so we've grouped them)
# slope effects
pairs(m.vis.wrkr.llo_mix, pars = c("b_mu1_lo_ground_truth:conditionHOPs",
                                   "b_mu1_lo_ground_truth:conditionmeans_only",
                                   "b_mu1_lo_ground_truth:conditionintervals_w_means"
))
```



- Summary

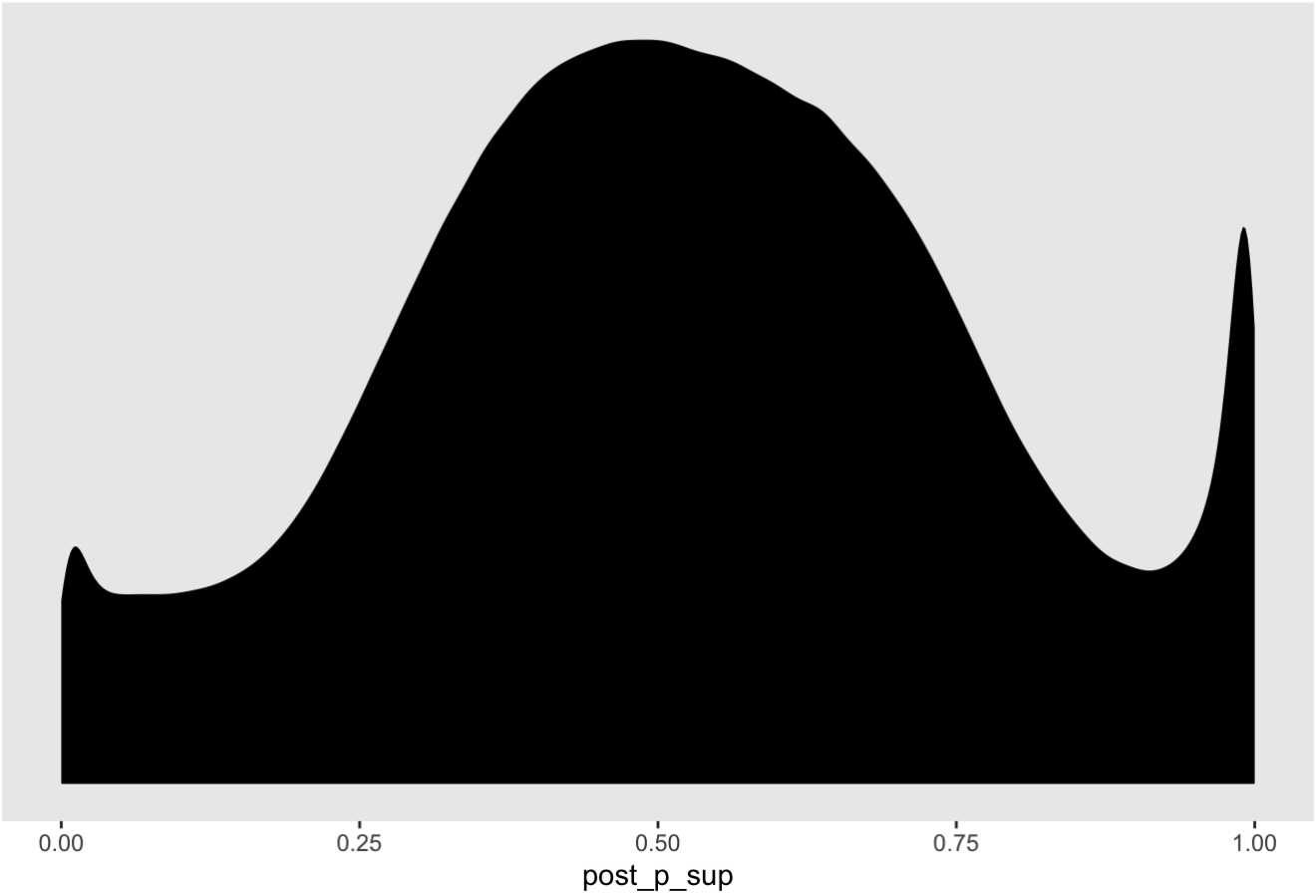
```
# model summary
print(m.vis.wrkr.llo_mix)
```

```
##  Family: mixture(gaussian, gaussian)
##   Links: mu1 = identity; sigma1 = identity; mu2 = identity; sigma2 = identity; the
ta1 = identity; theta2 = identity
## Formula: lo_p_sup ~ 1
##          mu1 ~ (1 + lo_ground_truth | worker_id) + lo_ground_truth:condition
##          mu2 ~ (1 | worker_id)
##          theta2 ~ (1 | worker_id) + 0 + condition
##    Data: model_df_llo (Number of observations: 1308)
## Samples: 2 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 2000
##
## Group-Level Effects:
## ~worker_id (Number of levels: 109)
##                                     Estimate Est.Error l-95% CI
## sd(mu1_Intercept)                       1.50      0.23     1.10
## sd(mu1_lo_ground_truth)                 0.79      0.11     0.60
## sd(mu2_Intercept)                       0.68      0.06     0.56
## sd(theta2_Intercept)                    8.56      2.85     4.76
## cor(mu1_Intercept,mu1_lo_ground_truth)  -0.15      0.17    -0.47
##                                     u-95% CI Eff.Sample Rhat
## sd(mu1_Intercept)                       1.99        665 1.00
## sd(mu1_lo_ground_truth)                 1.02        443 1.01
## sd(mu2_Intercept)                       0.80        311 1.01
## sd(theta2_Intercept)                   15.36        287 1.00
## cor(mu1_Intercept,mu1_lo_ground_truth)  0.21        391 1.00
##
## Population-Level Effects:
##                                                Estimate Est.Error l-95% CI
## mu1_Intercept                                     -0.11      0.15    -0.44
## mu2_Intercept                                      0.09      0.08    -0.07
## mu1_lo_ground_truth:conditionHOPs                   0.33      0.19    -0.04
## mu1_lo_ground_truth:conditionintervals_w_means      0.99      0.21     0.57
## mu1_lo_ground_truth:conditionmeans_only             0.19      0.23    -0.26
## theta2_conditionHOPs                                1.07      0.84    -0.54
## theta2_conditionintervals_w_means                   1.54      0.86    -0.12
## theta2_conditionmeans_only                          1.92      0.90     0.08
##                                                u-95% CI Eff.Sample Rhat
## mu1_Intercept                                      0.13        590 1.00
## mu2_Intercept                                      0.24        175 1.02
## mu1_lo_ground_truth:conditionHOPs                   0.70        518 1.00
## mu1_lo_ground_truth:conditionintervals_w_means      1.43        491 1.00
## mu1_lo_ground_truth:conditionmeans_only             0.62        746 1.00
## theta2_conditionHOPs                                2.72       1409 1.00
## theta2_conditionintervals_w_means                   3.19       1361 1.00
## theta2_conditionmeans_only                          3.64        851 1.00
##
## Family Specific Parameters:
##        Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## sigma1     1.06      0.05     0.96     1.17        333 1.00
## sigma2     0.40      0.01     0.37     0.43        536 1.01
##
## 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 out a posterior predictive distribution for probability of superiority.

```
# posterior predictive check
model_df_llo %>%
  select(lo_ground_truth, condition, worker_id) %>%
  add_predicted_draws(m.vis.wrkr.llo_mix, prediction = "lo_p_sup", seed = 1234) %>%
  mutate(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())
```

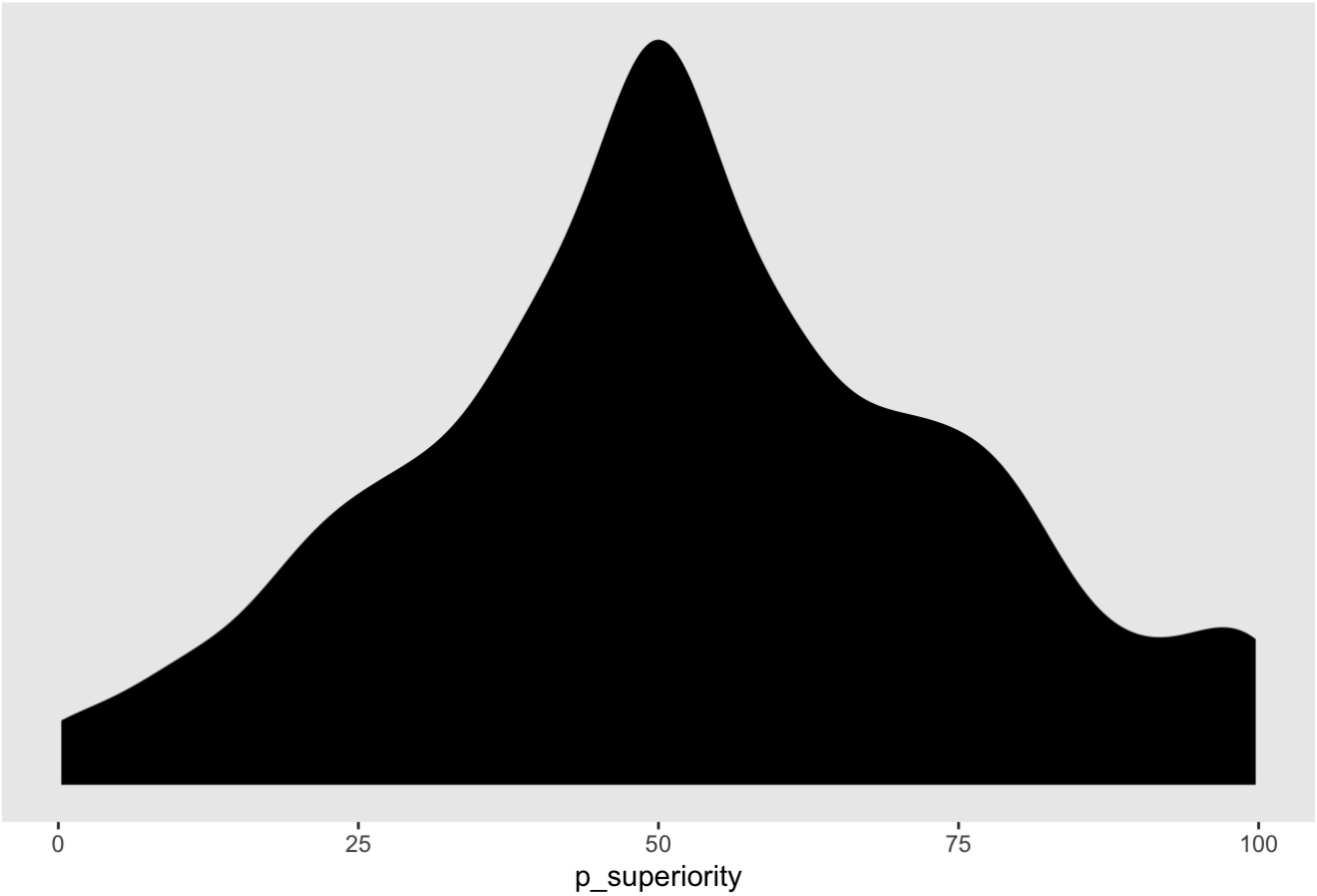
Posterior predictive distribution for probability of superiority



How does this compare to the empirical distribution of probability of superiority responses?

```
# posterior predictive check
model_df_llo %>%
  ggplot(aes(x = p_superiority)) +
  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())
```

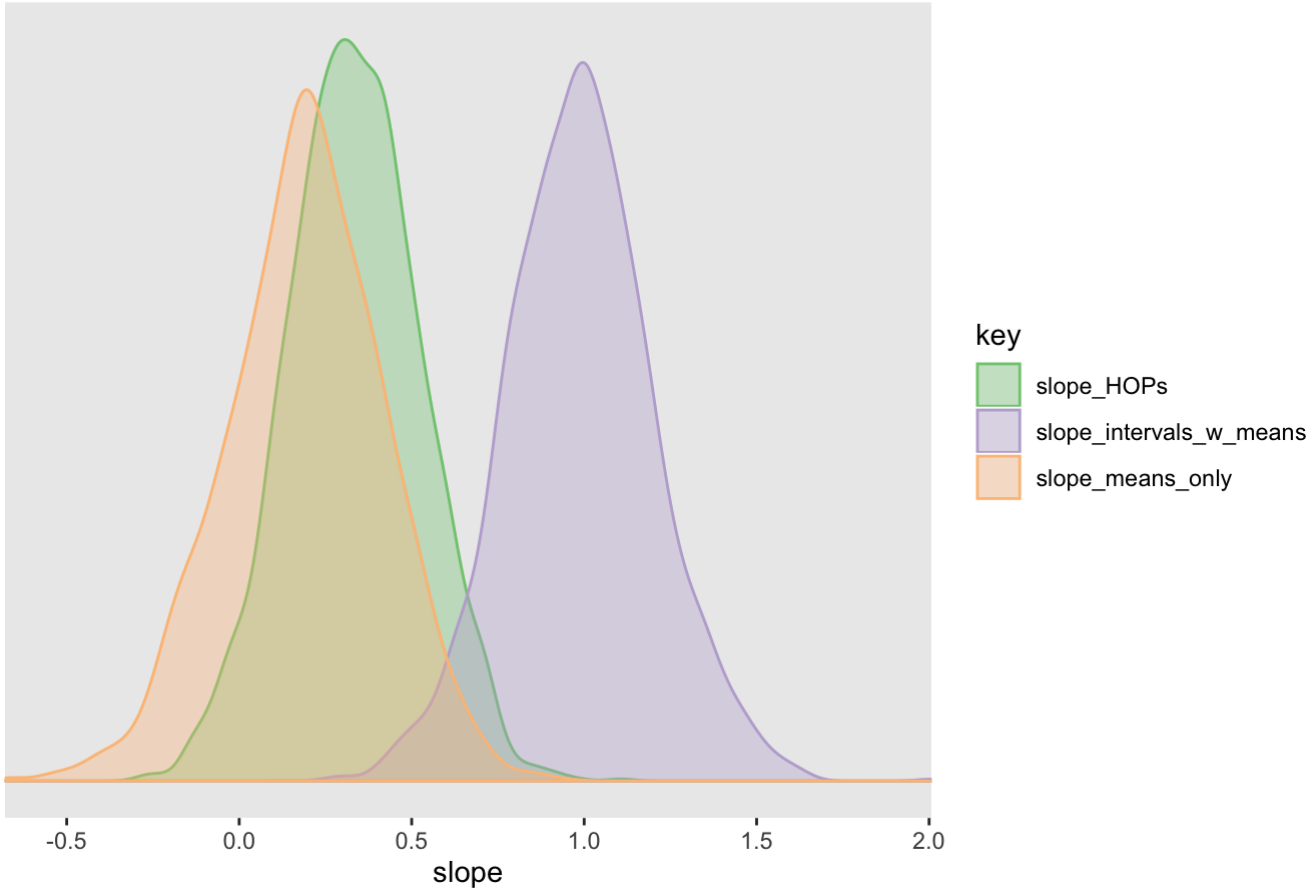
Posterior predictive distribution for probability of superiority



What does the posterior for the effect of each visualization condition look like?

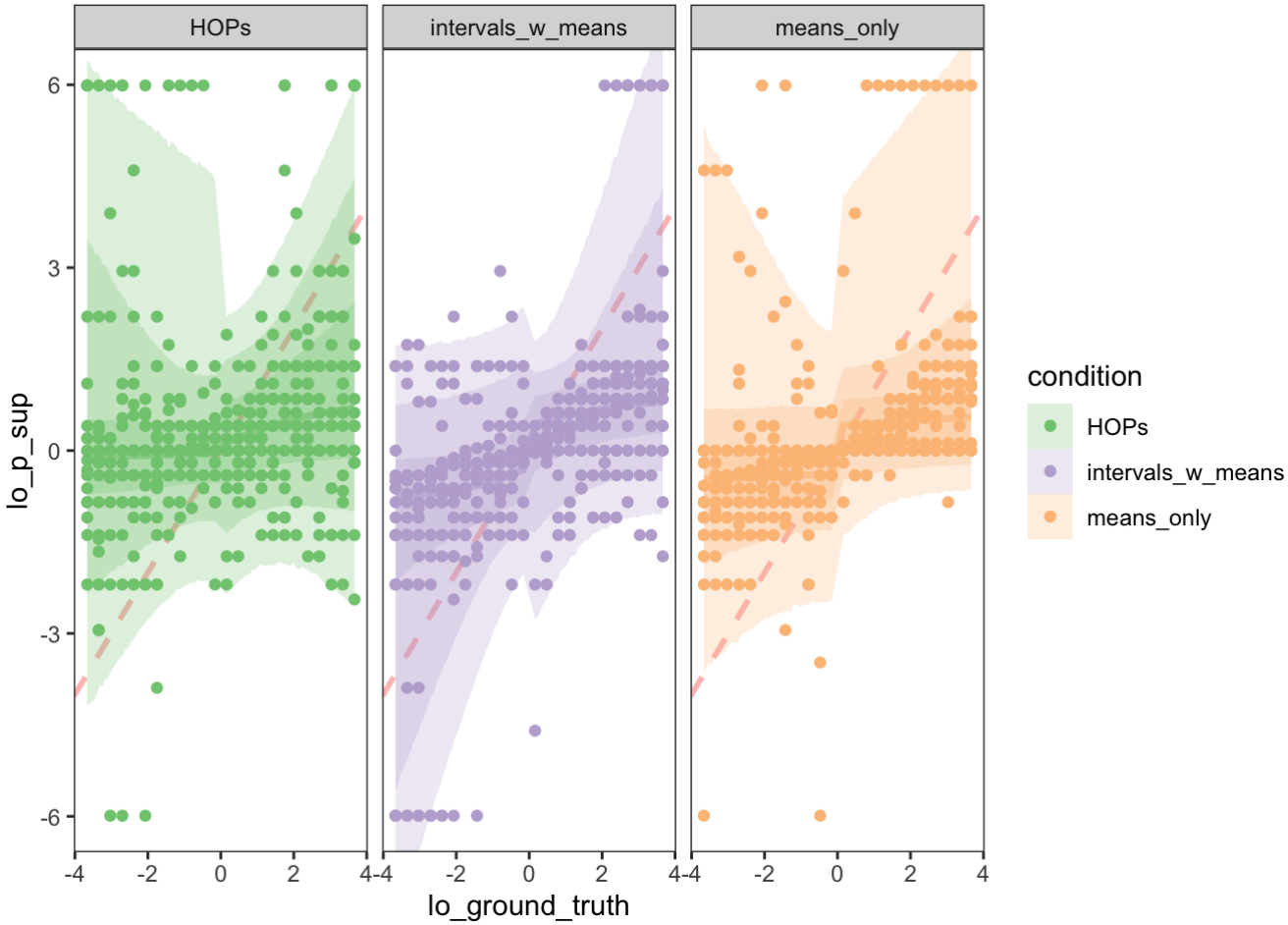
```
# use posterior samples to define distributions for the slope in each visualization c
ondition
posterior_samples(m.vis.wrkr.llo_mix) %>%
  transmute(slope_HOPs = `b_mu1_lo_ground_truth:conditionHOPs`,
            slope_intervals_w_means = `b_mu1_lo_ground_truth:conditionintervals_w_mea
ns`,
            slope_means_only = `b_mu1_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



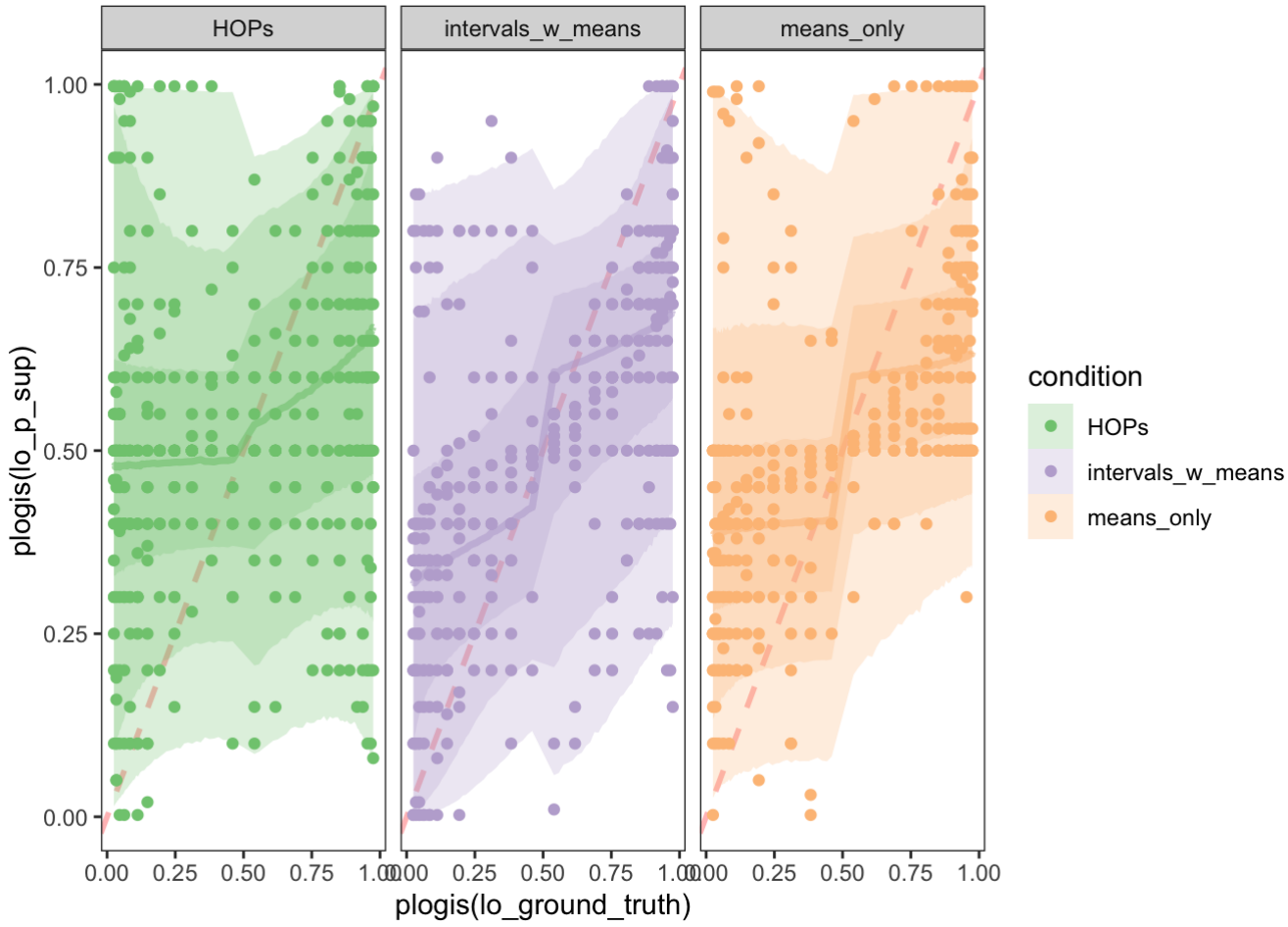
Let's take a look at some of the estimated linear models per visualization condition.

```
# this time we'll adopt functions from the tidybayes package to make plotting posteri
or predictions easier
model_df_llo %>%
  group_by(condition, worker_id) %>%
  data_grid(lo_ground_truth = seq_range(lo_ground_truth, n = 51)) %>%
  add_predicted_draws(m.vis.wrkr.llo_mix) %>%
  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_grid(. ~ condition)
```



What does this look like in probability units?

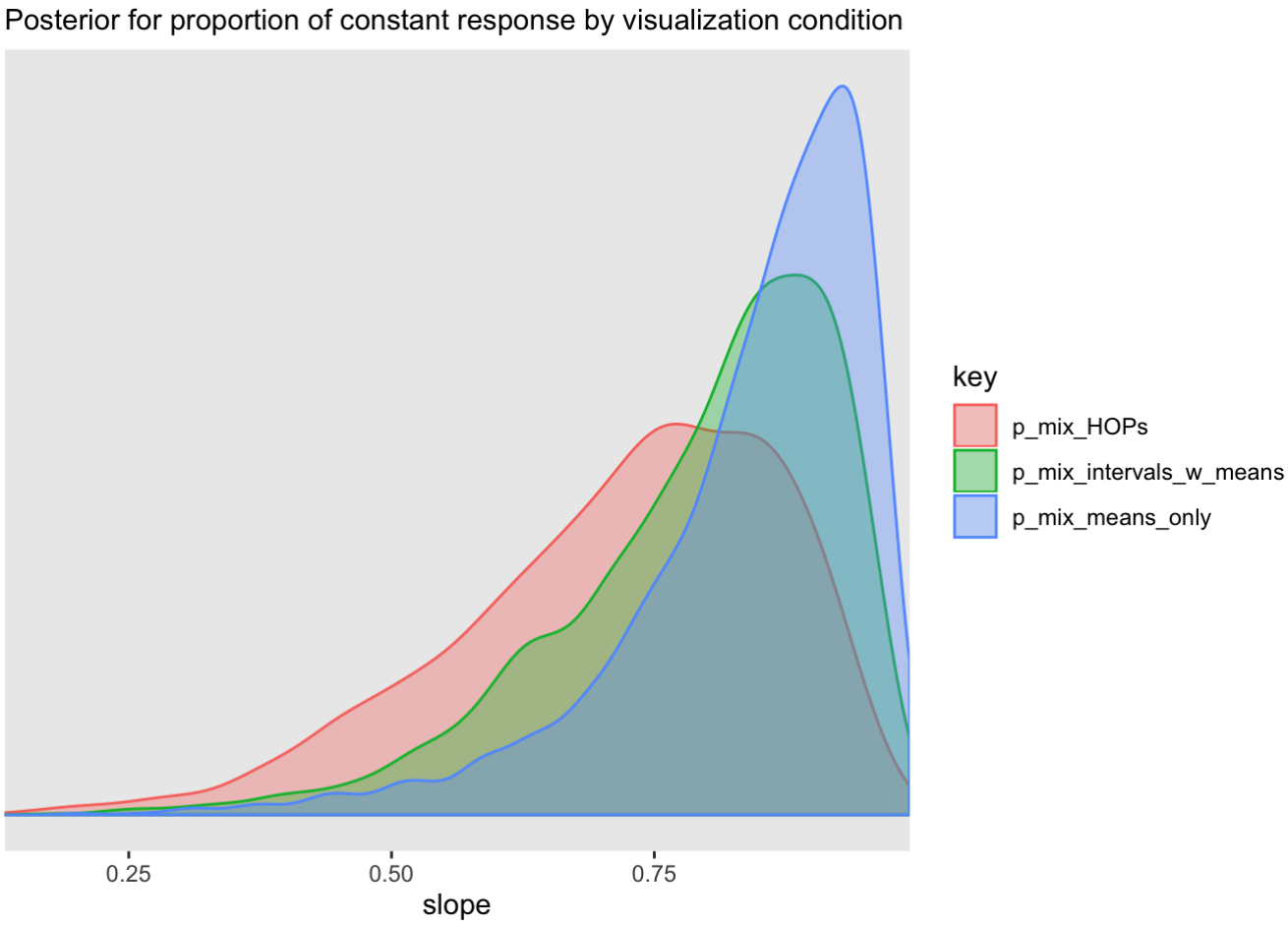
```
model_df_llo %>%
  group_by(condition, worker_id) %>%
  data_grid(lo_ground_truth = seq_range(lo_ground_truth, n = 51)) %>%
  add_predicted_draws(m.vis.wrkr.llo_mix) %>%
  ggplot(aes(x = plogis(lo_ground_truth), y = plogis(lo_p_sup), color = condition, fi
ll = condition)) +
  geom_abline(intercept = 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 = .2
5) +
  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_grid(. ~ condition)
```



The asymmetry in these predictions is due to the random effect of worker. Because each worker only sees one frame, predictions are asymmetrical about the inflection point where ground truth is 50%.

What about the mixture proportions? Let's plot the posterior for theta. Because theta is in log odds units we'll transform it into probability units.

```
# posteriors of mixture proportion
posterior_samples(m.vis.wrkr.llo_mix) %>%
  transmute(
      #p_mix_HOPs = plogis(b_theta2_Intercept),
      p_mix_HOPs = plogis(b_theta2_conditionHOPs),
      p_mix_intervals_w_means = plogis(b_theta2_conditionintervals_w_means),
      p_mix_means_only = plogis(b_theta2_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 proportion of constant response by visualization con
dition") +
  theme(panel.grid = element_blank())
```



These random constant response rates are higher than before, but otherwise the model looks good.

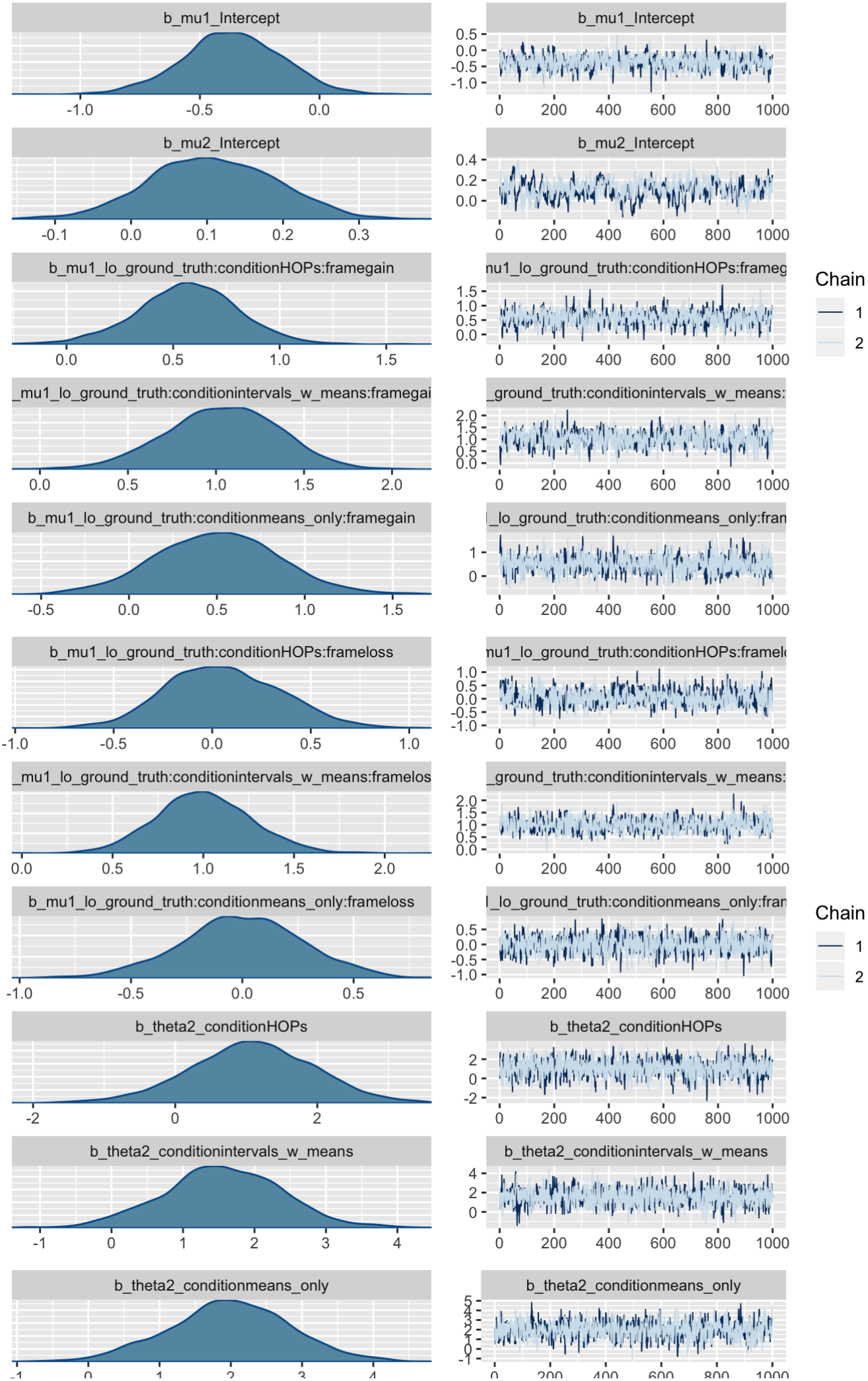
Adding a Predictor for Gain vs Loss Framing

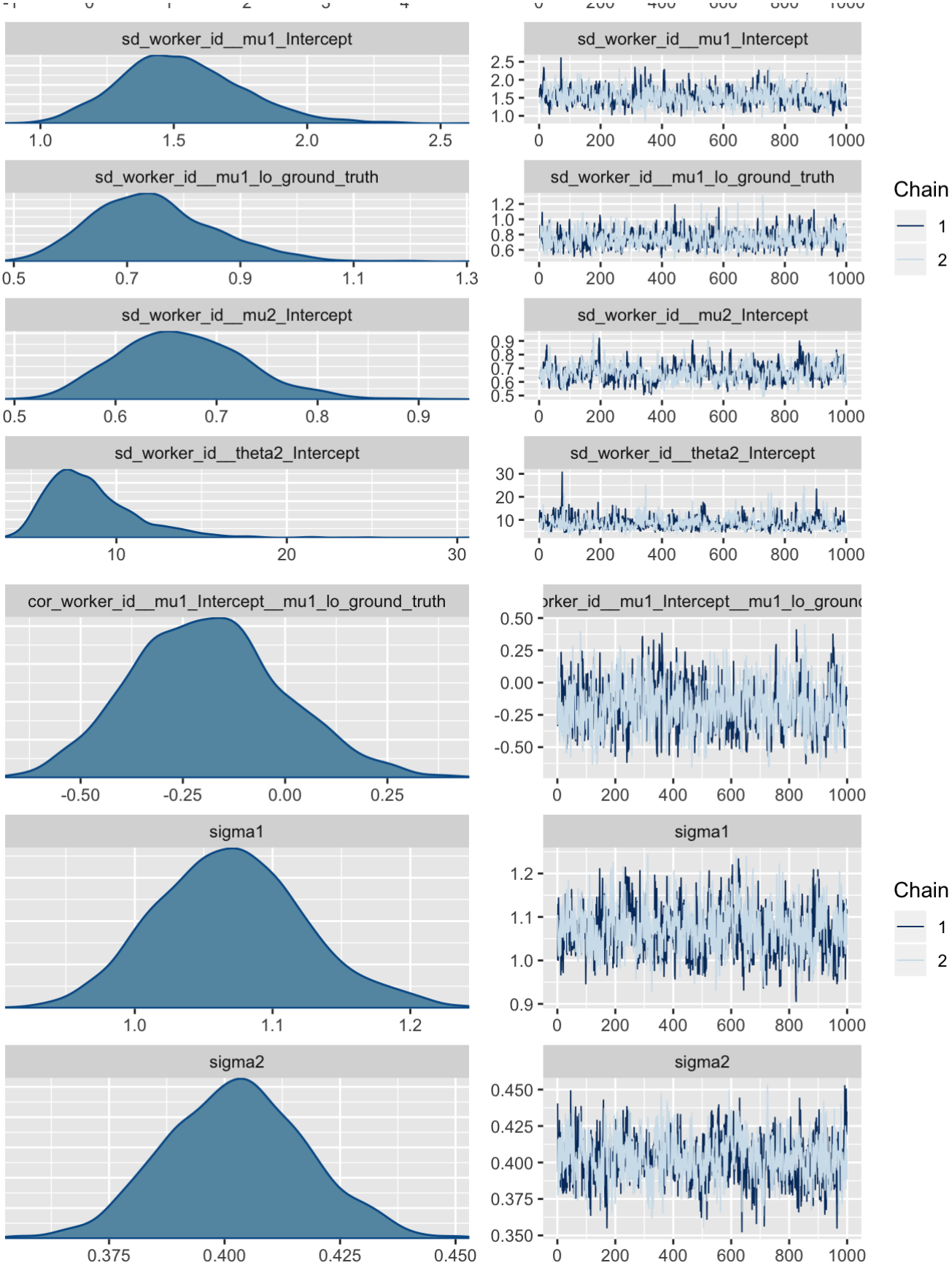
```
# define stanvars for multi_normal prior on condition effects
stanvars <- stanvar(rep(1, 3), "mu_theta2", scode = "  vector[3] mu_theta2;") +
  stanvar(diag(3), "Sigma_theta2", scode = "  matrix[3, 3] Sigma_theta2;")

# fit the model
m.vis.frame.wrkr.llo_mix <- brm(
   bf(lo_p_sup ~ 1,
    mu1 ~ (1 + lo_ground_truth|worker_id) + lo_ground_truth:condition:frame, # add an
interaction for framing
    mu2 ~ (1|worker_id), # random constant response per worker (to account for people
who always answer the same, often but not always 50%)
    theta2 ~ (1|worker_id) + 0 + condition # the proportion of responses that are con
stant
  ),
  data = model_df_llo,
  family = mixture(gaussian, gaussian, order = 'mu'),
  prior = c(
    prior(normal(0, 1), class = Intercept, dpar = mu1),
    prior(normal(0, 1), class = Intercept, dpar = mu2),
    prior("multi_normal(mu_theta2, Sigma_theta2)", class = b, dpar = theta2)
  ),
  stanvars = stanvars,
  inits = 1, chains = 2, cores = 2,
  control = list(adapt_delta = 0.999, max_treedepth=15),
  file = "model-fits/llo_mix_mdl_vis_frame_wrkr-first_block"
)
```

Check diagnostics:

- Trace plots

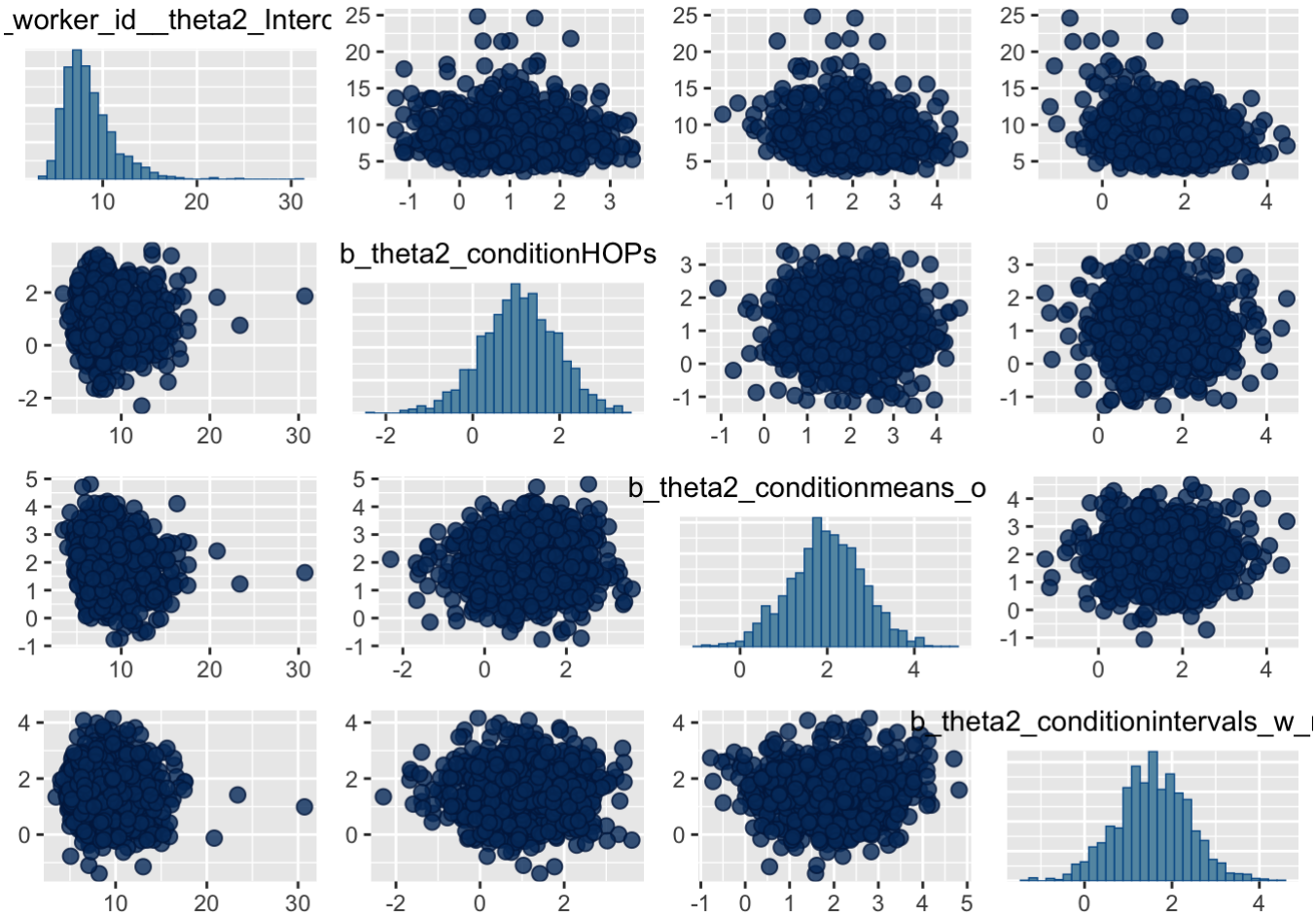
```
# trace plots
plot(m.vis.frame.wrkr.llo_mix)
```


b_mu1_Intercept
b_mu2_Intercept
b_mu1_lo_ground_truth:conditionHOPs:framegain
b_mu1_lo_ground_truth:conditionintervals_w_means:framegai
b_mu1_lo_ground_truth:conditionmeans_only:framegain
b_mu1_lo_ground_truth:conditionHOPs:frameloss
b_mu1_lo_ground_truth:conditionintervals_w_means:framelos
b_mu1_lo_ground_truth:conditionmeans_only:frameloss
b_theta2_conditionHOPs
b_theta2_conditionintervals_w_means
b_theta2_conditionmeans_only
Chain
1
2

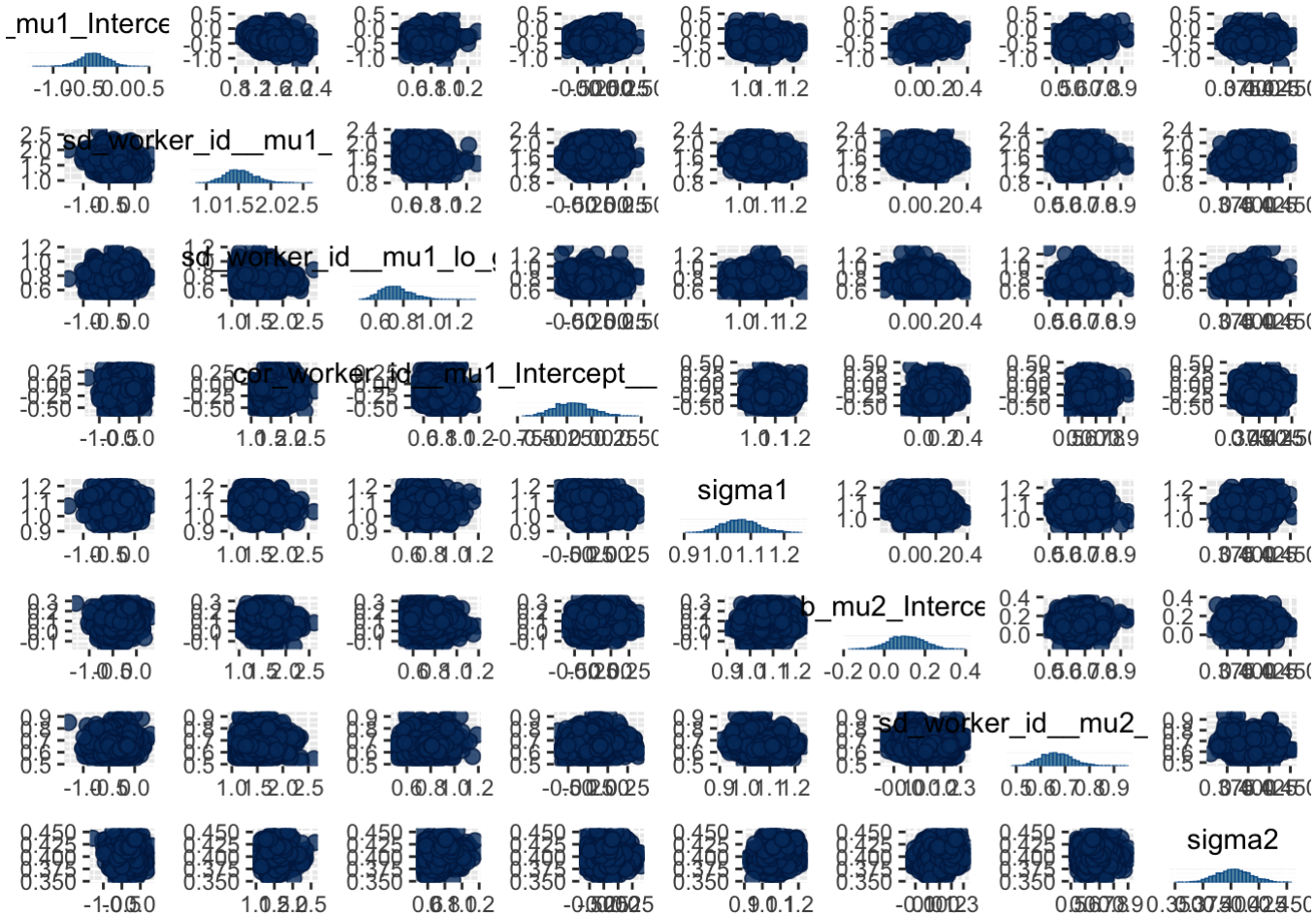


- Pairs plot

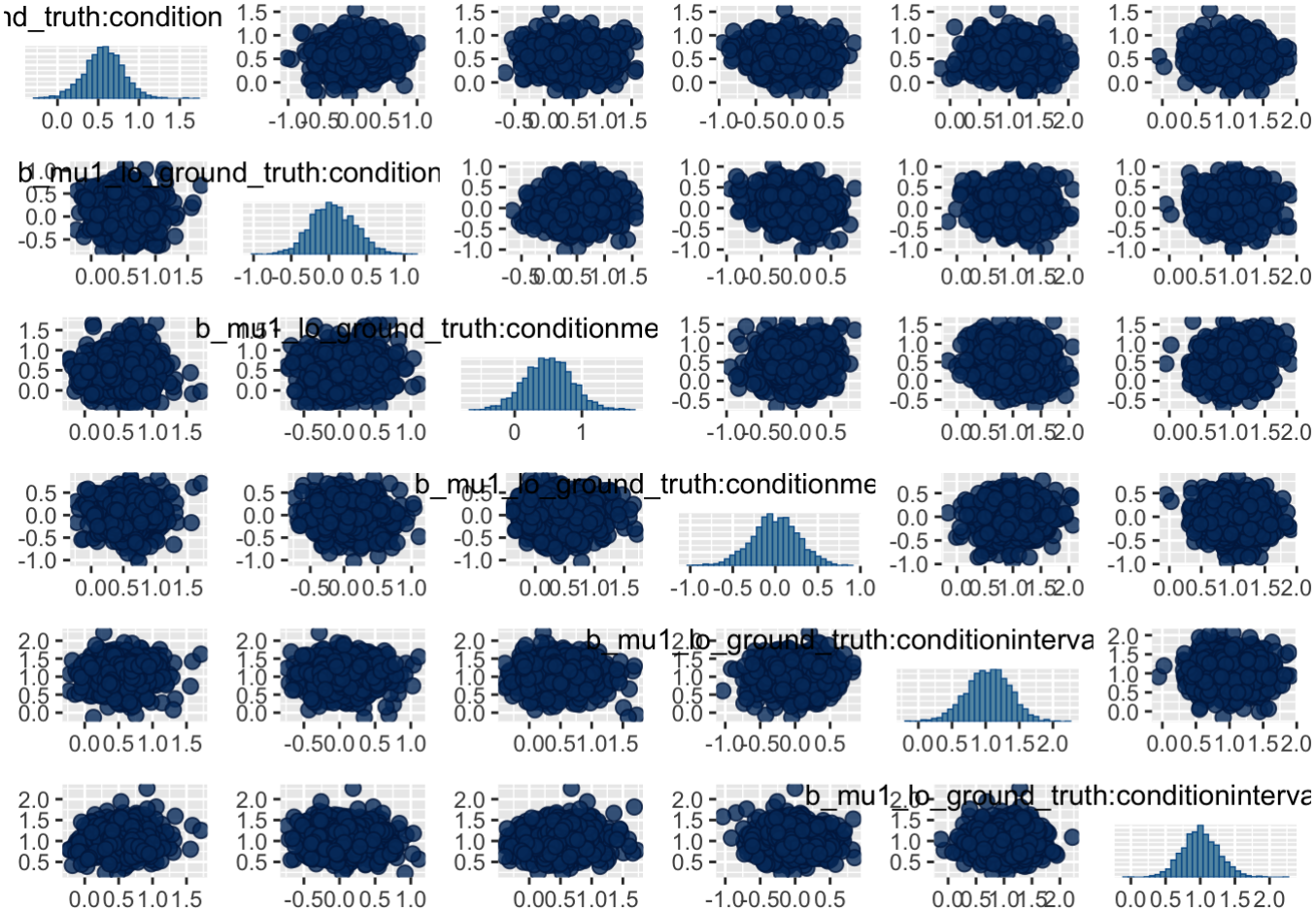
```
# pairs plot (too many things to view at once, so we've grouped them)
# mixture proportions
pairs(m.vis.frame.wrkr.llo_mix, pars = c("sd_worker_id__theta2_Intercept",
                                "b_theta2_conditionHOPs",
                                "b_theta2_conditionmeans_only",
                                "b_theta2_conditionintervals_w_means"))
```




```
# pairs plot (too many things to view at once, so we've grouped them)
# hyperparameters
pairs(m.vis.frame.wrkr.llo_mix, pars = c("b_mu1_Intercept",
                                "sd_worker_id__mu1_Intercept",
                                "sd_worker_id__mu1_lo_ground_truth",
                                "cor_worker_id__mu1_Intercept__mu1_lo_ground_truth",
                                "sigma1",
                                "b_mu2_Intercept",
                                "sd_worker_id__mu2_Intercept",
                                "sigma2"))
```




```
# pairs plot (too many things to view at once, so we've grouped them)
# slope effects
pairs(m.vis.frame.wrkr.llo_mix, pars = c("b_mu1_lo_ground_truth:conditionHOPs:framega
in",
                                         "b_mu1_lo_ground_truth:conditionHOPs:frame
loss",
                                         "b_mu1_lo_ground_truth:conditionmeans_onl
y:framegain",
                                         "b_mu1_lo_ground_truth:conditionmeans_onl
y:frameloss",
                                         "b_mu1_lo_ground_truth:conditionintervals_
w_means:framegain",
                                         "b_mu1_lo_ground_truth:conditionintervals_
w_means:frameloss"))
```



- Summary

```
# model summary
print(m.vis.frame.wrkr.llo_mix)
```

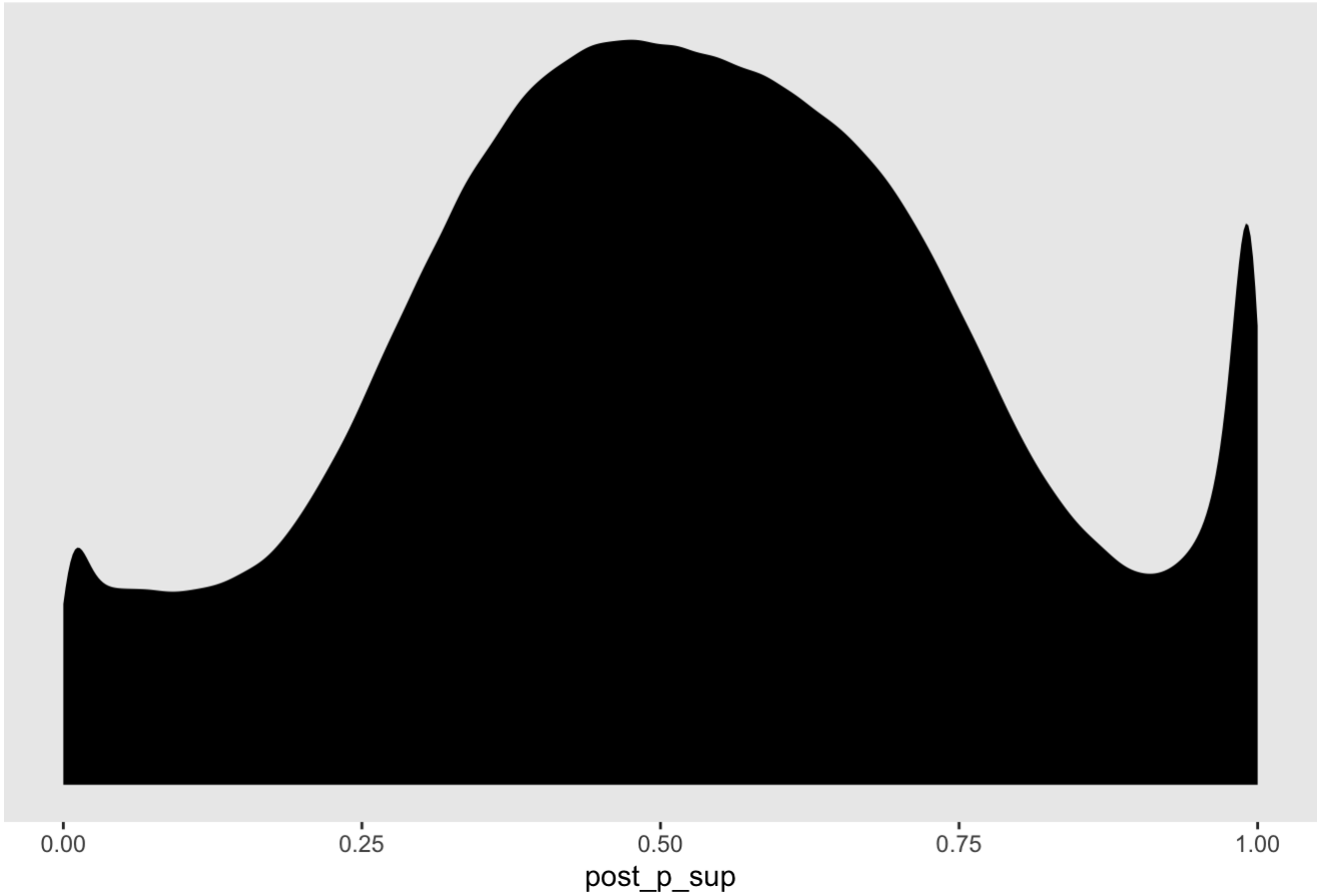
```
##  Family: mixture(gaussian, gaussian)
##   Links: mu1 = identity; sigma1 = identity; mu2 = identity; sigma2 = identity; the
ta1 = identity; theta2 = identity
## Formula: lo_p_sup ~ 1
##          mu1 ~ (1 + lo_ground_truth | worker_id) + lo_ground_truth:condition:frame
##          mu2 ~ (1 | worker_id)
##          theta2 ~ (1 | worker_id) + 0 + condition
##    Data: model_df_llo (Number of observations: 1308)
## Samples: 2 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 2000
##
## Group-Level Effects:
## ~worker_id (Number of levels: 109)
##                                     Estimate Est.Error l-95% CI
## sd(mu1_Intercept)                       1.53      0.23     1.13
## sd(mu1_lo_ground_truth)                 0.75      0.11     0.57
## sd(mu2_Intercept)                       0.67      0.06     0.56
## sd(theta2_Intercept)                    8.51      2.69     4.94
## cor(mu1_Intercept,mu1_lo_ground_truth)  -0.19      0.18    -0.53
##                                     u-95% CI Eff.Sample Rhat
## sd(mu1_Intercept)                       2.03        598 1.00
## sd(mu1_lo_ground_truth)                 0.99        639 1.00
## sd(mu2_Intercept)                       0.80        244 1.00
## sd(theta2_Intercept)                   14.86        420 1.00
## cor(mu1_Intercept,mu1_lo_ground_truth)  0.19        472 1.01
##
## Population-Level Effects:
##                                                       Estimate
## mu1_Intercept                                            -0.37
## mu2_Intercept                                             0.11
## mu1_lo_ground_truth:conditionHOPs:framegain                0.57
## mu1_lo_ground_truth:conditionintervals_w_means:framegain   1.04
## mu1_lo_ground_truth:conditionmeans_only:framegain          0.50
## mu1_lo_ground_truth:conditionHOPs:frameloss                0.06
## mu1_lo_ground_truth:conditionintervals_w_means:frameloss   1.00
## mu1_lo_ground_truth:conditionmeans_only:frameloss         -0.00
## theta2_conditionHOPs                                      1.08
## theta2_conditionintervals_w_means                         1.56
## theta2_conditionmeans_only                                1.99
##                                                       Est.Error
## mu1_Intercept                                              0.21
## mu2_Intercept                                              0.08
## mu1_lo_ground_truth:conditionHOPs:framegain                 0.24
## mu1_lo_ground_truth:conditionintervals_w_means:framegain    0.32
## mu1_lo_ground_truth:conditionmeans_only:framegain           0.35
## mu1_lo_ground_truth:conditionHOPs:frameloss                 0.29
## mu1_lo_ground_truth:conditionintervals_w_means:frameloss    0.26
## mu1_lo_ground_truth:conditionmeans_only:frameloss           0.28
## theta2_conditionHOPs                                       0.87
## theta2_conditionintervals_w_means                          0.83
## theta2_conditionmeans_only                                 0.85
##                                                       l-95% CI u-95% CI
## mu1_Intercept                                            -0.78     0.04
## mu2_Intercept                                            -0.06     0.27
## mu1_lo_ground_truth:conditionHOPs:framegain                0.08     1.04
## mu1_lo_ground_truth:conditionintervals_w_means:framegain   0.38     1.64
## mu1_lo_ground_truth:conditionmeans_only:framegain         -0.19     1.20
```

```
## mu1_lo_ground_truth:conditionHOPs:frameloss                  -0.48     0.65
## mu1_lo_ground_truth:conditionintervals_w_means:frameloss     0.50     1.53
## mu1_lo_ground_truth:conditionmeans_only:frameloss           -0.56     0.53
## theta2_conditionHOPs                                        -0.68     2.78
## theta2_conditionintervals_w_means                           -0.11     3.21
## theta2_conditionmeans_only                                   0.29     3.69
##                                                          Eff.Sample Rhat
## mu1_Intercept                                                   644 1.00
## mu2_Intercept                                                   103 1.02
## mu1_lo_ground_truth:conditionHOPs:framegain                      687 1.00
## mu1_lo_ground_truth:conditionintervals_w_means:framegain        706 1.00
## mu1_lo_ground_truth:conditionmeans_only:framegain                802 1.00
## mu1_lo_ground_truth:conditionHOPs:frameloss                      596 1.00
## mu1_lo_ground_truth:conditionintervals_w_means:frameloss        910 1.00
## mu1_lo_ground_truth:conditionmeans_only:frameloss               1111 1.00
## theta2_conditionHOPs                                           1250 1.00
## theta2_conditionintervals_w_means                              1575 1.00
## theta2_conditionmeans_only                                      871 1.00
##
## Family Specific Parameters:
##        Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## sigma1     1.07      0.05     0.97     1.19        423 1.00
## sigma2     0.40      0.02     0.37     0.43        486 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 out a posterior predictive distribution for probability of superiority.

```
# posterior predictive check
model_df_llo %>%
  select(lo_ground_truth, condition, frame, worker_id) %>%
  add_predicted_draws(m.vis.frame.wrkr.llo_mix, prediction = "lo_p_sup", seed = 1234)
%>%
  mutate(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())
```

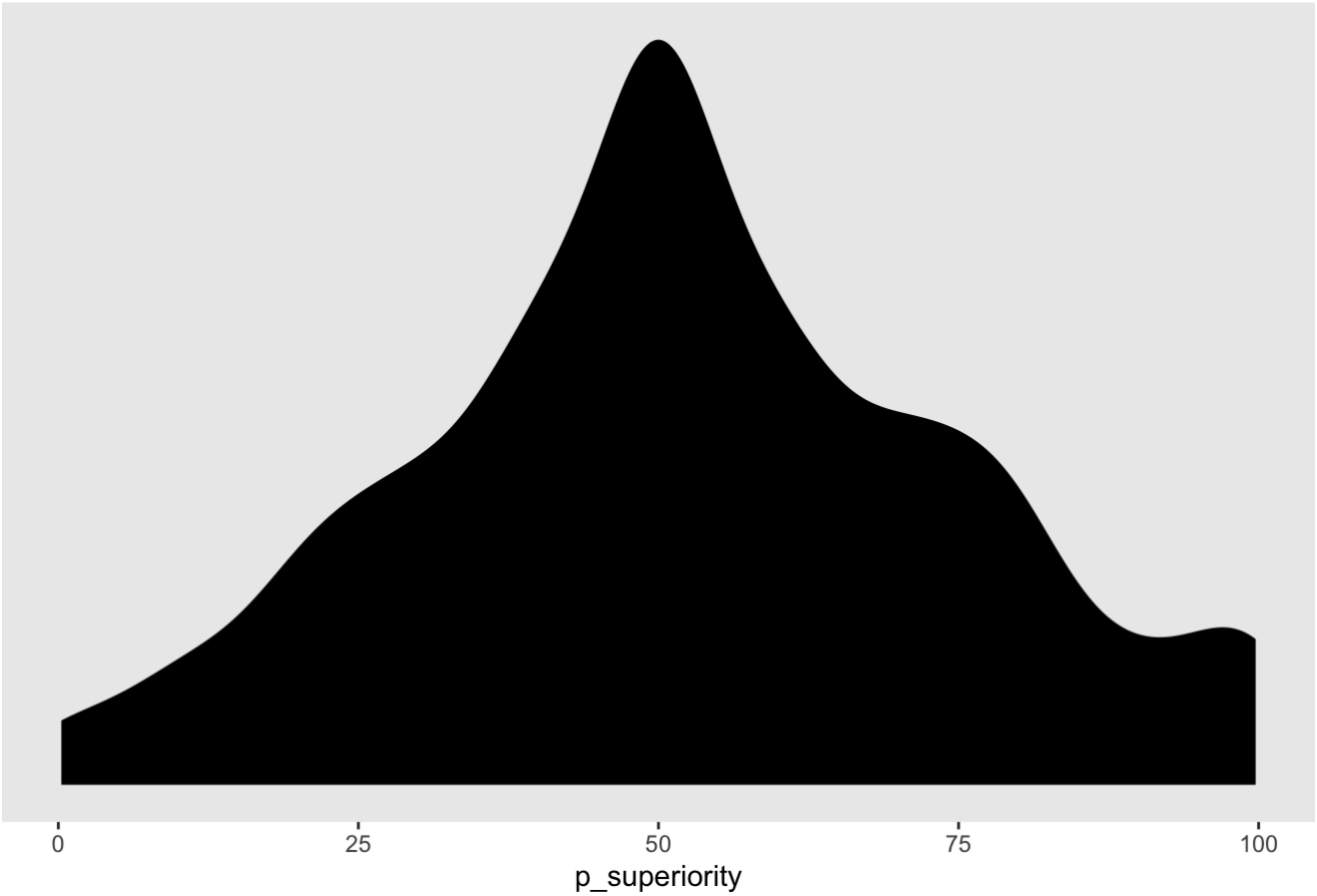
Posterior predictive distribution for probability of superiority



How does this compare to the empirical distribution of probability of superiority responses?

```
# posterior predictive check
model_df_llo %>%
  ggplot(aes(x = p_superiority)) +
  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())
```

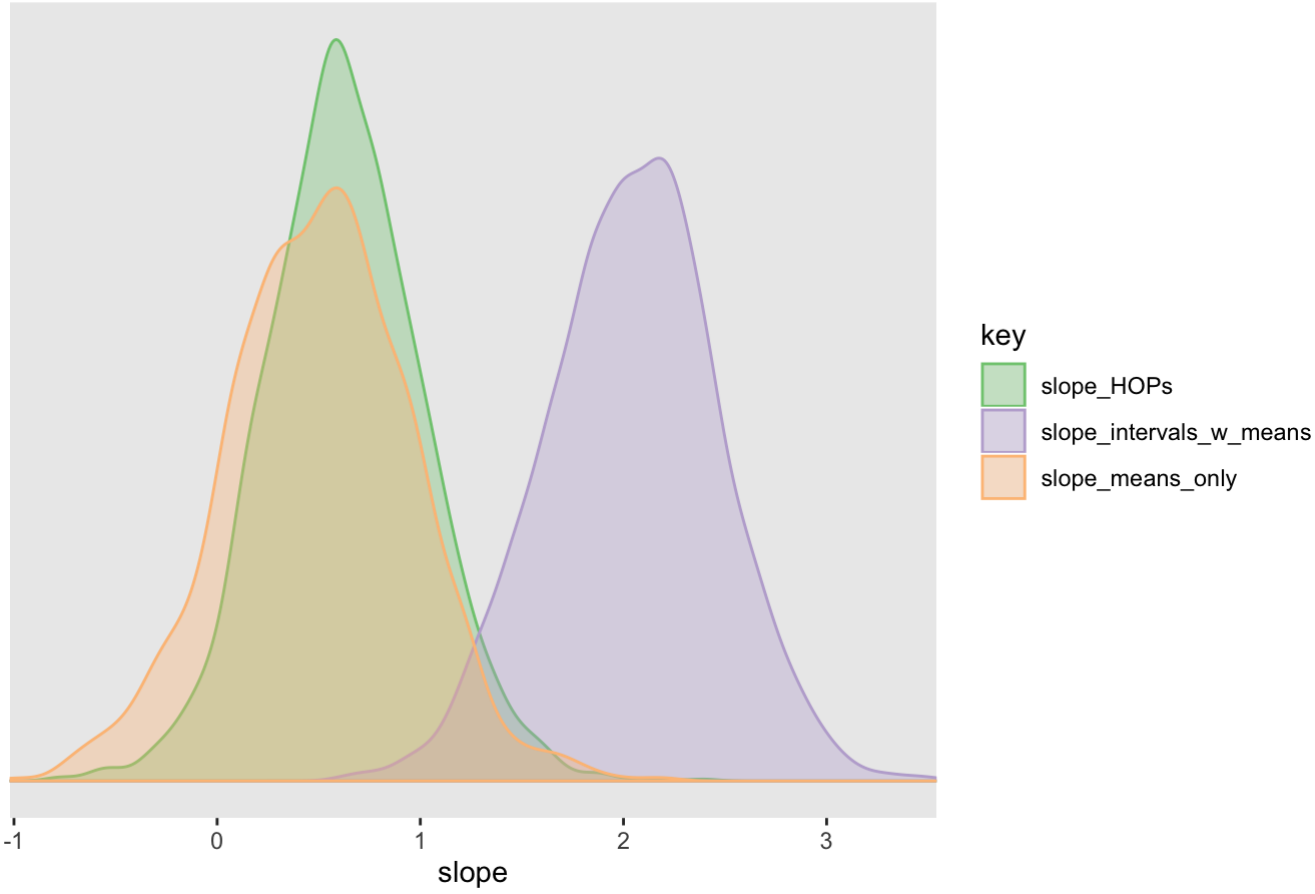
Posterior predictive distribution for probability of superiority



What does the posterior for the effect of each visualization condition look like?

```
# use posterior samples to define distributions for the slope in each visualization c
ondition
posterior_samples(m.vis.frame.wrkr.llo_mix) %>%
  transmute(slope_HOPs = `b_mu1_lo_ground_truth:conditionHOPs:framegain` + `b_mu1_lo_
ground_truth:conditionHOPs:frameloss`,
            slope_intervals_w_means = `b_mu1_lo_ground_truth:conditionintervals_w_mea
ns:framegain` + `b_mu1_lo_ground_truth:conditionintervals_w_means:frameloss`,
            slope_means_only = `b_mu1_lo_ground_truth:conditionmeans_only:framegain`
 + `b_mu1_lo_ground_truth:conditionmeans_only:frameloss`) %>%
  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())
```

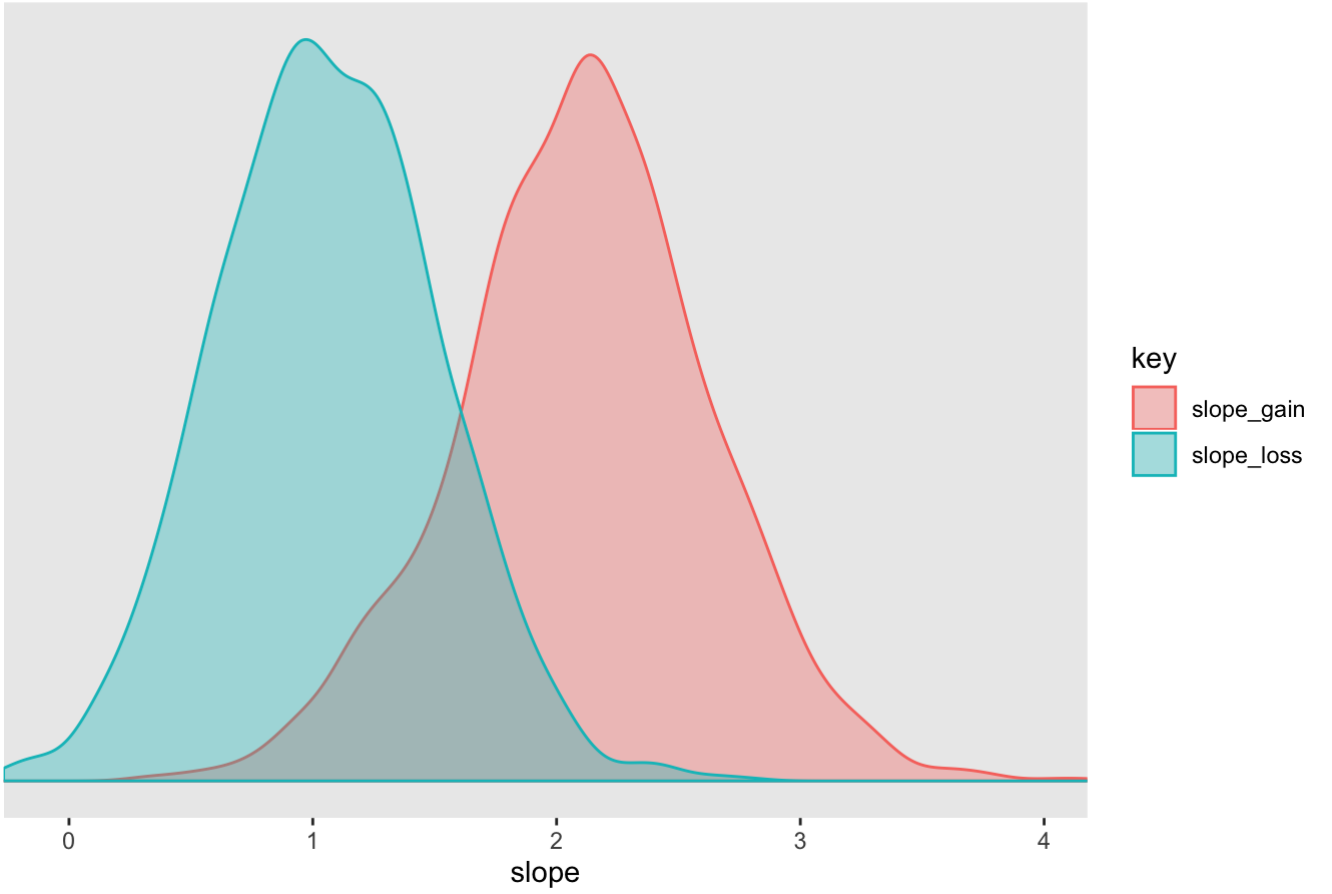
Posterior for slopes by visualization condition



What does the posterior for the effect of framing look like?

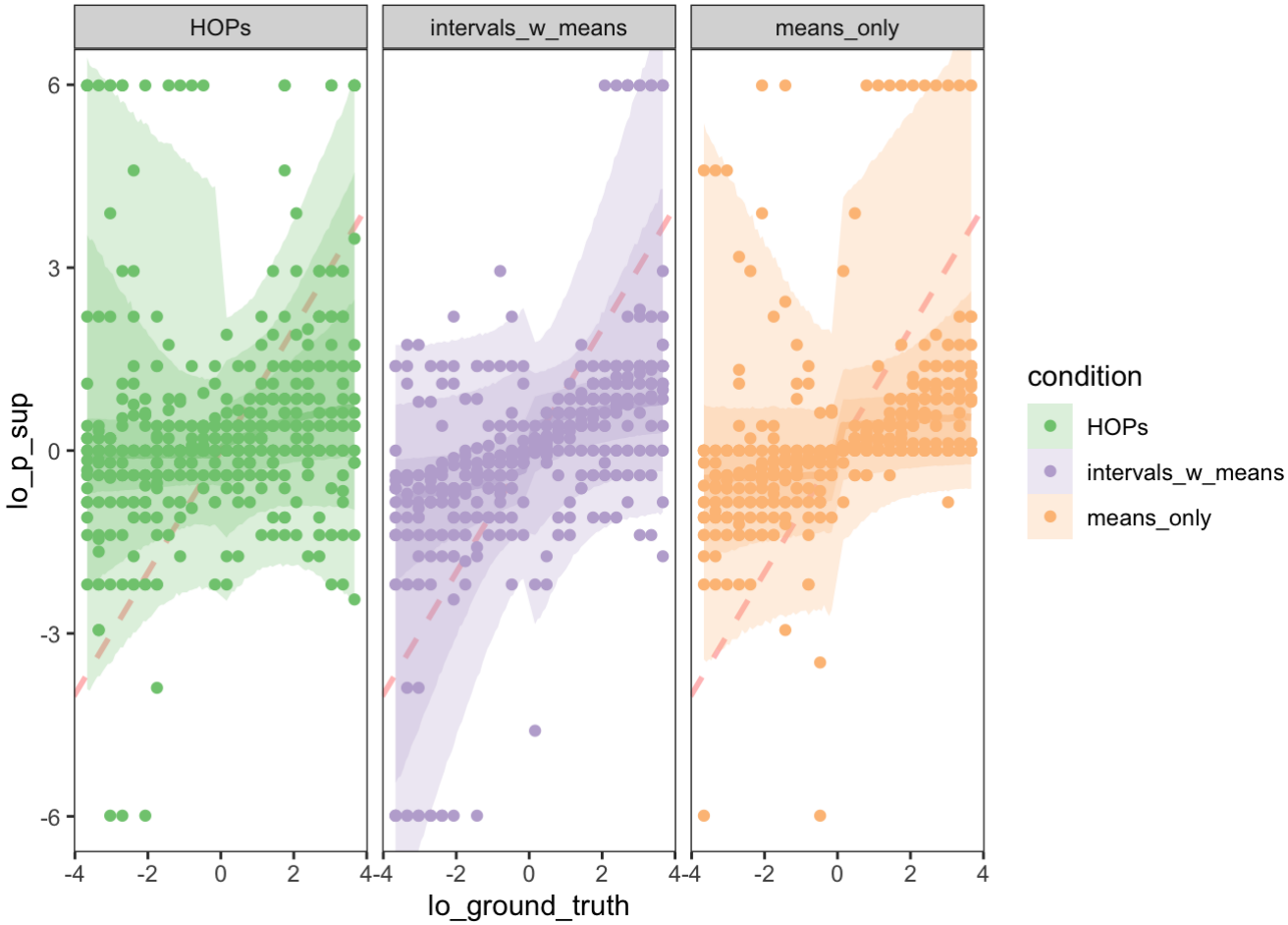
```
# use posterior samples to define distributions for the slope in the gain vs loss fra
ming conditions
posterior_samples(m.vis.frame.wrkr.llo_mix) %>%
  transmute(slope_gain = `b_mu1_lo_ground_truth:conditionHOPs:framegain` + `b_mu1_lo_
ground_truth:conditionintervals_w_means:framegain` +  `b_mu1_lo_ground_truth:conditio
nmeans_only:framegain`,
            slope_loss = `b_mu1_lo_ground_truth:conditionHOPs:frameloss` + `b_mu1_lo_
ground_truth:conditionintervals_w_means:frameloss` + `b_mu1_lo_ground_truth:condition
means_only:frameloss`) %>%
  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 gain vs loss framing") +
  theme(panel.grid = element_blank())
```

Posterior for slopes for gain vs loss framing



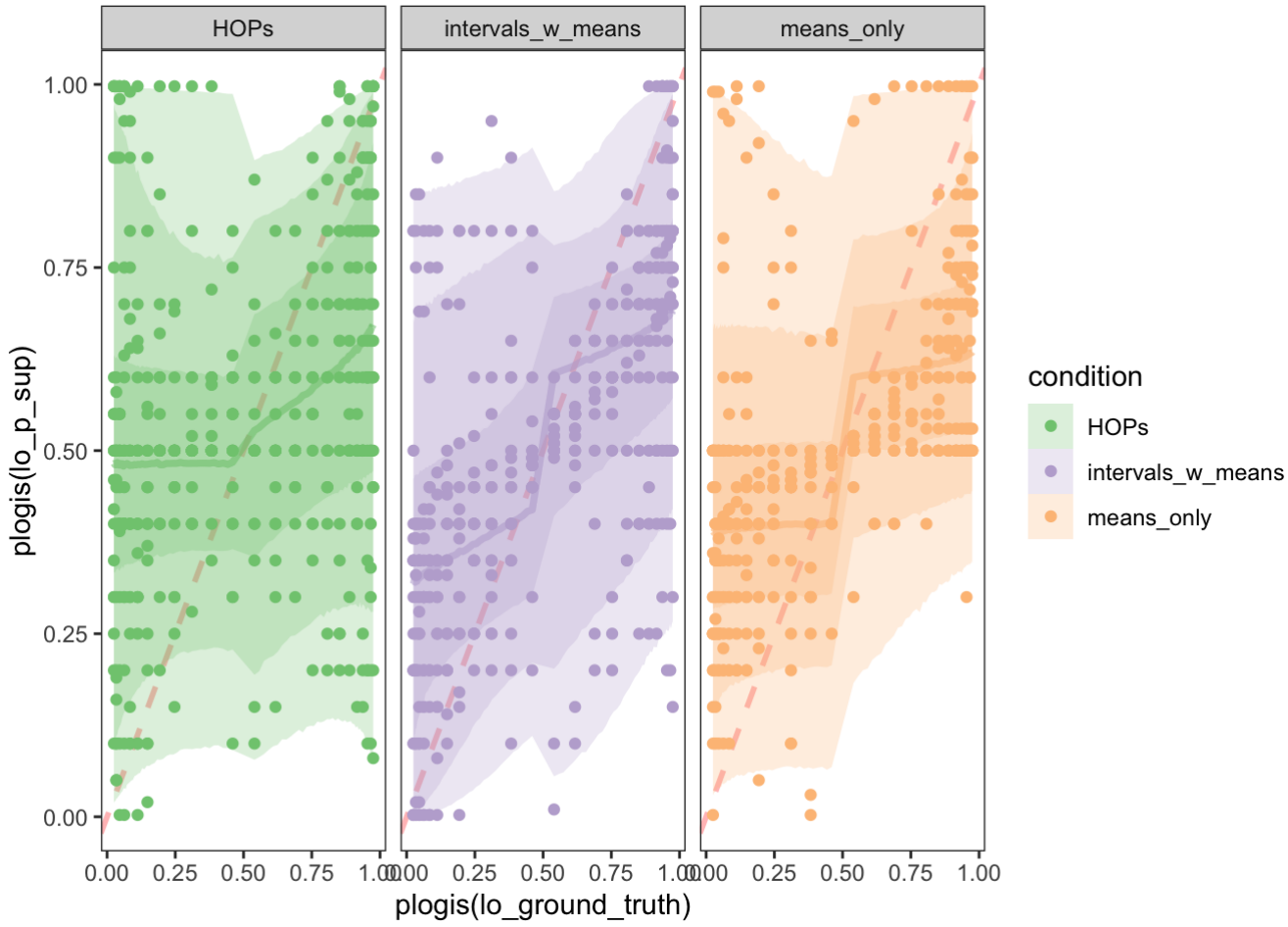
Let's take a look at some of the estimated linear models per visualization condition.

```
model_df_llo %>%
  group_by(condition, frame, worker_id) %>%
  data_grid(lo_ground_truth = seq_range(lo_ground_truth, n = 51)) %>%
  add_predicted_draws(m.vis.frame.wrkr.llo_mix) %>%
  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_grid(. ~ condition)
```



What does this look like in probability units?

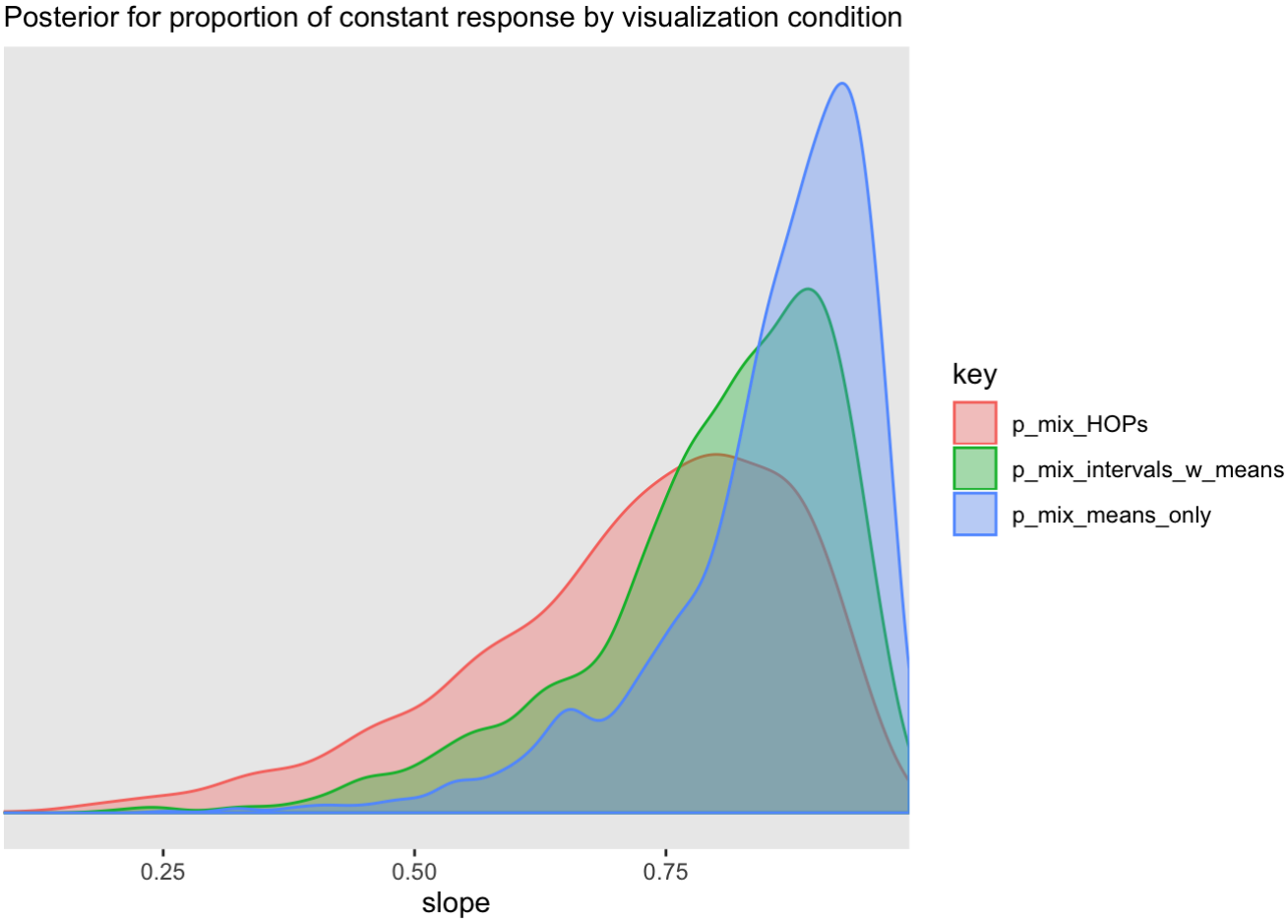
```
model_df_llo %>%
  group_by(condition, frame, worker_id) %>%
  data_grid(lo_ground_truth = seq_range(lo_ground_truth, n = 51)) %>%
  add_predicted_draws(m.vis.frame.wrkr.llo_mix) %>%
  ggplot(aes(x = plogis(lo_ground_truth), y = plogis(lo_p_sup), color = condition, fi
ll = condition)) +
  geom_abline(intercept = 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 = .2
5) +
  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_grid(. ~ condition)
```



Again, we see more pronouced asymmetries in this model, especially for HOPs.

What about the mixture proportions? Let's plot the posterior for theta. Because theta is in log odds units we'll transform it into probability units.

```
# posteriors of mixture proportion
posterior_samples(m.vis.frame.wrkr.llo_mix) %>%
  transmute(
      #p_mix_HOPs = plogis(b_theta2_Intercept),
      p_mix_HOPs = plogis(b_theta2_conditionHOPs),
      p_mix_intervals_w_means = plogis(b_theta2_conditionintervals_w_means),
      p_mix_means_only = plogis(b_theta2_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 proportion of constant response by visualization con
dition") +
  theme(panel.grid = element_blank())
```



Again, these constant response rates are higher than when we include both blocks of trials.

Model Comparison

As before, let's use WAIC to check whether adding parameters to account for framing is worth it insofar as the added parameters contribute more to predictive validity than they contribute to overfitting. Recall that lower values of WAIC indicate a better fitting model.

```
waic(m.vis.wrkr.llo_mix, m.vis.frame.wrkr.llo_mix)
```

```
##                                                 WAIC    SE
## m.vis.wrkr.llo_mix                           2731.72 89.85
## m.vis.frame.wrkr.llo_mix                     2735.28 88.63
## m.vis.wrkr.llo_mix - m.vis.frame.wrkr.llo_mix   -3.56  3.62
```

Again, the fact that WAIC values for the two models are approximately equal suggests that we don't get much improvement in model fit from adding the parameters for gain vs loss framing to this model. Maybe there is a marginal improvement with framing parameters, but it is within a standard error.