Pilot 5

Alex Kale

9/26/2019

Pilot 5

In pilot five, we reintroduce a manipulation of the level of uncertainty in the distributions we show to subjects. We also change our approach to sampling the ground truth probability of superiority so that we can place the utility optimal decision threshold where we want to, instead of having it be constrained by our ground truth sampling.

Betting Task: Scenario and Payoff Scheme

Users play a fantasy sports game where they win awards depending on how many points their team scores. They are presented with charts comparing how many points their team is predicted to score with or without adding a new player to their team.

Users are told they will *win an award* worth $X if their team *scores more than 100 points* (an arbitrary number). They can improve their chances of getting that award if they pay $C for a new player The optimal decision rule maximizes expected gains.

$$X * p(award| \sim player) < X * p(award|player) - C$$

If we assume a constant ratio between the value of the award and the cost of the new player $K = \frac{X}{C}$, the decision rule can be expressed as in terms of the difference in probability of winning the award with and without a new player.

$$p(award|player) - p(award| \sim player) > \frac{1}{K}$$

The more the new player increases their chances of winning the award, the more evidence there is in favor of intervening by paying for the new player.

Each trial, users will receive feedback based on their decision and a simulated outcome regarding the award in question. We will tally the dollar value of awards won minus the cost of new players across trials to determine the user's fantasy sports account balance. Users will receive a bonus through MTurk that is porportional to the value of their account at the end of the HIT.

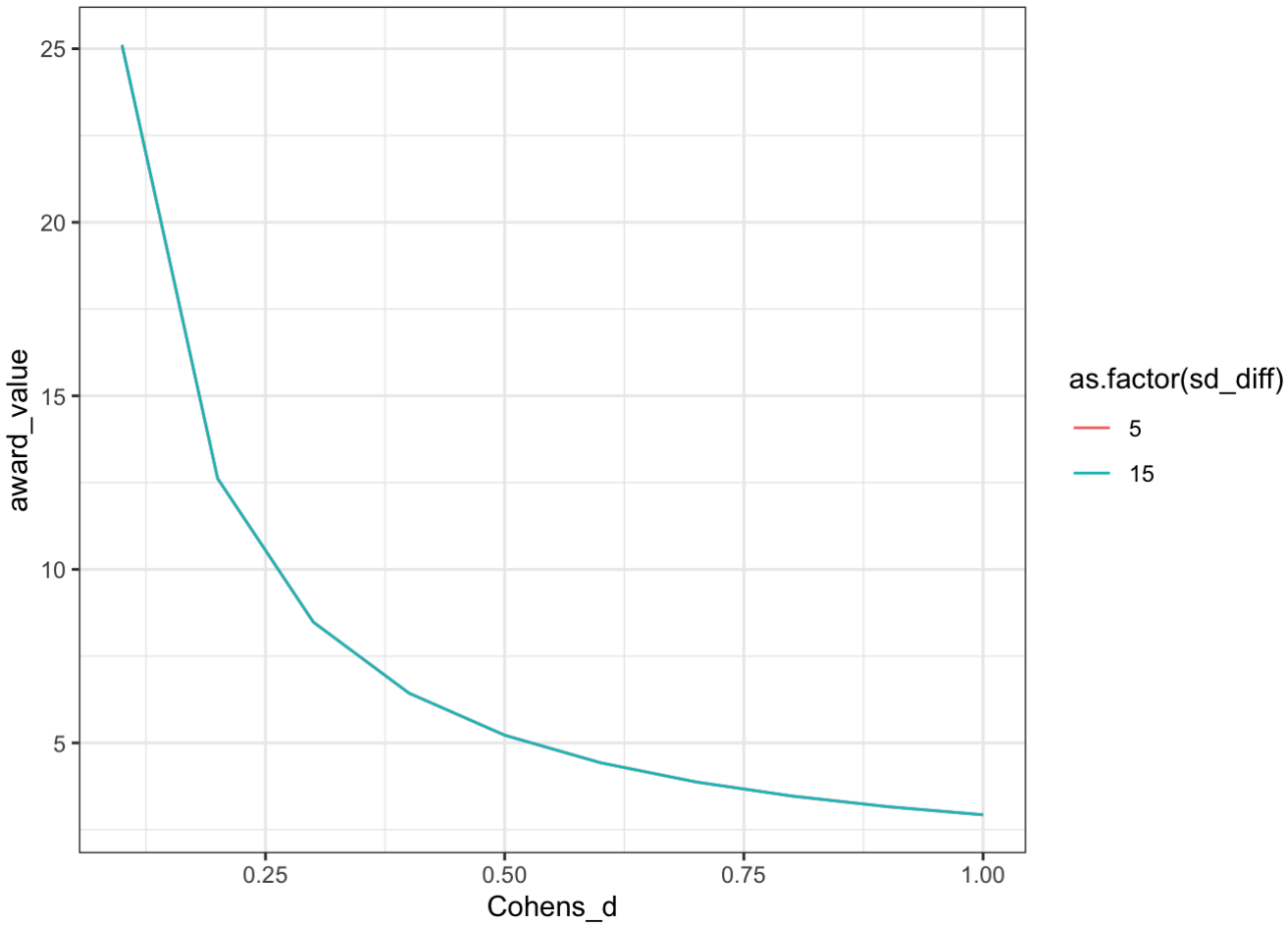
Visualization Conditions

Users will be shown predicted distributions of how many points their team will score with vs without the new player. Uncertainty about the predicted number of points scored will be visualized as *densities, quantile dotplots, intervals, HOPs*. Each user will complete a block of trials with the uncertainty encoding alone and then a second set of trials with extrinsic marks added to encode the mean. These conditions span a continuum of how much the make the mean available to users, from emphasizing the mean to only encoding it implicitly.

Data Conditions

In this study, we will select the effect size that we want for the decision threshold in terms of Cohen's d (standardized mean differences). We will then convert this value to the award value (K in the formulas above) which properly incentivize the task. The chart below shows that the award value is a function of the effect size threshold which is invariant across levels of uncertainty in the distributions we will visualize. This invariance is critical to the effective manipulation of the level of uncertainty without messing of the incentives of the task.

```
df <- data.frame(
    "Cohens_d" = rep(seq(0.1, 1, length.out = 10), 2),
    "baseline" = 0.5,
    "sd_diff" = sort(rep(c(5, 15), 10))
  )

df <- df %>%
  mutate(
    award_value = 1 / (pnorm(sd_diff * (Cohens_d / sqrt(2)) / sqrt(sd_diff ^ 2 / 2) - qn
orm(baseline)) - baseline)
  )

df %>%
  ggplot(aes(x = Cohens_d, y = award_value, color = as.factor(sd_diff))) +
  geom_line() +
  theme_bw()
```



In pilot 5, we change our approach to manipulating the ground truth probability of superiority, i.e., the probability of the team scoring or giving up more points with vs without the new player. We will sample linear intervals in logodds units to give perceptually uniform steps in probability of superiority as we did before. However, now we

will use this sampling strategy on both sides of the chose decision threshold. Note that we only sample p_superiority values of 0.5 or greater, where the new player is expected to *improve* the team's preformance.

We select decision thresholds at Cohen's d of 0.9 and 0.32 which were found to be large and medium effect sizes in preregistered psychology studies. For reference, 0.9 is about the average effect size in experimental psychology, and 0.32 is about average for preregistered between subjects studies across subfields of psychology. We choose these values because they are empirically estabilished, they are reasonable thresholds for our experiment (not too high or low as to make the task too easy). The decision threshold will be a within subjects manipulation, so we are sampling two separate sets of ground truth values here.

```
# target decision-making threshold(s)
Cohens_d <- c(0.32, 0.9)

# levels of ground truth we want
n_trials <- 8

# linear sampling of log odds for a span of ground truth probability of superiority betw
een 0.55 and 0.95
logodds <- c(
    # sampling around the low effect size threshold
    seq(log(0.55 / (1 - 0.55)), log(pnorm((Cohens_d[1] / sqrt(2)) - 0.05) / (1 - pnorm
((Cohens_d[1] / sqrt(2)) - 0.05))), length.out = n_trials / 2),
    seq(log(pnorm((Cohens_d[1] / sqrt(2)) + 0.05) / (1 - pnorm((Cohens_d[1] / sqrt(2)) +
0.05))), log(0.95 / (1 - 0.95)), length.out = n_trials / 2),
    # sampling around the high effect size threshold
    seq(log(0.55 / (1 - 0.55)), log(pnorm((Cohens_d[2] / sqrt(2)) - 0.05) / (1 - pnorm
((Cohens_d[2] / sqrt(2)) - 0.05))), length.out = n_trials / 2),
    seq(log(pnorm((Cohens_d[2] / sqrt(2)) + 0.05) / (1 - pnorm((Cohens_d[2] / sqrt(2)) +
0.05))), log(0.95 / (1 - 0.95)), length.out = n_trials / 2)
  )

# convert from log odds to probability of superiority
p_superiority <- 1 / (1 + exp(-logodds))

# initialize data conditions dataframe
conds_df <- data.frame(
    "p_superiority" = p_superiority,
    "Cohens_d_threshold" = sort(rep(Cohens_d, length(p_superiority) / 2))
  )

head(conds_df)
```

```
##   p_superiority Cohens_d_threshold
## 1     0.5500000               0.32
## 2     0.5566747               0.32
## 3     0.5633291               0.32
## 4     0.5699607               0.32
## 5     0.6088313               0.32
## 6     0.7818388               0.32
```

We set the baseline probability of winning/keeping the award without the new player to a constant value of 0.5. The team is as likely as a coin flip to win or keep the award without the new player. This represents the scenario where there is the maximum uncertainty about outcomes without intervention. We also add a constant called point threshold which how many points users need to score in order to win an award on a given trial.

```
# baseline probability of winning/keeping an award without the new player
conds_df <- conds_df %>%
  mutate(
    baseline = 0.5,
    point_threshold = 100
  )

head(conds_df)
```

```
##   p_superiority Cohens_d_threshold baseline point_threshold
## 1     0.5500000               0.32      0.5             100
## 2     0.5566747               0.32      0.5             100
## 3     0.5633291               0.32      0.5             100
## 4     0.5699607               0.32      0.5             100
## 5     0.6088313               0.32      0.5             100
## 6     0.7818388               0.32      0.5             100
```

We also want to create stimuli for the practice trials. To make these trials easy, we choose probability of superiority values near 1. This way it should be obvious that the new player is worth the cost.

```
# create df containting rows for practice trials
prac_df <- data.frame(
  "p_superiority" = 0.999,
  Cohens_d_threshold = Cohens_d,
  "baseline" = .5,
  "point_threshold" = 100)

# append to conditions dataframe
conds_df <- rbind(conds_df, prac_df)

head(prac_df)
```

```
##   p_superiority Cohens_d_threshold baseline point_threshold
## 1         0.999               0.32      0.5             100
## 2         0.999               0.90      0.5             100
```

Since judging probability of superiority might be difficult for participants, we are including a mock task to help them understand what we are asking. We ask them judge a case where probability of superiority is 50%.

```
# create df containting rows for mock trial in each condition
mock_df <- data.frame(
  "p_superiority" = 0.5,
  Cohens_d_threshold = Cohens_d,
  "baseline" = .5,
  "point_threshold" = 100)

# add to conds_df
conds_df <- rbind(conds_df, mock_df)

print(mock_df)
```

```
##   p_superiority Cohens_d_threshold baseline point_threshold
## 1           0.5               0.32      0.5             100
## 2           0.5               0.90      0.5             100
```

We manipulate the uncertainty in predictions by using two different values for the the standard deviation of the distribution of the difference in points between the team with and without the new player (sd_diff). We set the standard deviation of the difference in points to 5 and 15 in different trials so that each level of ground truth probability of superiority is presented at each level of uncertainty.

```
# double the length of the dataframe to add two levels of uncertainty
conds_df <- map_df(seq_len(2), ~conds_df)
# add two levels of standard deviation of the difference in the number of points for the team with vs without the new player
conds_df$sd_diff <- sort(rep(c(5, 15), length(conds_df$p_superiority) / 2)) # std(with - without)

head(conds_df)
```

```
##   p_superiority Cohens_d_threshold baseline point_threshold sd_diff
## 1     0.5500000               0.32      0.5             100       5
## 2     0.5566747               0.32      0.5             100       5
## 3     0.5633291               0.32      0.5             100       5
## 4     0.5699607               0.32      0.5             100       5
## 5     0.6088313               0.32      0.5             100       5
## 6     0.7818388               0.32      0.5             100       5
```

Since we only want practice trials and attention checks at low uncertainty where p_superiority is 0.5 and at high uncertainty where p_superiority is 0.999, we need to drop some rows.

```
# drop all but two attention check trials
drop_df <- tibble(
  p_superiority = c(0.5, 0.999),
  sd_diff = c(15, 5)
)
conds_df <- conds_df %>% anti_join(drop_df, by = c("p_superiority", "sd_diff"))
```

We derive the mean difference in the number of points scored by the team with minus without the new player (mean_diff) from sd_diff and p_superiority.

```
# add columns for the mean and standard deviation of the difference in the number of poi
nts for the team with vs without the new player
conds_df <- conds_df %>%
  mutate(mean_diff = sd_diff * qnorm(p_superiority)) # mean(with - without)

head(conds_df)
```

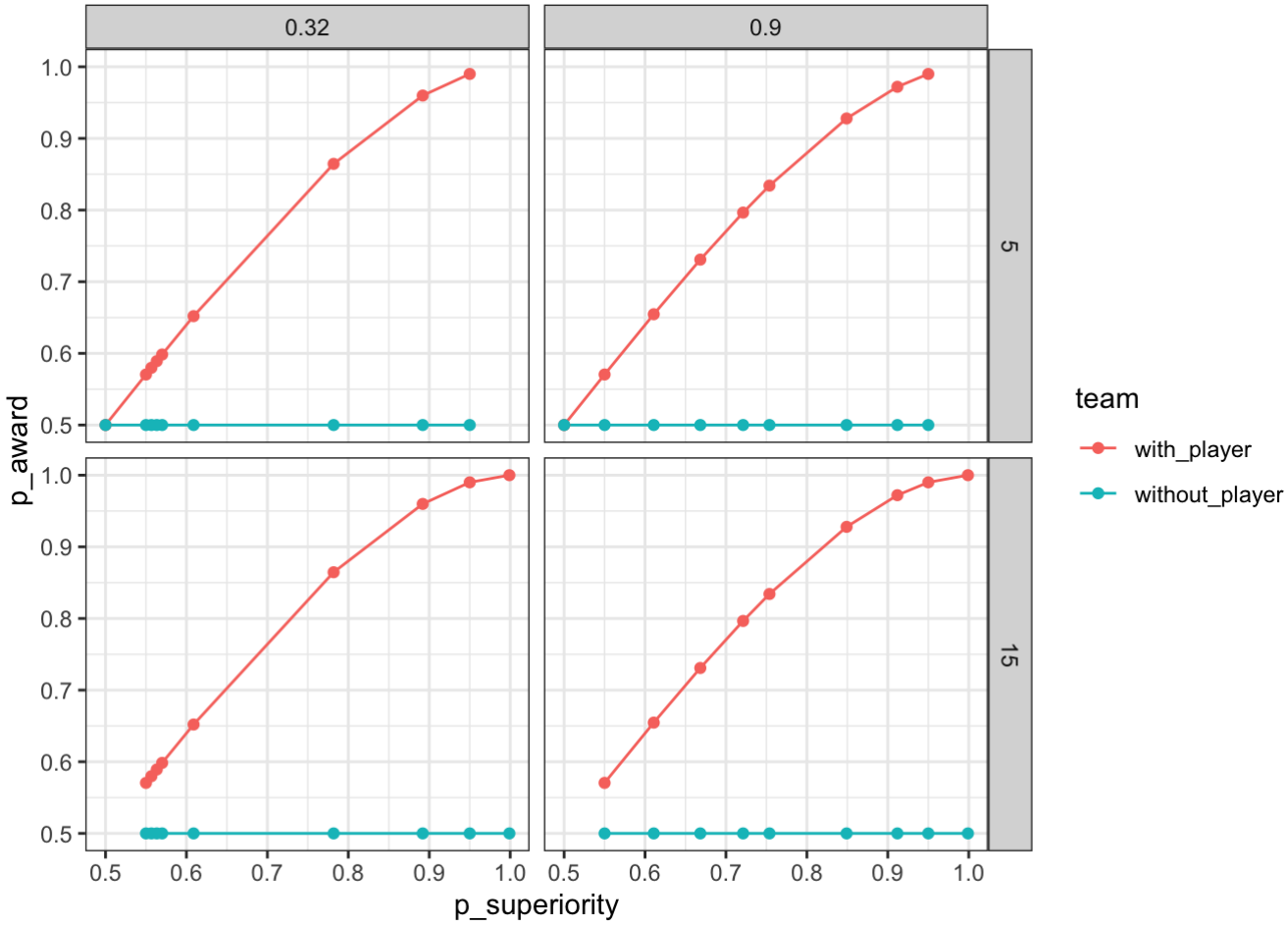
```
##   p_superiority Cohens_d_threshold baseline point_threshold sd_diff
## 1     0.5500000               0.32      0.5             100       5
## 2     0.5566747               0.32      0.5             100       5
## 3     0.5633291               0.32      0.5             100       5
## 4     0.5699607               0.32      0.5             100       5
## 5     0.6088313               0.32      0.5             100       5
## 6     0.7818388               0.32      0.5             100       5
##   mean_diff
## 1 0.6283067
## 2 0.7127189
## 3 0.7970757
## 4 0.8813708
## 5 1.3813708
## 6 3.8920921
```

Now we calculate the summary statistics for the team with and without the new player, making the dataframe double its length up to this point. We derive the standard deviation of the points scored by the teams with and without the new player (sd) from sd_diff, variance sum law, and the assumption that the teams with or without the new player have equal and independent variances. We derive the mean number points scored by the teams with and without the new player (mean) from the threshold for winning the award, the sd of points for each version of the team, and the mean_diff between the number of points for with minus without the new player. We derive the probability of winning the award from the threshold, mean, and sd.

```
# double the length of the dataframe to add information per version of the team, creatin
g a stimulus dataframe with a row per distribution to visualize
stim_df <- map_df(seq_len(2), ~conds_df)
stim_df$team <- as.factor(sort(rep(c("with_player","without_player"), length(stim_df$p_s
uperiority) / 2)))

# add columns for the mean and standard deviation of points for each team and the probab
ility of winning the award
stim_df <- stim_df %>%
  mutate(sd = sqrt(stim_df$sd_diff ^ 2 / 2), # assume equal and independent variances
         mean = if_else(team == "without_player",
                        # team without the new player is at baseline
                        point_threshold - sd * qnorm(1 - baseline),
                        # team with new player is at difference from baseline
                        point_threshold - sd * qnorm(1 - baseline) + mean_diff),
         p_award = pnorm((mean - point_threshold) / sd) # probability of exceeding thresh
old to win award
  )

# spread values per machine across columns to get back to a conditions dataframe one row
per trial
conds_df <- stim_df %>% # explanation: https://kieranhealy.org/blog/archives/2018/11/06/
spreading-multiple-values/
  gather(variable, value, -(p_superiority:team)) %>%
  unite(temp, team, variable) %>%
  spread(temp, value)

head(conds_df)
```

```
##   p_superiority Cohens_d_threshold baseline point_threshold sd_diff
## 1          0.50               0.32      0.5             100       5
## 2          0.50               0.90      0.5             100       5
## 3          0.55               0.32      0.5             100       5
## 4          0.55               0.32      0.5             100      15
## 5          0.55               0.90      0.5             100       5
## 6          0.55               0.90      0.5             100      15
##   mean_diff with_player_mean with_player_p_award with_player_sd
## 1 0.0000000         100.0000           0.5000000       3.535534
## 2 0.0000000         100.0000           0.5000000       3.535534
## 3 0.6283067         100.6283           0.5705254       3.535534
## 4 1.8849202         101.8849           0.5705254      10.606602
## 5 0.6283067         100.6283           0.5705254       3.535534
## 6 1.8849202         101.8849           0.5705254      10.606602
##   without_player_mean without_player_p_award without_player_sd
## 1                 100                   0.5          3.535534
## 2                 100                   0.5          3.535534
## 3                 100                   0.5          3.535534
## 4                 100                   0.5         10.606602
## 5                 100                   0.5          3.535534
## 6                 100                   0.5         10.606602
```

This results in an experimental design where the probability of winning the award with the new player increases monotonically with p_superiority. This means that users should intervene only at high values of p_superiority. Even though the decision rule is not defined in terms of p_superiority, users can user effect size as a proxy for the decision task.



Contract Values and Thresholds

Since we hold the cost of the new player constant at $1M, we can think of K as the *value of the award in millions of dollars*. We need to set K based on the Cohen's d thresholds we chose above so that there is an *equal number of trials where users should vs shouldn't intervene*.

```
# ratio (K) of value of contract (X) over cost of intervention (C)
conds_df <- conds_df %>%
  mutate(
    K = round(1 / (pnorm(sd_diff * (Cohens_d_threshold / sqrt(2)) / sqrt(sd_diff ^ 2 / 2
) - qnorm(baseline)) - baseline), 3)
  )
```

Let's check that we have an equal number of trials where intervening is and is not the optimal choice. This includes the mock and practice trial stimuli at probability of superiority values equal to 0.5 and 0.999, respectively. We will place one of these two at random in the middle of each block of the study as an attention check.

```
# determine whether or not intervention is utility optimal on each trial
conds_df <- conds_df %>%
  mutate(
    should_intervene = (with_player_p_award - without_player_p_award) > (1 / K) # decisi
on rule
  )

conds_df %>%
  group_by(Cohens_d_threshold) %>%
  summarise(
    intervene = sum(should_intervene), n_trials = n()
  )
```

```
## # A tibble: 2 x 3
##   Cohens_d_threshold intervene n_trials
##                <dbl>     <int>    <int>
## 1               0.32         9       18
## 2               0.9          9       18
```

What exactly are the values of probability of superiority at these thresholds? These thresholds are both slightly lower than what we used in previous pilots. This should make our effects larger in probability units since the thresholds are nowhere near the boundaries of the probability scale.

```
conds_df <- conds_df %>%
  mutate(
    p_sup_threshold = pnorm(sqrt(sd_diff ^ 2 / 2) / sd_diff * (qnorm((1 / K) + baseline)
+ qnorm(baseline)))
  )

# unique(conds_df$p_sup_threshold)
conds_df %>%
  group_by(Cohens_d_threshold) %>%
  summarise(
    threshold = unique(round(p_sup_threshold, 5))
  )
```

```
## # A tibble: 2 x 2
##   Cohens_d_threshold threshold
##                <dbl>     <dbl>
## 1               0.32     0.590
## 2               0.9      0.738
```

Expected Bonuses

We also need to set the *starting value of the fantasy sports account* and the *exchange rate* of actual dollars in MTurk bonus per million of dollars in account value.

First, we *set the starting value of the user's account equal to the maximum possible amount participants could lose*, if they buy the new player every trial and always fail to win the award. We'll to this separately for each value of K.

```
# starting value depends on maximum possible loss (if they pay for the new machine every
time and never gain\keep the contract)
conds_df <- conds_df %>% mutate(
    starting_account_value = (4 * n_trials + 2) * K
  )

conds_df %>%
  group_by(Cohens_d_threshold, K, starting_account_value) %>%
  summarise()
```

```
## # A tibble: 2 x 3
## # Groups:   Cohens_d_threshold, K [2]
##   Cohens_d_threshold     K starting_account_value
##                <dbl> <dbl>                  <dbl>
## 1               0.32  7.97                   271.
## 2               0.9   3.16                   108.
```

Next, *we set the exchange rate to range from $0 to $3 depending on decision quality*. To figure out how to calculate bonuses, we want to know what account values would look like at the end of the experiment if users guessed on every trial vs if they made the optimal choice on every trial. To learn this, we'll run a simulation of two response patterns: random guessing vs utility optimal decision-making.

Before we run the simulation, we'll need to adjust the conds_df dataframe to match out intended trial structure. This means doubling the number of trials to account for the within-subjects manipultion of extrinsic means but keeping only two attention checks.

```
# double trials for within-subjects manipulation of extrinsic means
conds_df <- map_df(seq_len(2), ~conds_df)
conds_df$means <- as.factor(sort(rep(c("present", "absent"), length(conds_df$p_superiori
ty) / 2)))
```

```
# drop all but two attention checks
conds_df <- conds_df %>%
  filter(row_number() %in% 3:(n() - 2)) %>%
  arrange(Cohens_d_threshold, means, sd_diff, p_superiority)
```

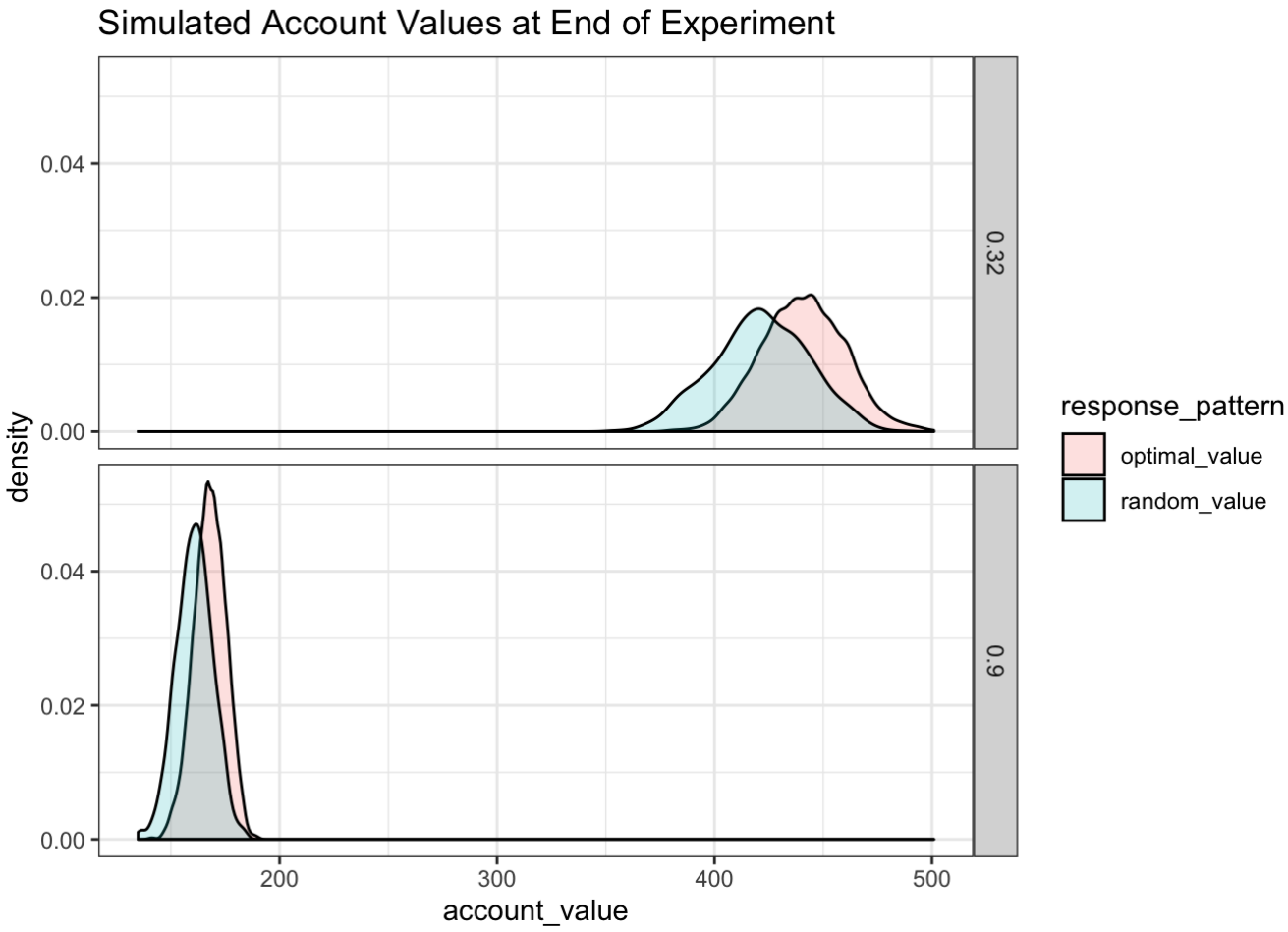
The resulting dataframe contains two sets of trials, one for each level of the Cohen's d threshold since this is a between subjects manipulation. This gives us two times our intended 34 trials. Now let's run our simulation.

```
# set up for simulation
n_iter <- 1e3
simulation_df <- NULL

# generate outcome, whether award is won/kept or not
outcome <- function(intervene, p_with, p_without) {
  if (intervene) {
    return(runif(1) <= p_with)
  } else {
    return(runif(1) <= p_without)
  }
}

for (i in 1:n_iter) {
  temp <- conds_df %>%
    mutate(
      # generate payoff for random guess
      random_guess = as.logical(rbinom(n(), 1, 0.5)),
      random_outcome = as.logical(pmap(list(random_guess, with_player_p_award, without_p
layer_p_award), outcome)),
      random_payoff = if_else(random_outcome,
                              K - random_guess,
                              as.numeric(-random_guess)),
      random_correct = (random_guess == should_intervene),
      # generate payoff for optimal guess
      optimal_outcome = as.logical(pmap(list(should_intervene, with_player_p_award, with
out_player_p_award), outcome)),
      optimal_payoff = if_else(optimal_outcome,
                               K - should_intervene,
                               as.numeric(-should_intervene)),
      optimal_correct = TRUE
    ) %>%
    group_by(Cohens_d_threshold, K, starting_account_value) %>% # summarize separately d
epending on decision rule
    summarize(
      iter = i,
      random_value = mean(starting_account_value) + sum(random_payoff),
      random_correct = sum(random_correct),
      optimal_value = mean(starting_account_value) + sum(optimal_payoff),
      optimal_correct = sum(optimal_correct)
    )
  simulation_df = rbind(simulation_df, temp)
}

head(simulation_df)
```

```
## # A tibble: 6 x 8
## # Groups:   Cohens_d_threshold, K [2]
##   Cohens_d_thresh…     K starting_accoun…  iter random_value random_correct
##              <dbl> <dbl>            <dbl> <int>        <dbl>          <int>
## 1             0.32  7.97             271.     1         403.             20
## 2             0.9   3.16             108.     1         168.             17
## 3             0.32  7.97             271.     2         418.             21
## 4             0.9   3.16             108.     2         171.             17
## 5             0.32  7.97             271.     3         400.             21
## 6             0.9   3.16             108.     3         160.             19
## # … with 2 more variables: optimal_value <dbl>, optimal_correct <int>
```

Let's plot the results of our simulation. We expect to see a discrepancy depending on the effect size threshold since the two conditions we've chosen correspond to different award values.

```
simulation_df %>%
  select(K, Cohens_d_threshold, iter, random_value, optimal_value) %>%
  gather(response_pattern, account_value, -K, -Cohens_d_threshold, -iter) %>%
  ggplot(aes(x = account_value, fill = response_pattern)) +
  geom_density(alpha = 0.2) +
  theme_bw() +
  labs(
    title = "Simulated Account Values at End of Experiment"
  ) +
  facet_grid(Cohens_d_threshold ~ .)
```



We can see that account values aren't reliably different between these two strategies. This is problem for differentiating between good and poor performance.

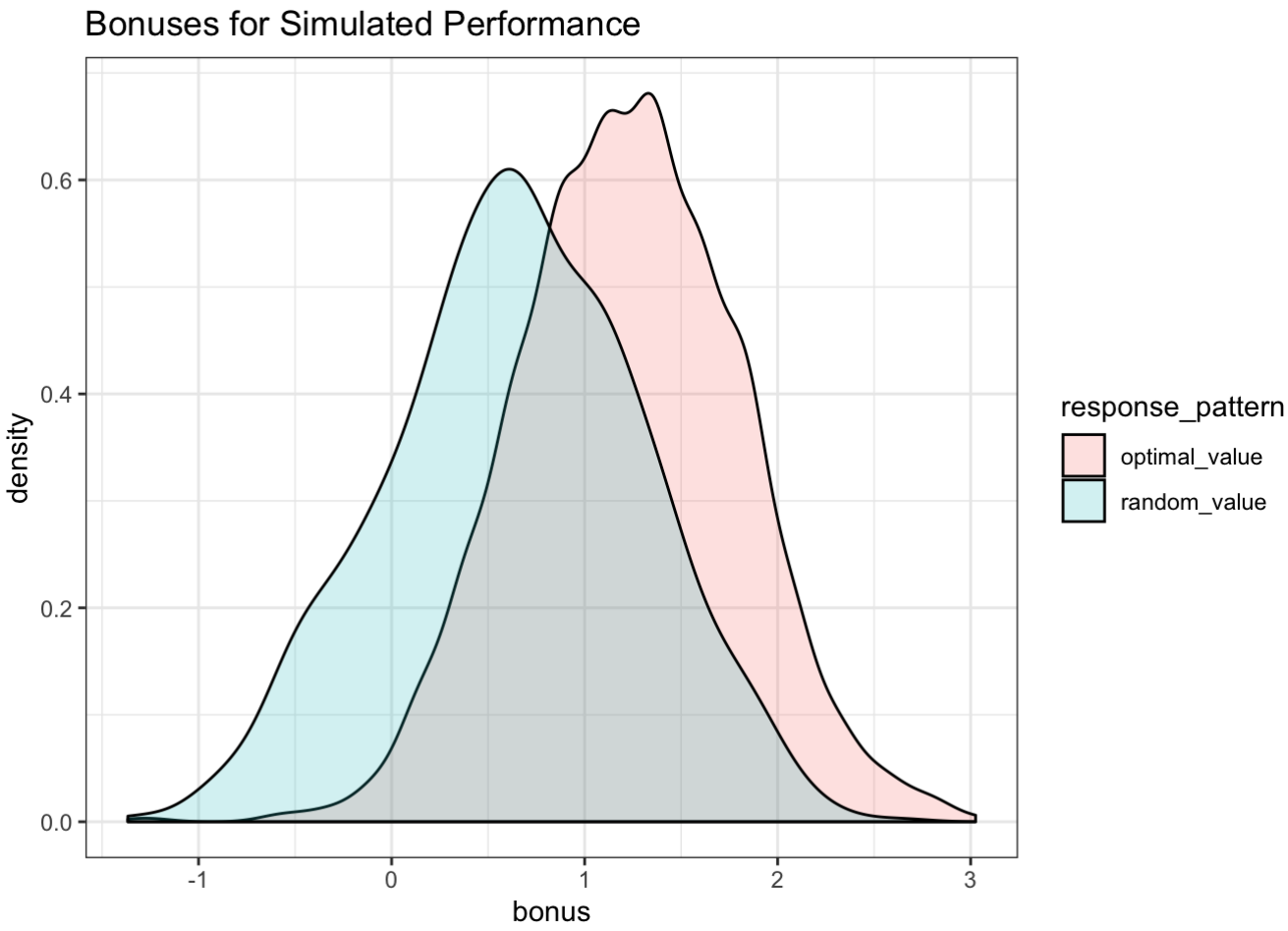
In order to make the incentives more fair, we'll try setting the low end of likely account values under the optimal response distribution to correspond to no bonus. Above that every unit of company value should count for some exchange rate with a maximum bonus of $3. This bonus structure will reward performance the most for account values that are unlikely to be obtainted by random guessing. However, luck still has a large impact on this payoff scheme.

Let's do this separately for each of our two effect size thresholds so that we can try to match the expected value of bonuses across groups of participants.

Let's start with the low effect size threshold condition.

```
# set exchange rate and cutoff
cutoff <- 400
exchange <- 0.03

simulation_df %>%
  filter(Cohens_d_threshold == 0.32) %>%
  select(iter, random_value, optimal_value) %>%
  gather(response_pattern, account_value, -K, -Cohens_d_threshold, -iter) %>%
  mutate(bonus = (account_value - cutoff) * exchange) %>%
  ggplot(aes(x = bonus, fill = response_pattern)) +
  geom_density(alpha = 0.2) +
  theme_bw() +
  labs(
    title = "Bonuses for Simulated Performance"
  )
```

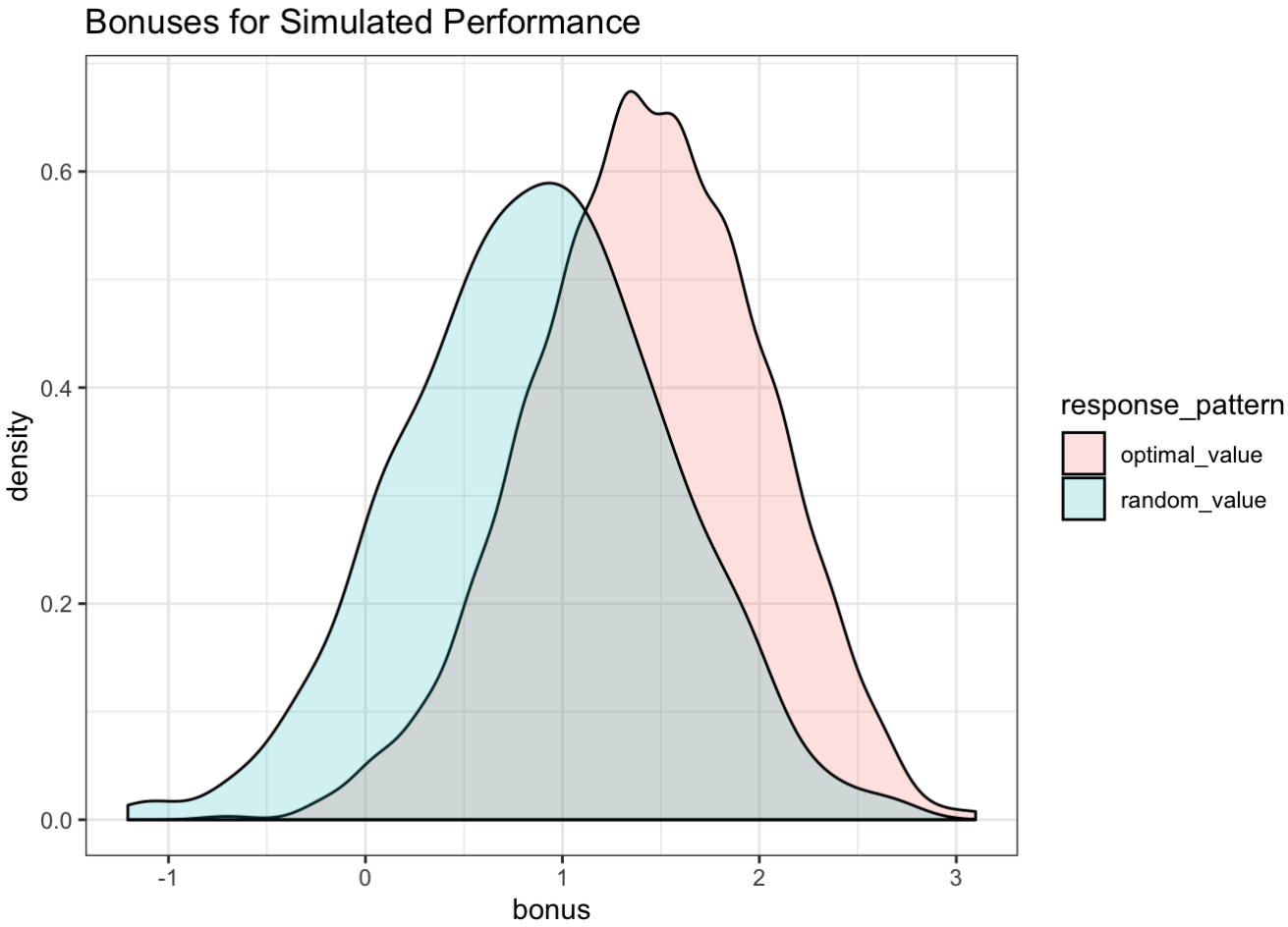
```
## Adding missing grouping variables: `Cohens_d_threshold`, `K`
```



Now, let's try to get the high effect size threshold condition to match.

```
# set exchange rate and cutoff
cutoff <- 150
exchange <- 0.08

simulation_df %>%
  filter(Cohens_d_threshold == 0.9) %>%
  select(iter, random_value, optimal_value) %>%
  gather(response_pattern, account_value, -K, -Cohens_d_threshold, -iter) %>%
  mutate(bonus = (account_value - cutoff) * exchange) %>%
  ggplot(aes(x = bonus, fill = response_pattern)) +
  geom_density(alpha = 0.2) +
  theme_bw() +
  labs(
    title = "Bonuses for Simulated Performance"
  )
```

```
## Adding missing grouping variables: `Cohens_d_threshold`, `K`
```



It’s hard to get a complete match, but these incentives seem to do a decent job without breaking exchange rates into fractions of a cent.