StimuliGeneration

Alex Kale

8/6/2019

This file contains code to generate stimuli for our effect size judgment and decision-making experiment.

Data Conditions

We manipulate the probability of the team scoring or giving up more points with vs without the new player (p_superiority). We employ two sampling strategies, one which optimizes for each of the two questions we ask participants: 1. Linear intervals in logodds units to give perceptually uniform steps in probability of superiority. 2. Probability of superiority values near the utility optimal decision threshold (i.e., p_superiority == [0.13, 0.87]).

When p_superiority is greater than 0.5, the decision task is framed as a gain scenario where the user's team needs to score at least 100 points to win an award. When p_superiority is less than 0.5, the decision task is framed as a loss scenario where the user's team needs to give up fewer than 75 points to keep an award.

```
# linear sampling of log odds for full span of ground truth probability of superiorit
y between 0.025 and 0.975
n_trials.full_span <- 20
logodds.full_span <- seq(log(0.025 / (1 - 0.025)), log(0.975 / (1 - 0.975)), length.o
ut = n_trials.full_span)

# linear sampling of log odds near the decision threshold (p_superiority == [0.13, 0.
87])
n_trials.near_threshold <- 8
logodds.near_threshold <- c(seq(log(0.1 / (1 - 0.1)), log(0.2 /(1 - 0.2)), length.out
= n_trials.near_threshold / 2), # near threshold for loss frame
                            seq(log(0.8 / (1 - 0.8)), log(0.9 /(1 - 0.9)), length.out
= n_trials.near_threshold / 2)) # near threshold for gain frame

# combine the sampling strategies and convert from log odds to probability of superio
rity
logodds <- sort(c(logodds.full_span, logodds.near_threshold))
p_superiority <- 1 / (1 + exp(-logodds))
n_trials <- length(p_superiority)

print(p_superiority)
```

```
##  [1] 0.02500000 0.03633635 0.05253624 0.07539348 0.10000000 0.10707153
##  [7] 0.12709249 0.14990182 0.16021834 0.20000000 0.20591399 0.27605902
## [13] 0.35928769 0.45194404 0.54805596 0.64071231 0.72394098 0.79408601
## [19] 0.80000000 0.83978166 0.85009818 0.87290751 0.89292847 0.90000000
## [25] 0.92460652 0.94746376 0.96366365 0.97500000
```

This time around, 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.

```
# baseline probability of winning/keeping an award without the new player
baseline <- c(.5) # previously c(.15, .5, 8.5)

# initialize data conditions dataframe
conds_df <- data.frame(
  "p_superiority" = rep(p_superiority, length(baseline)),
  "baseline" = sort(rep(baseline, length(p_superiority))))

head(conds_df)
```

```
##   p_superiority baseline
## 1    0.02500000      0.5
## 2    0.03633635      0.5
## 3    0.05253624      0.5
## 4    0.07539348      0.5
## 5    0.10000000      0.5
## 6    0.10707153      0.5
```

We also want to create stimuli for the practice trials. To make these trials easy, we choose a baseline probability of 0.95 and probability of superiority values near 0.5. This way it should be obvious that the new player is not worth the cost, and we can use these trials as an attention check. We create a gain framing version where probability of superiority is 0.6 and a loss framing version where probability of superiority is 0.4.

```
# create df containting rows for practice trials
prac_df <- data.frame(
  "p_superiority" = c(.4, .6),
  "baseline" = c(.95))
# append to conditions dataframe
conds_df <- rbind(conds_df, prac_df)

head(prac_df)
```

```
##   p_superiority baseline
## 1           0.4     0.95
## 2           0.6     0.95
```

When p_superiority is greater than 0.5, the decision task is framed as a gain scenario where the user needs to score at least 100 points win an award. When p_superiority is less than 0.5, the decision task is framed as a loss scenario where the user needs to give up fewer than 75 points to keep an award.

```
# label gain vs loss framing trials based on p_superiority and add award thresholds
conds_df <- conds_df %>%
  mutate(frame = if_else(p_superiority > .5, "gain", "loss"),
         threshold = if_else(frame=="gain",
                             100, # points scored required to win award
                             75)) # points given up required to lose award

head(conds_df)
```

```
##   p_superiority baseline frame threshold
## 1    0.02500000      0.5  loss        75
## 2    0.03633635      0.5  loss        75
## 3    0.05253624      0.5  loss        75
## 4    0.07539348      0.5  loss        75
## 5    0.10000000      0.5  loss        75
## 6    0.10707153      0.5  loss        75
```

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%. We'll need versions of this stimulus condition for each baseline condition and each problem framing.

```
# create df containting rows for mock trial in each condition
mock_df <- data_grid(conds_df, p_superiority = c(.5), baseline = c(.5), frame = uniqu
e(frame)) %>%
  mutate(threshold = if_else(frame=="gain",
                             100, # points scored required to win award
                             75)) # points given up required to lose award
# append to conditions dataframe
conds_df <- rbind(conds_df, mock_df)

print(mock_df)
```

```
## # A tibble: 2 x 4
##   p_superiority baseline frame threshold
##           <dbl>    <dbl> <chr>     <dbl>
## 1           0.5      0.5 gain        100
## 2           0.5      0.5 loss         75
```

We control the standard deviation of the distribution of the difference in points between the team with and without the new player (sd_diff) by setting it to 15. In the gain framing this is 15 points scored. In the loss framing, this is 15 points given up. We can think of this variable as constant across trials. We then 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
 points for the team with vs without the new player
# depending on the gain vs loss frame, these values represent points scored vs points
given up
conds_df <- conds_df %>%
  mutate(sd_diff = 15, # std(new - old)
         mean_diff = sd_diff * qnorm(p_superiority)) # mean(new - old)

head(conds_df)
```

```
##   p_superiority baseline frame threshold sd_diff mean_diff
## 1    0.02500000      0.5  loss        75      15 -29.39946
## 2    0.03633635      0.5  loss        75      15 -26.92321
## 3    0.05253624      0.5  loss        75      15 -24.31117
## 4    0.07539348      0.5  loss        75      15 -21.55136
## 5    0.10000000      0.5  loss        75      15 -19.22327
## 6    0.10707153      0.5  loss        75      15 -18.63380
```

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/keeping 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/keeping the award from the threshold, mean, and sd.

```
# double the length of the dataframe to add information per version of the team, with
a row per distribution to visualize
conds_df <- map_df(seq_len(2), ~conds_df)
conds_df$team <- as.factor(sort(rep(c("With the New Player","Without the New Player"
), length(conds_df$p_superiority) / 2)))

# reorder teams for plotting in consistent order
conds_df$Team <- factor(conds_df$team, levels = c("With the New Player","Without the
 New Player"))

# add columns for the mean and standard deviation of points for each team and the pro
bability of winning/keeping the award
conds_df <- conds_df %>%
  mutate(sd = sqrt(conds_df$sd_diff ^ 2 / 2), # assume equal and independent variance
s
        mean = if_else(Team == "Without the New Player",
                       if_else(frame == "gain", # team without the new player is at b
aseline
                               threshold - sd * qnorm(1 - baseline),
                               threshold - sd * qnorm(baseline)),
                       if_else(frame == "gain", # team with new player is at differen
ce from baseline
                               threshold - sd * qnorm(1 - baseline) + mean_diff,
                               threshold - sd * qnorm(baseline) + mean_diff)),
        p_award = if_else(frame=="gain", # probability of exceeding threshold to win/
keep award
                          1 - pnorm((threshold - mean)/sd),
                          pnorm((threshold - mean)/sd)))

head(conds_df)
```

```
##   p_superiority baseline frame threshold sd_diff mean_diff
## 1    0.02500000      0.5  loss        75      15 -29.39946
## 2    0.03633635      0.5  loss        75      15 -26.92321
## 3    0.05253624      0.5  loss        75      15 -24.31117
## 4    0.07539348      0.5  loss        75      15 -21.55136
## 5    0.10000000      0.5  loss        75      15 -19.22327
## 6    0.10707153      0.5  loss        75      15 -18.63380
##                  team               Team      sd     mean   p_award
## 1 With the New Player With the New Player 10.6066 45.60054 0.9972127
## 2 With the New Player With the New Player 10.6066 48.07679 0.9944311
## 3 With the New Player With the New Player 10.6066 50.68883 0.9890494
## 4 With the New Player With the New Player 10.6066 53.44864 0.9789172
## 5 With the New Player With the New Player 10.6066 55.77673 0.9650368
## 6 With the New Player With the New Player 10.6066 56.36620 0.9605250
```

We name the conditions based on the the baseline and probability of superiority, so we can later filter the rows belonging to the same stimulus.

```
# name conditions
conds_df <- conds_df %>%
  rowwise() %>% # need to name each row differently
  mutate(
    condition = paste(c(baseline, "base", round(p_superiority, 3), "p_sup"), collapse
= "_"),
    condition = if_else(p_superiority == 0.5, # for mock trials where p_sup == 0.5, a
dd framing to condition name to prevent overwriting
                        paste(c(condition, frame), collapse = "_"),
                        condition)) %>%
  ungroup() # need to undo rowwise

head(conds_df)
```

```
## # A tibble: 6 x 12
##   p_superiority baseline frame threshold sd_diff mean_diff team  Team
##           <dbl>    <dbl> <chr>     <dbl>   <dbl>     <dbl> <fct> <fct>
## 1         0.025      0.5 loss         75      15     -29.4 With… With…
## 2         0.0363     0.5 loss         75      15     -26.9 With… With…
## 3         0.0525     0.5 loss         75      15     -24.3 With… With…
## 4         0.0754     0.5 loss         75      15     -21.6 With… With…
## 5         0.1        0.5 loss         75      15     -19.2 With… With…
## 6         0.107      0.5 loss         75      15     -18.6 With… With…
## # … with 4 more variables: sd <dbl>, mean <dbl>, p_award <dbl>,
## #   condition <chr>
```

Since HOPs, quantile dotplots, and densities rely on samples from the underlying data generating process, we need to generate those. However, we don't want to change the shape of our dataframe, so we nest these samples (i.e., draws) inside lists and will later unnest them as needed to produce these specific charts.

```
# for HOPs and quantile dotplots we need to add draws to our dataframe
n <- 1000 # number of samples
n_dots <- 20 # number of dots for quantile dotplots
conds_df$sample_n <- n
conds_df <- conds_df %>% as.tibble() %>%
  mutate(draws = pmap(list(sample_n, mean, sd), rnorm), # get a list of draws from th
e distribution for each condition
         draw_n = list(seq(1, n)), # number each sample in order to animate multiple
 views simultaneously
         quantiles = map(draws, ~ quantile(unlist(.x), ppoints(n_dots)))) # use draw
 to get quantiles
```

```
## Warning: `as.tibble()` is deprecated, use `as_tibble()` (but mind the new semantic
s).
## This warning is displayed once per session.
```

```
         # leave these draws and quantiles nested in the dataframe for later use sinc
e they are not relevant to most visualizations

head(conds_df)
```

```
## # A tibble: 6 x 16
##   p_superiority baseline frame threshold sd_diff mean_diff team  Team
##           <dbl>    <dbl> <chr>     <dbl>   <dbl>     <dbl> <fct> <fct>
## 1         0.025      0.5 loss         75      15     -29.4 With… With…
## 2         0.0363     0.5 loss         75      15     -26.9 With… With…
## 3         0.0525     0.5 loss         75      15     -24.3 With… With…
## 4         0.0754     0.5 loss         75      15     -21.6 With… With…
## 5         0.1        0.5 loss         75      15     -19.2 With… With…
## 6         0.107      0.5 loss         75      15     -18.6 With… With…
## # … with 8 more variables: sd <dbl>, mean <dbl>, p_award <dbl>,
## #   condition <chr>, sample_n <dbl>, draws <list>, draw_n <list>,
## #   quantiles <list>
```

We need to save this dataframe for analysis.

```
# save conds_df with the draws used to create these stimuli (for use in analysis)
save(conds_df, file = "stimuli/conds_df.Rda")
```

Visualization Stimuli

Here, we define functions for each chart type we plan to show users, and we show the gain framing practice trial as an example.

First, let's isolate the data we want to plot.

```
# get the data for the gain framing practice trial to use as an example
gain_prac_df <- conds_df %>% filter(p_superiority == 0.6)

head(gain_prac_df)
```

```
## # A tibble: 2 x 16
##   p_superiority baseline frame threshold sd_diff mean_diff team  Team
##           <dbl>    <dbl> <chr>     <dbl>   <dbl>     <dbl> <fct> <fct>
## 1           0.6     0.95 gain        100      15      3.80 With… With…
## 2           0.6     0.95 gain        100      15      3.80 With… With…
## # … with 8 more variables: sd <dbl>, mean <dbl>, p_award <dbl>,
## #   condition <chr>, sample_n <dbl>, draws <list>, draw_n <list>,
## #   quantiles <list>
```

Before we start building charting functions, we want a helper function to wrap captions and prevent them from running of the edge of our charts.

We also set up some parameters that will remain consistent across charts, including separate x-axis domains for gain/loss framing, parameters specific to HOPs (i.e., frame rate and number of frames), and sizes for geometries and text, respectively.

```
# select limits for x-axis
data_domain_gain <- c(50, 175)
data_domain_loss <- c(0, 125)

# HOPs frame rate
frame_rate <- 2.5
# select number of draws for HOPs conditions
n_draws_hops <- 50

# geom sizes
means_size <- 3
HOPs_size <- 10
HOPs_mean_size_factor <- 1.5
interval_mean_size_factor <- 1.85

# text formating
title_size <- 20
label_size <- 14
caption_size <- 16
char_before_wrap <- 90
```

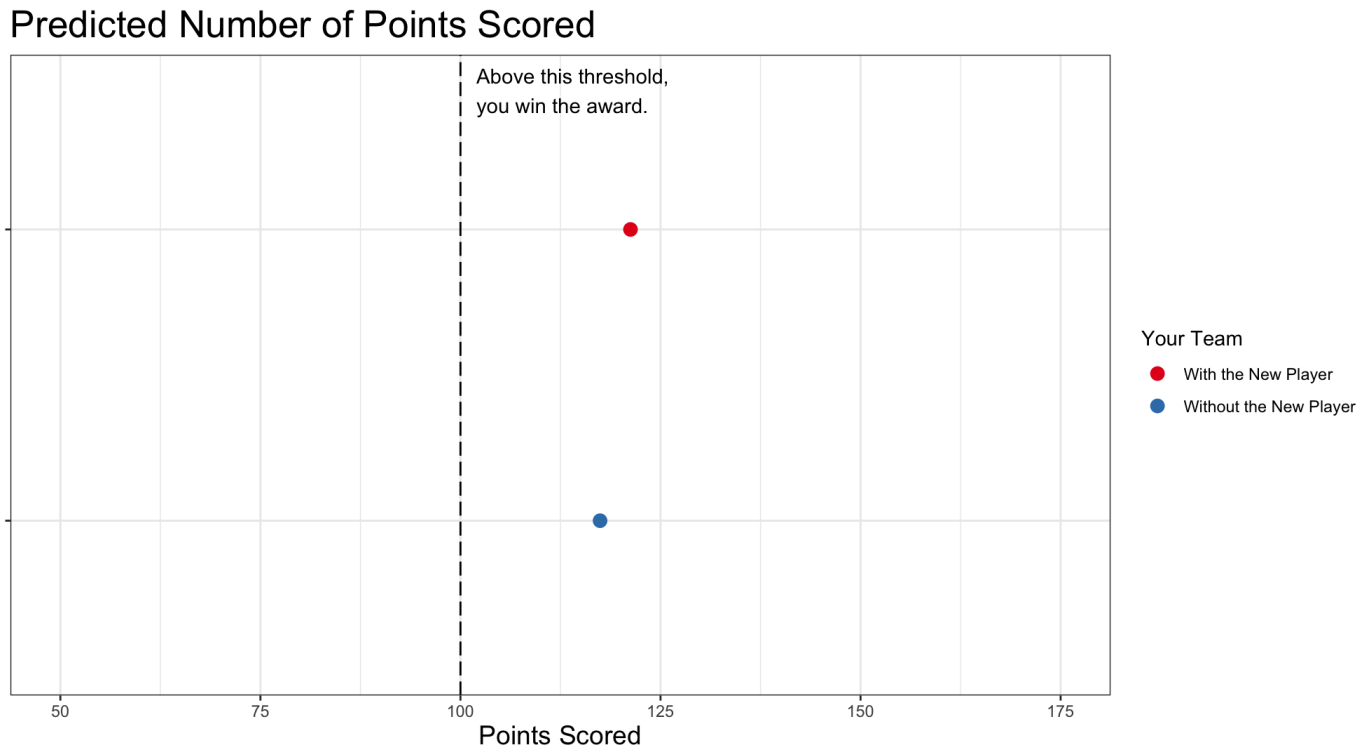
Means Only

A chart function for visualizations showing only means.

```
means_only <- function(df, data_domain, title, x_label, caption, decision_threshold,
 threshold_label) {
  plt <- df %>% ggplot(aes(x = mean, y = reorder(Team, desc(Team)), color = Team)) +
      geom_point(size = means_size) +
      theme_bw() +
      scale_color_brewer(palette = "Set1") +
      xlim(data_domain[1], data_domain[2]) +
      labs(
        title = title,
        x = x_label,
        y = NULL,
        color = "Your Team",
        caption = wrap_label(caption, char_before_wrap)) +
      theme(
        axis.title = element_text(size=label_size),
        axis.text.y = element_blank(),
        plot.title = element_text(size = title_size),
        plot.caption = element_text(size = caption_size, hjust = 0, vjust = -1)) +
      geom_vline(xintercept = decision_threshold, linetype = "longdash") + # award th
reshold
      annotate("text", x = (decision_threshold + 2), y = 2.4, label = threshold_labe
l, hjust = 0, vjust = 0)

  return(plt)
}

means_only(df = gain_prac_df,
           data_domain = data_domain_gain,
           title = "Predicted Number of Points Scored",
           x_label = "Points Scored",
           caption = "Dots represent the average number of points that could be score
d by your team with (top) and without the new player (bottom).",
           decision_threshold = 100,
           threshold_label = "Above this threshold,\nyou win the award.")
```



Dots represent the average number of points that could be scored by your team with (top) and without the new player (bottom).

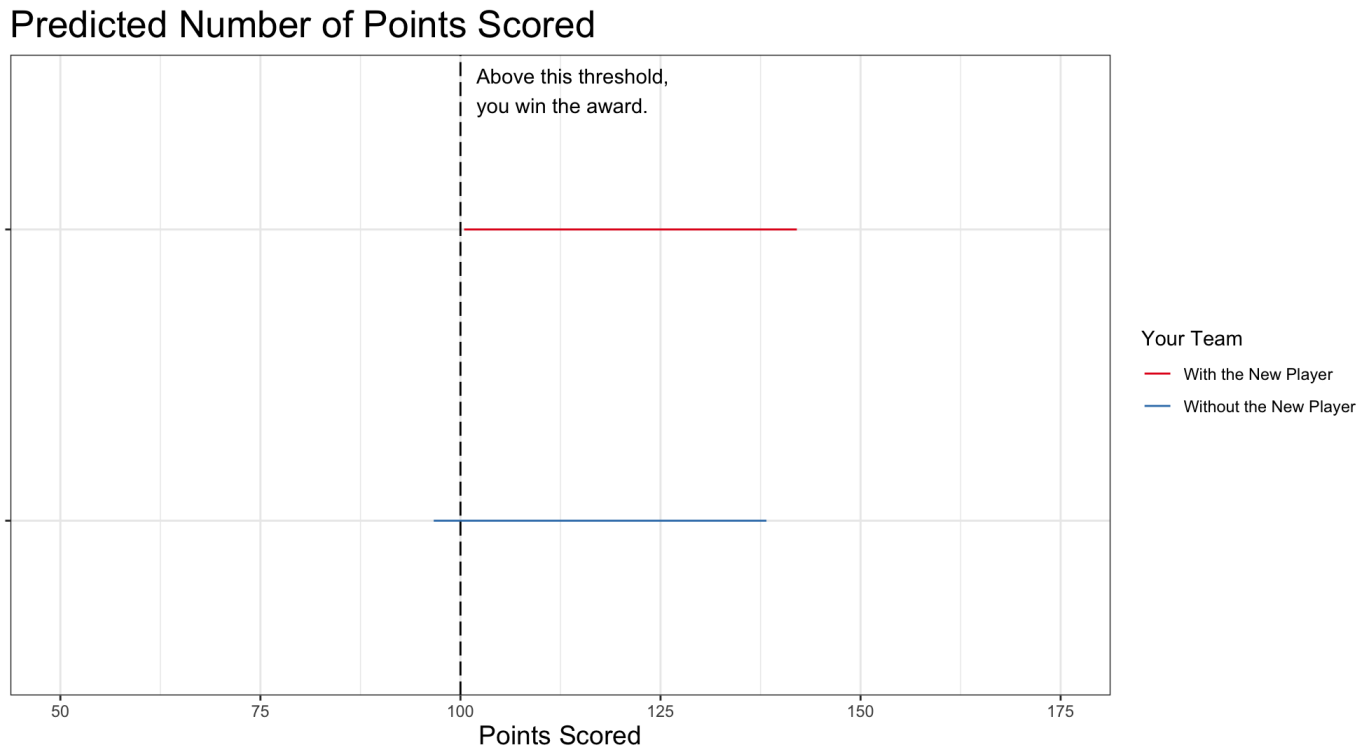
Intervals Only

A chart function for visualizations showing only 95% containment intervals.

```
intervals_only <- function(df, data_domain, title, x_label, caption, decision_thresho
ld, threshold_label) {
  plt <- df %>% ggplot(aes(y = mean, x = reorder(Team, desc(Team)), color = Team)) +
      geom_errorbar(aes(ymin = mean + qnorm(0.025) * sd, ymax = mean + qnorm(0.975) *
sd, width = 0)) +
      coord_flip() +
      theme_bw() +
      scale_color_brewer(palette = "Set1") +
      ylim(data_domain[1], data_domain[2]) +
      labs(
        title = title,
        x = NULL,
        y = x_label,
        color = "Your Team",
        caption = wrap_label(caption, char_before_wrap)) +
      theme(
        axis.title = element_text(size=label_size),
        axis.text.y = element_blank(),
        plot.title = element_text(size = title_size),
        plot.caption = element_text(size = caption_size, hjust = 0, vjust = -1)) +
      geom_hline(yintercept = decision_threshold, linetype = "longdash") + # award th
reshold
      annotate("text", y = (decision_threshold + 2), x = 2.4, label = threshold_labe
l, hjust = 0, vjust = 0)

  return(plt)
}

intervals_only(df = gain_prac_df,
               data_domain = data_domain_gain,
               title = "Predicted Number of Points Scored",
               x_label = "Points Scored",
               caption = "Intervals contain 95% of the possible numbers of points tha
t could be scored by your team with (top) and without the new player (bottom).",
               decision_threshold = 100,
               threshold_label = "Above this threshold,\nyou win the award.")
```



Intervals contain 95% of the possible numbers of points that could be scored by your team with (top) and without the new player (bottom).

Intervals With Means

A chart function for visualizations showing only 95% containment intervals with means.

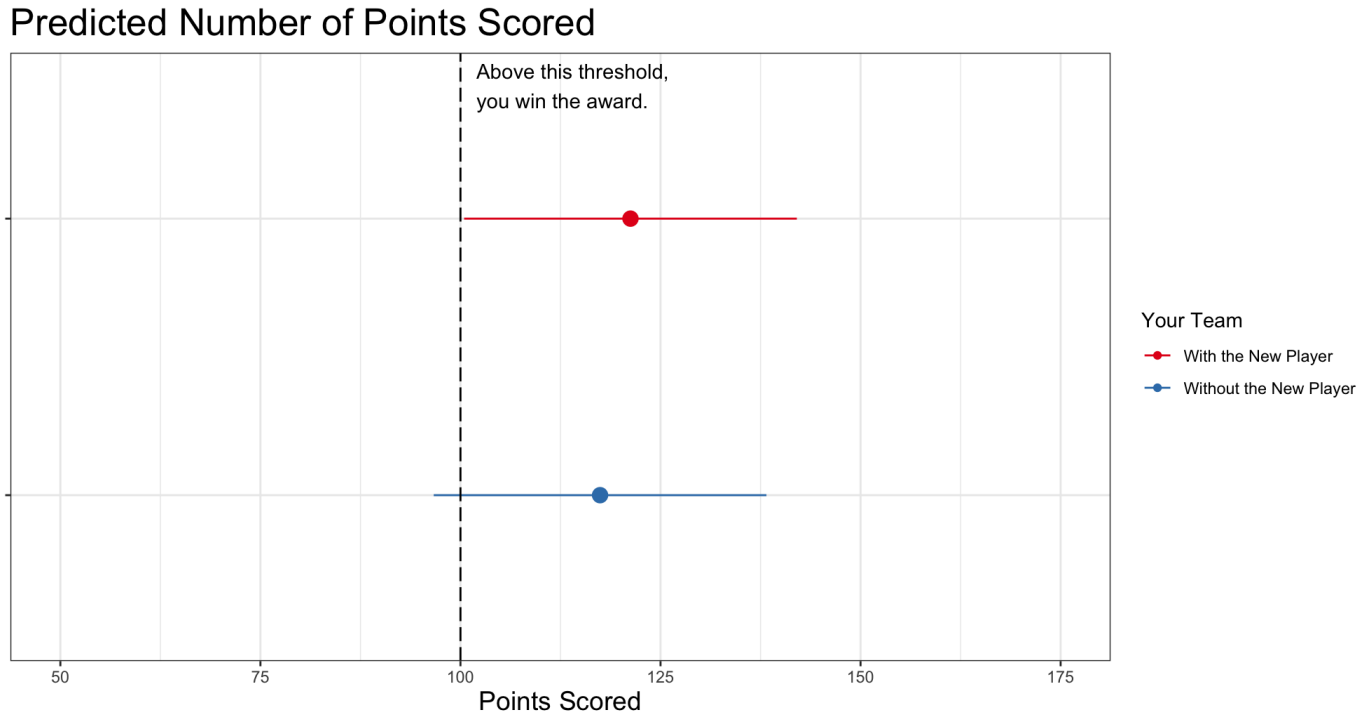
```
intervals_w_means <- function(df, data_domain, title, x_label, caption, decision_thre
shold, threshold_label) {
  plt <- df %>% ggplot(aes(y = mean, x = reorder(Team, desc(Team)), color = Team)) +
      geom_pointrange(aes(ymin = mean + qnorm(0.025) * sd, ymax = mean + qnorm(0.975)
* sd), show.legend = FALSE, fatten = means_size * interval_mean_size_factor) +
      geom_line(aes(y = mean - 1000)) + geom_point(aes(y = mean - 1000)) + # hack to
 get legend symbols oriented properly
      coord_flip() +
      theme_bw() +
      scale_color_brewer(palette = "Set1") +
      ylim(data_domain[1], data_domain[2]) +
      labs(
        title = title,
        x = NULL,
        y = x_label,
        color = "Your Team",
        caption = wrap_label(caption, char_before_wrap)) +
      theme(
        axis.title = element_text(size=label_size),
        axis.text.y = element_blank(),
        plot.title = element_text(size = title_size),
        plot.caption = element_text(size = caption_size, hjust = 0, vjust = -1)) +
      geom_hline(yintercept = decision_threshold, linetype = "longdash") + # award th
reshold
      annotate("text", y = (decision_threshold + 2), x = 2.4, label = threshold_labe
l, hjust = 0, vjust = 0)

  return(plt)
}

intervals_w_means(df = gain_prac_df,
                  data_domain = data_domain_gain,
                  title = "Predicted Number of Points Scored",
                  x_label = "Points Scored",
                  caption = "Dots represent the average number of points that could b
e scored by your team with (top) and without the new player (bottom). Intervals conta
in 95% of the possible numbers of points that could be scored.",
                  decision_threshold = 100,
                  threshold_label = "Above this threshold,\nyou win the award.")
```

```
## Warning: Removed 2 rows containing missing values (geom_path).
```

```
## Warning: Removed 2 rows containing missing values (geom_point).
```



Dots represent the average number of points that could be scored by your team with (top) and without the new player (bottom). Intervals contain 95% of the possible numbers of points that could be scored.

Hypothetical Outcome Plots (HOPs)

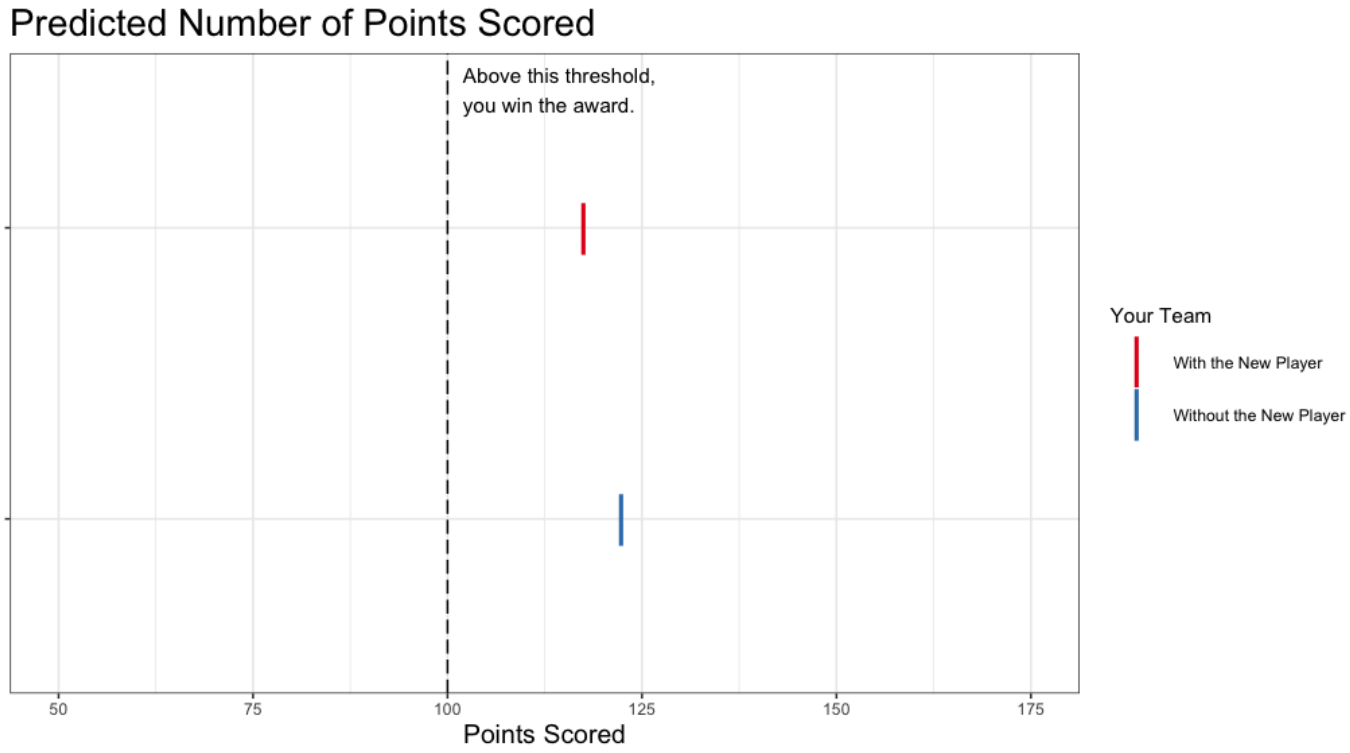
A chart function for HOPs of the possible points for each version of the team.

```
hops <- function(df, n_draws, frames_per_second, data_domain, title, x_label, captio
n, decision_threshold, threshold_label, dimensions) {
  plt <- df %>% select(-one_of(c("quantiles"))) %>% unnest() %>%
      filter(draw_n %in% 1:n_draws) %>% # filter to set number of draws
      ggplot(aes(y = draws, x = reorder(Team, desc(Team)), color = Team)) +
      geom_point(shape = 124, size = HOPs_size) +
      coord_flip() +
      theme_bw() +
      scale_color_brewer(palette = "Set1") +
      ylim(data_domain[1], data_domain[2]) +
      labs(
        title = title,
        x = NULL,
        y = x_label,
        color = "Your Team",
        caption = wrap_label(caption, char_before_wrap)) +
      theme(
        axis.title = element_text(size=label_size),
        axis.text.y = element_blank(),
        plot.title = element_text(size = title_size),
        plot.caption = element_text(size = caption_size, hjust = 0, vjust = -1)) +
      geom_hline(yintercept = decision_threshold, linetype = "longdash") + # award th
reshold
      annotate("text", y = (decision_threshold + 2), x = 2.4, label = threshold_labe
l, hjust = 0, vjust = 0) +
      transition_manual(draw_n)
    animation <- animate(plt, fps = frames_per_second, nframes = 10 * frames_per_seco
nd, res = 100, width = dimensions[1]*100, height = dimensions[2]*100)

  return(animation)
}

hops(df = gain_prac_df,
     n_draws = n_draws_hops,
     frames_per_second = frame_rate,
     data_domain = data_domain_gain,
     title = "Predicted Number of Points Scored",
     x_label = "Points Scored",
     caption = "Lines represent individual predictions of the number of points that c
ould be scored by your team with (top) and without the new player (bottom).",
     decision_threshold = 100,
     threshold_label = "Above this threshold,\nyou win the award.",
     dimensions = c(10.26667, 6.16000))
```

```
## Warning: Unquoting language objects with `!!!` is deprecated as of rlang 0.4.0.
## Please use `!!` instead.
##
##   # Bad:
##   dplyr::select(data, !!!enquo(x))
##
##   # Good:
##   dplyr::select(data, !!enquo(x))    # Unquote single quosure
##   dplyr::select(data, !!!enquos(x))  # Splice list of quosures
##
## This warning is displayed once per session.
```



Lines represent individual predictions of the number of points that could be scored by your team with (top) and without the new player (bottom).

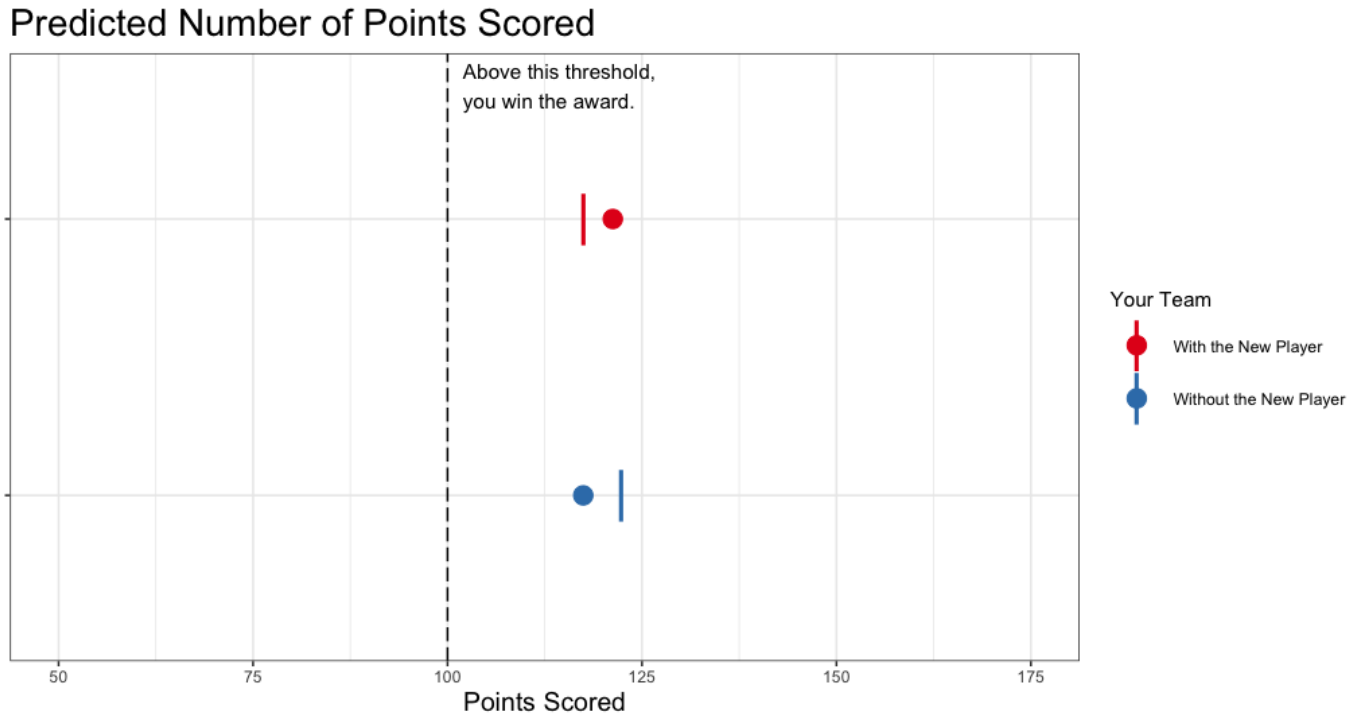
Hypothetical Outcome Plots (HOPs) with Means

A chart function for HOPs of the possible points for each version of the team, with means added.

```
hops_w_means <- function(df, n_draws, frames_per_second, data_domain, title, x_label,
caption, decision_threshold, threshold_label, dimensions) {
  plt <- df %>% select(-one_of(c("quantiles"))) %>% unnest() %>%
      filter(draw_n %in% 1:n_draws) %>% # filter to set number of draws
      ggplot(aes(y = mean, x = reorder(Team, desc(Team)), color = Team)) +
      geom_point(size = means_size * HOPs_mean_size_factor) +
      geom_point(aes(y = draws), shape = 124, size = HOPs_size) +
      coord_flip() +
      theme_bw() +
      scale_color_brewer(palette = "Set1") +
      ylim(data_domain[1], data_domain[2]) +
      labs(
        title = title,
        x = NULL,
        y = x_label,
        color = "Your Team",
        caption = wrap_label(caption, char_before_wrap)) +
      theme(
        axis.title = element_text(size=label_size),
        axis.text.y = element_blank(),
        plot.title = element_text(size = title_size),
        plot.caption = element_text(size = caption_size, hjust = 0, vjust = -1)) +
      geom_hline(yintercept = decision_threshold, linetype = "longdash") + # award th
reshold
      annotate("text", y = (decision_threshold + 2), x = 2.4, label = threshold_labe
l, hjust = 0, vjust = 0) +
      transition_manual(draw_n)
    animation <- animate(plt, fps = frames_per_second, nframes = 10 * frames_per_seco
nd, res = 100, width = dimensions[1]*100, height = dimensions[2]*100)

  return(animation)
}

hops_w_means(df = gain_prac_df,
             n_draws = n_draws_hops,
             frames_per_second = frame_rate,
             data_domain = data_domain_gain,
             title = "Predicted Number of Points Scored",
             x_label = "Points Scored",
             caption = "Dots represent the average number of points that could be sco
red by your team with (top) and without the new player (bottom). Lines represent indi
vidual predictions of the number of points that could be scored.",
             decision_threshold = 100,
             threshold_label = "Above this threshold,\nyou win the award.",
             dimensions = c(10.26667, 6.16000))
```



Dots represent the average number of points that could be scored by your team with (top) and without the new player (bottom). Lines represent individual predictions of the number of points that could be scored.

Quantile Dotplots

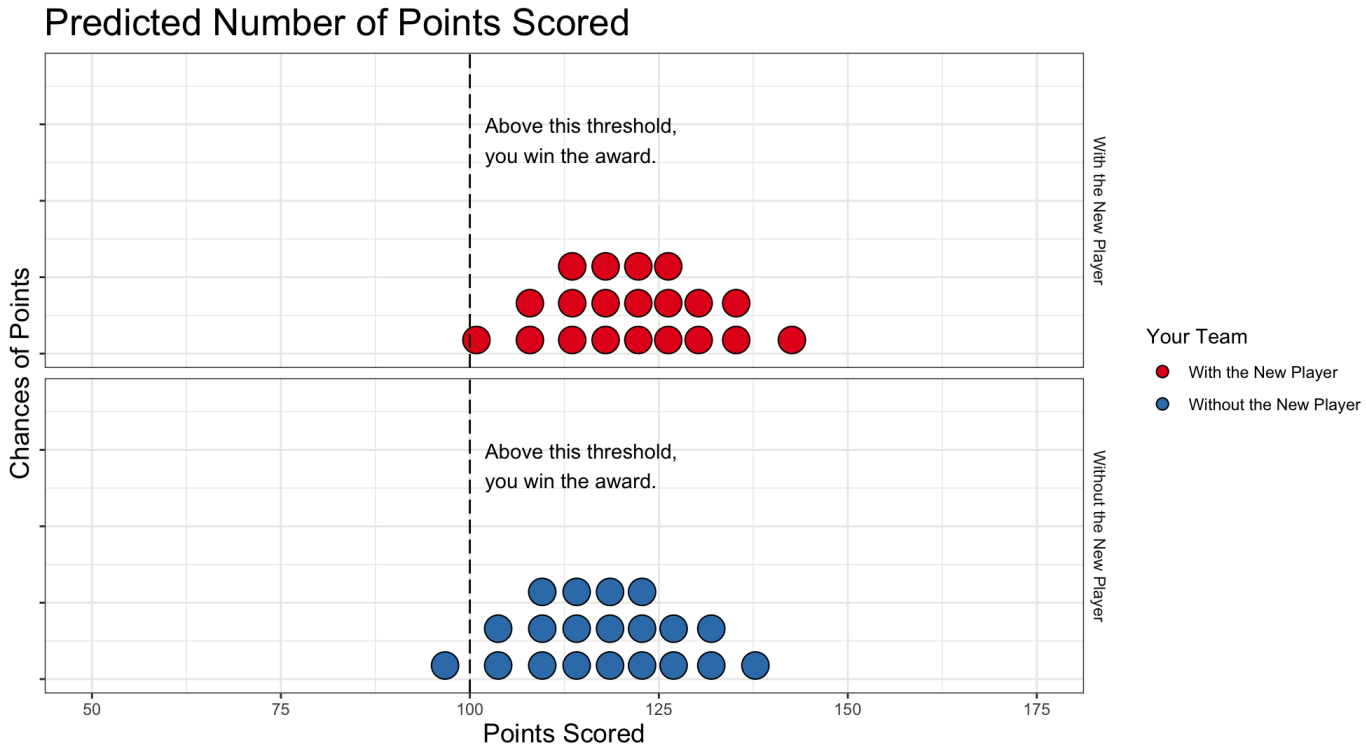
A chart function for quantile dotplots of the possible points for each version of the team.

```
quantile_dotplots <- function(df, data_domain, title, x_label, caption, decision_thre
shold, threshold_label) {
  plt <- df %>% select(-one_of(c("draws", "draw_n"))) %>% unnest() %>%
      ggplot(aes(x = quantiles, fill = Team)) +
      geom_dotplot(binwidth = 4, binaxis = "x", dotsize = .9, stackratio = 1.35) +
      theme_bw() +
      scale_fill_brewer(palette = "Set1") +
      facet_grid(Team ~ .) +
      xlim(data_domain[1], data_domain[2]) +
      ylim(0, .075) +
      labs(
        title = title,
        x = x_label,
        y = "Chances of Points",
        fill = "Your Team",
        caption = wrap_label(caption, char_before_wrap)) +
      theme(
        strip.background = element_blank(),
        strip.text.x = element_blank(),
        axis.title = element_text(size=label_size),
        axis.text.y = element_blank(),
        plot.title = element_text(size = title_size),
        plot.caption = element_text(size = caption_size, hjust = 0, vjust = -1)) +

      geom_vline(xintercept = decision_threshold, linetype = "longdash") + # award th
reshold
      annotate("text", x = (decision_threshold + 2), y = 0.05, label = threshold_labe
l, hjust = 0, vjust = 0)

  return(plt)
}

quantile_dotplots(df = gain_prac_df,
                  data_domain = data_domain_gain,
                  title = "Predicted Number of Points Scored",
                  x_label = "Points Scored",
                  caption = "Each dot represents a 5% chance that different numbers o
f points could be scored by your team with (top) and without the new player (botto
m).",
                  decision_threshold = 100,
                  threshold_label = "Above this threshold,\nyou win the award.")
```



Each dot represents a 5% chance that different numbers of points could be scored by your team with (top) and without the new player (bottom).

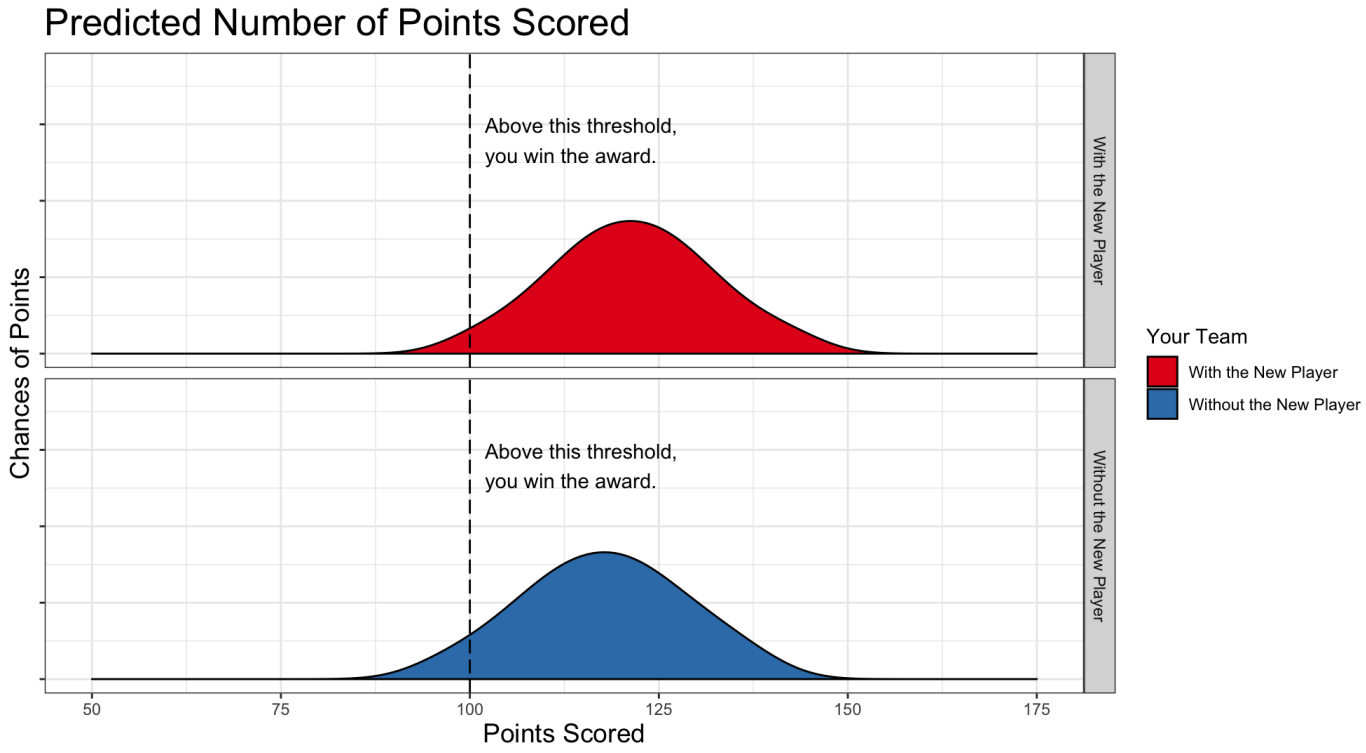
Densities

A chart function for continuous probability densities of the possible points for each version of the team.

```
densities <- function(df, data_domain, title, x_label, caption, decision_threshold, t
hreshold_label) {
  plt <- df %>% select(-one_of(c("draws", "draw_n"))) %>% unnest() %>%
      ggplot(aes(x = quantiles, fill = Team)) +
      geom_density() +
      theme_bw() +
      scale_fill_brewer(palette = "Set1") +
      facet_grid(Team ~ .) +
      xlim(data_domain[1], data_domain[2]) +
      ylim(0, .075) +
      labs(
        title = title,
        x = x_label,
        y = "Chances of Points",
        fill = "Your Team",
        caption = wrap_label(caption, char_before_wrap)) +
      theme(
        axis.title = element_text(size=label_size),
        axis.text.y = element_blank(),
        plot.title = element_text(size = title_size),
        plot.caption = element_text(size = caption_size, hjust = 0, vjust = -1)) +
      geom_vline(xintercept = decision_threshold, linetype = "longdash") + # award th
reshold
      annotate("text", x = (decision_threshold + 2), y = 0.05, label = threshold_labe
l, hjust = 0, vjust = 0)

  return(plt)
}

densities(df = gain_prac_df,
          data_domain = data_domain_gain,
          title = "Predicted Number of Points Scored",
          x_label = "Points Scored",
          caption = "The height of the shape represents the chances that different nu
mbers of points could be scored by your team with (top) and without the new player (b
ottom).",
          decision_threshold = 100,
          threshold_label = "Above this threshold,\nyou win the award.")
```



The height of the shape represents the chances that different numbers of points could be scored by your team with (top) and without the new player (bottom).

Stimuli Generation

We create one of each chart type for each data condition above and save to a folder called stimuli.

```
# set plot dimensions
dims_pix <- c(770, 462) # pixel dimensions
ppi <- 75 # assume 75 ppi for the avg monitor
dims <- dims_pix / ppi # dimensions in inches

# cycle through rows in the table of data conditions
for (c in unique(conds_df$condition)) {
  # isolaten data for the current condtion
  use_df <- conds_df %>% filter(condition %in% c)

  if (all(use_df$frame=="gain")) { # stimuli for gain framing trials
    # means only
    plt <- means_only(df = use_df,
      data_domain = data_domain_gain,
      title = "Predicted Number of Points Scored",
      x_label = "Points Scored",
      caption = "Dots represent the average number of points that could be scored by
 your team with (top) and without the new player (bottom).",
      decision_threshold = 100,
      threshold_label = "Above this threshold,\nyou win the award.")
    fname <- paste("stimuli/means_only-", c,".svg", sep = "")
    ggsave(file = fname, plot = plt, width = dims[1], height = dims[2])

    # intervals only
    plt <- intervals_only(df = use_df,
      data_domain = data_domain_gain,
      title = "Predicted Number of Points Scored",
      x_label = "Points Scored",
      caption = "Intervals contain 95% of the possible numbers of points that could b
e scored by your team with (top) and without the new player (bottom).",
      decision_threshold = 100,
      threshold_label = "Above this threshold,\nyou win the award.")
    fname <- paste("stimuli/intervals_only-", c,".svg", sep = "")
    ggsave(file = fname, plot = plt, width = dims[1], height = dims[2])

    # intervals with means
    plt <- intervals_w_means(df = use_df,
      data_domain = data_domain_gain,
      title = "Predicted Number of Points Scored",
      x_label = "Points Scored",
      caption = "Dots represent the average number of points that could be scored by
 your team with (top) and without the new player (bottom). Intervals contain 95% of t
he possible numbers of points that could be scored.",
      decision_threshold = 100,
      threshold_label = "Above this threshold,\nyou win the award.")
    fname <- paste("stimuli/intervals_w_means-", c, ".svg", sep = "")
    ggsave(file = fname, plot = plt, width = dims[1], height = dims[2])

    # hops
    plt <- hops(df = use_df,
      n_draws = n_draws_hops,
      frames_per_second = frame_rate,
      data_domain = data_domain_gain,
      title = "Predicted Number of Points Scored",
      x_label = "Points Scored",
      caption = "Lines represent individual predictions of the number of points that
 could be scored by your team with (top) and without the new player (bottom).",
```

```
      decision_threshold = 100,
      threshold_label = "Above this threshold,\nyou win the award.",
      dimensions = dims)
    fname <- paste("stimuli/HOPs-", c, ".gif", sep = "")
    anim_save(filename = fname, animation = plt)

    # hops with means
    plt <- hops_w_means(df = use_df,
      n_draws = n_draws_hops,
      frames_per_second = frame_rate,
      data_domain = data_domain_gain,
      title = "Predicted Number of Points Scored",
      x_label = "Points Scored",
      caption = "Dots represent the average number of points that could be scored by
 your team with (top) and without the new player (bottom). Lines represent individual
predictions of the number of points that could be scored.",
      decision_threshold = 100,
      threshold_label = "Above this threshold,\nyou win the award.",
      dimensions = dims)
    fname <- paste("stimuli/HOPs_w_means-", c, ".gif", sep = "")
    anim_save(filename = fname, animation = plt)

    # quantile dotplots
    plt <- quantile_dotplots(df = use_df,
      data_domain = data_domain_gain,
      title = "Predicted Number of Points Scored",
      x_label = "Points Scored",
      caption = "Each dot represents a 5% chance that different numbers of points cou
ld be scored by your team with (top) and without the new player (bottom).",
      decision_threshold = 100,
      threshold_label = "Above this threshold,\nyou win the award.")
    fname <- paste("stimuli/QDPs-", c, ".svg", sep = "")
    ggsave(file = fname, plot = plt, width = dims[1], height = dims[2])

    # densities
    plt <- densities(df = use_df,
      data_domain = data_domain_gain,
      title = "Predicted Number of Points Scored",
      x_label = "Points Scored",
      caption = "The height of the shape represents the chances that different number
s of points could be scored by your team with (top) and without the new player (botto
m).",
      decision_threshold = 100,
      threshold_label = "Above this threshold,\nyou win the award.")
    fname <- paste("stimuli/densities-", c, ".svg", sep = "")
    ggsave(file = fname, plot = plt, width = dims[1], height = dims[2])


  } else { # stimuli for loss framing trials
    # means only
    plt <- means_only(df = use_df,
      data_domain = data_domain_loss,
      title = "Predicted Number of Points Given Up",
      x_label = "Points Given Up",
      caption = "Dots represent the average number of points that could be given up b
y your team with (top) and without the new player (bottom).",
      decision_threshold = 75,
      threshold_label = "Above this threshold,\nyou lose the award.")
```

```
    fname <- paste("stimuli/means_only-", c,".svg", sep = "")
    ggsave(file = fname, plot = plt, width = dims[1], height = dims[2])

    # intervals only
    plt <- intervals_only(df = use_df,
      data_domain = data_domain_loss,
      title = "Predicted Number of Points Given Up",
      x_label = "Points Given Up",
      caption = "Intervals contain 95% of the possible numbers of points that could b
e given up by your team with (top) and without the new player (bottom).",
      decision_threshold = 75,
      threshold_label = "Above this threshold,\nyou lose the award.")
    fname <- paste("stimuli/intervals_only-", c, ".svg", sep = "")
    ggsave(file = fname, plot = plt, width = dims[1], height = dims[2])

     # intervals with means
    plt <- intervals_w_means(df = use_df,
      data_domain = data_domain_loss,
      title = "Predicted Number of Points Given Up",
      x_label = "Points Given Up",
      caption = "Dots represent the average number of points that could be given up b
y your team with (top) and without the new player (bottom). Intervals contain 95% of
 the possible numbers of points that could be given up.",
      decision_threshold = 75,
      threshold_label = "Above this threshold,\nyou lose the award.")
    fname <- paste("stimuli/intervals_w_means-", c, ".svg", sep = "")
    ggsave(file = fname, plot = plt, width = dims[1], height = dims[2])

    # hops
    plt <- hops(df = use_df,
      n_draws = n_draws_hops,
      frames_per_second = frame_rate,
      data_domain = data_domain_loss,
      title = "Predicted Number of Points Given Up",
      x_label = "Points Given Up",
      caption = "Lines represent individual predictions of the number of points that
 could be given up by your team with (top) and without the new player (bottom).",
      decision_threshold = 75,
      threshold_label = "Above this threshold,\nyou lose the award.",
      dimensions = dims)
    fname <- paste("stimuli/HOPs-", c, ".gif", sep = "")
    anim_save(filename = fname, animation = plt)

    # hops with means
    plt <- hops_w_means(df = use_df,
      n_draws = n_draws_hops,
      frames_per_second = frame_rate,
      data_domain = data_domain_loss,
      title = "Predicted Number of Points Given Up",
      x_label = "Points Given Up",
      caption = "Dots represent the average number of points that could be given up b
y your team with (top) and without the new player (bottom). Lines represent individua
l predictions of the number of points that could be given up.",
      decision_threshold = 75,
      threshold_label = "Above this threshold,\nyou lose the award.",
      dimensions = dims)
    fname <- paste("stimuli/HOPs_w_means-", c, ".gif", sep = "")
    anim_save(filename = fname, animation = plt)
```

```
    # quantile dotplots
    plt <- quantile_dotplots(df = use_df,
      data_domain = data_domain_loss,
      title = "Predicted Number of Points Given Up",
      x_label = "Points Given Up",
      caption = "Each dot represents a 5% chance that different numbers of points cou
ld be given up by your team with (top) and without the new player (bottom).",
      decision_threshold = 75,
      threshold_label = "Above this threshold,\nyou lose the award.")
    fname <- paste("stimuli/QDPs-", c, ".svg", sep = "")
    ggsave(file = fname, plot = plt, width = dims[1], height = dims[2])

    # densities
    plt <- densities(df = use_df,
      data_domain = data_domain_loss,
      title = "Predicted Number of Points Given Up",
      x_label = "Points Given Up",
      caption = "The height of the shape represents the chances that different number
s of points could be given up by your team with (top) and without the new player (bot
tom).",
      decision_threshold = 75,
      threshold_label = "Above this threshold,\nyou lose the award.")
    fname <- paste("stimuli/densities-", c, ".svg", sep = "")
    ggsave(file = fname, plot = plt, width = dims[1], height = dims[2])
  }
}
```