IDS 126 - Project

Kyle Weng

2020-11-25

```
setwd("J:/Academic Archives/FA 2020/IDSEcPs 126/Project")
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0
## -- Conflicts ------------------------------------------------------------------------------------ tidy
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(magrittr)
```

```
##
## Attaching package: 'magrittr'
## The following object is masked from 'package:purrr':
##
##     set_names
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
library(ggplot2)
library(rstanarm)
```

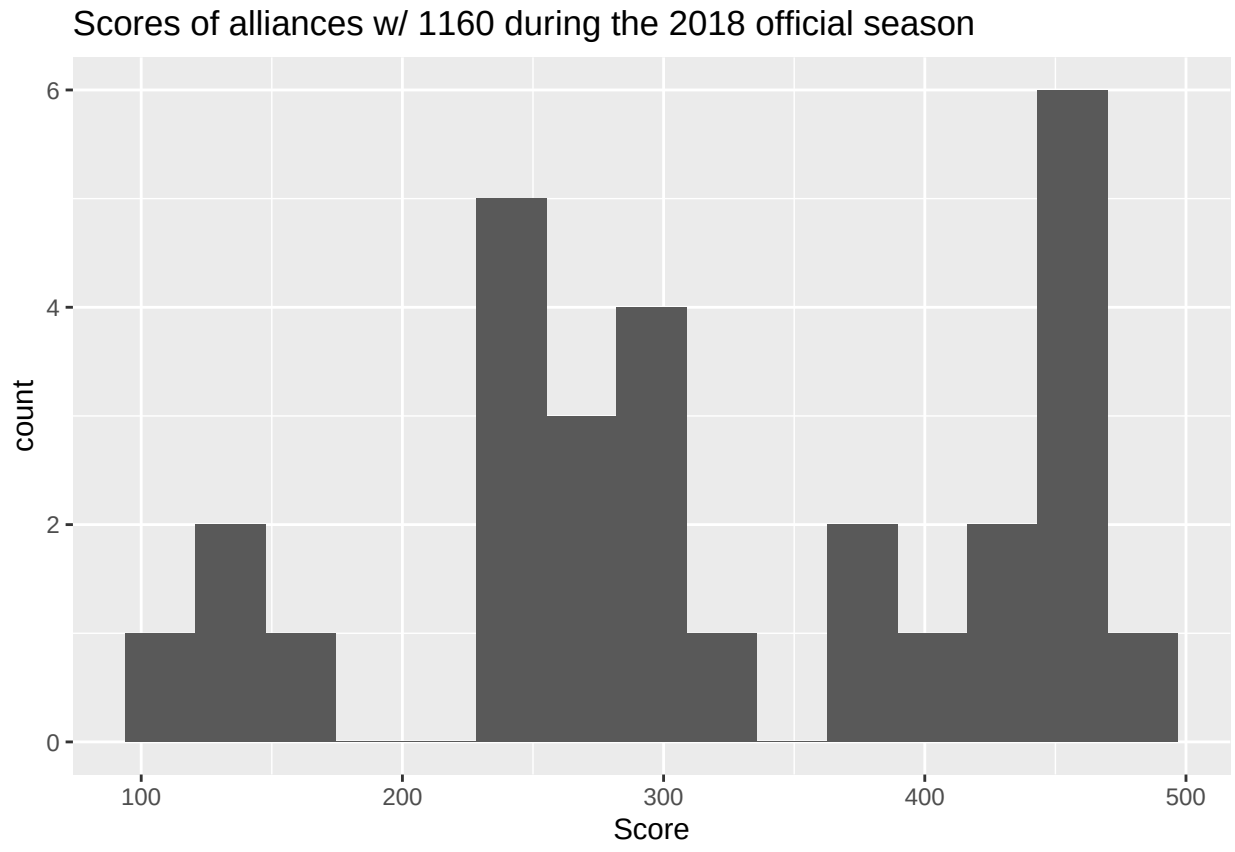
```
## Warning: package 'rstanarm' was built under R version 4.0.3
## Loading required package: Rcpp
## This is rstanarm version 2.21.1
## - See https://mc-stan.org/rstanarm/articles/priors for changes to default priors!
## - Default priors may change, so it's safest to specify priors, even if equivalent to the defaults.
## - For execution on a local, multicore CPU with excess RAM we recommend calling
##   options(mc.cores = parallel::detectCores())
```

```
library(loo)
```

```
## Warning: package 'loo' was built under R version 4.0.3
## This is loo version 2.3.1
## - Online documentation and vignettes at mc-stan.org/loo
```

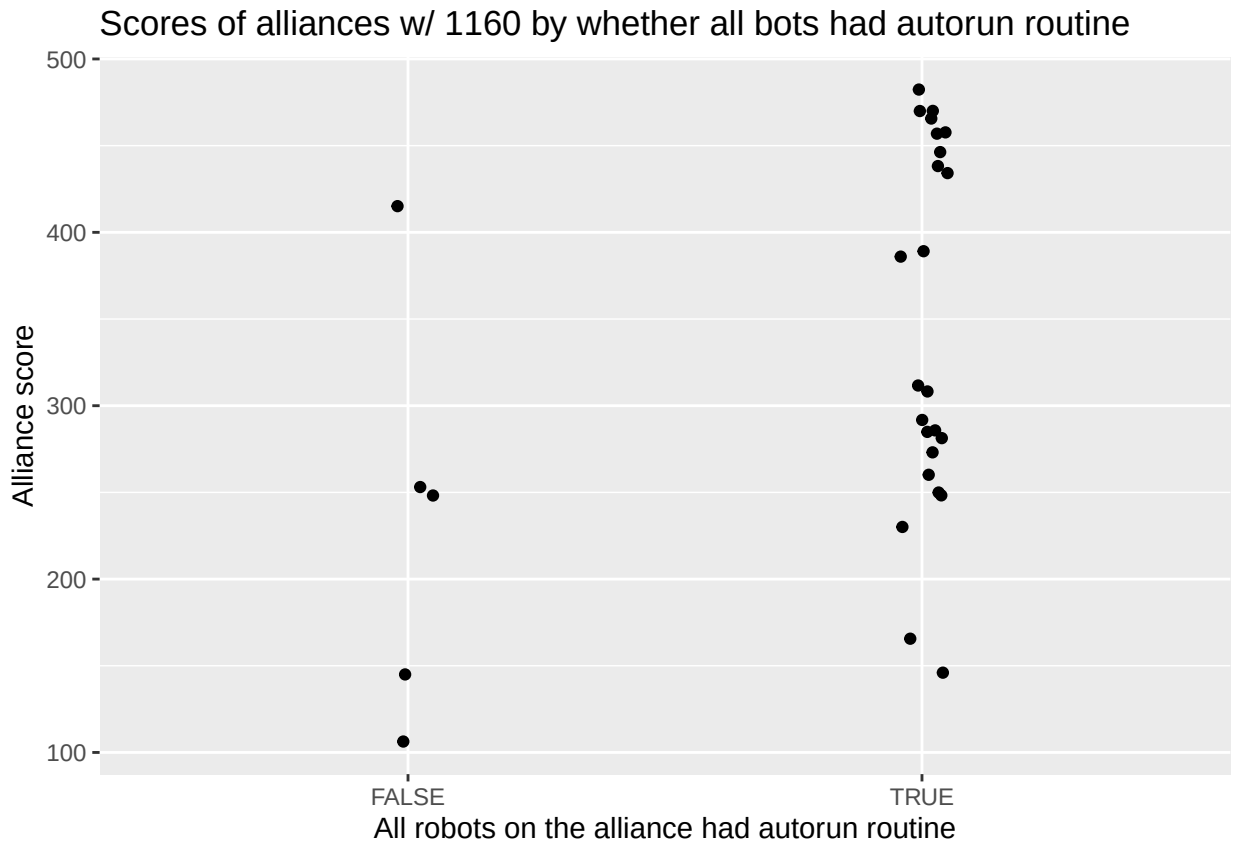
```
## - As of v2.0.0 loo defaults to 1 core but we recommend using as many as possible. Use the 'cores' ar
## - Windows 10 users: loo may be very slow if 'mc.cores' is set in your .Rprofile file (see https://gi
library(bayesplot)
## Warning: package 'bayesplot' was built under R version 4.0.3
## This is bayesplot version 1.7.2
## - Online documentation and vignettes at mc-stan.org/bayesplot
## - bayesplot theme set to bayesplot::theme_default()
##    * Does _not_ affect other ggplot2 plots
##    * See ?bayesplot_theme_set for details on theme setting
library(MatchIt)
## Warning: package 'MatchIt' was built under R version 4.0.3
library(cobalt)
## Warning: package 'cobalt' was built under R version 4.0.3
##  cobalt (Version 4.2.4, Build Date: 2020-11-05 17:30:21 UTC)
##
## Attaching package: 'cobalt'
## The following object is masked from 'package:MatchIt':
##
##     lalonde
library(survey)
## Warning: package 'survey' was built under R version 4.0.3
## Loading required package: grid
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
## Loading required package: survival
##
## Attaching package: 'survey'
## The following object is masked from 'package:graphics':
##
##     dotchart
library(broom)

options(mc.cores=parallel::detectCores())

# first, we look at just data w/ my old FRC team (1160) from 2018
frc1160 <- read.csv("frc1160_2018.csv") %>% as_tibble(.)
```

```r
# filtering data - exclude beach blitz, as it reported incomplete data
frc1160_official <- filter(frc1160, event_key != "2018cabl")
frc1160_official %<>% mutate(blue = alliances_blue_team_keys_0 == "frc1160" |
                               alliances_blue_team_keys_1 == "frc1160" |
                               alliances_blue_team_keys_2 == "frc1160",
                             alliance_score = alliances_blue_score * as.integer(blue) +
                               alliances_red_score * as.integer(!blue),
                             blue_autorun_all = score_breakdown_blue_autoRobot1 == "AutoRun" &
                               score_breakdown_blue_autoRobot2 == "AutoRun" &
                               score_breakdown_blue_autoRobot3 == "AutoRun",
                             red_autorun_all = score_breakdown_red_autoRobot1 == "AutoRun" &
                               score_breakdown_red_autoRobot2 == "AutoRun" &
                               score_breakdown_red_autoRobot3 == "AutoRun",
                             alliance_autorun_all = blue_autorun_all * as.integer(blue) +
                               red_autorun_all * as.integer(!blue))

# preliminary graphs
ggplot(data = frc1160_official) +
  geom_histogram(mapping = aes(x = alliance_score), bins = 15) +
  labs(
    x = paste("Score"),
    title = paste("Scores of alliances w/ 1160 during the 2018 official season")
  )
```

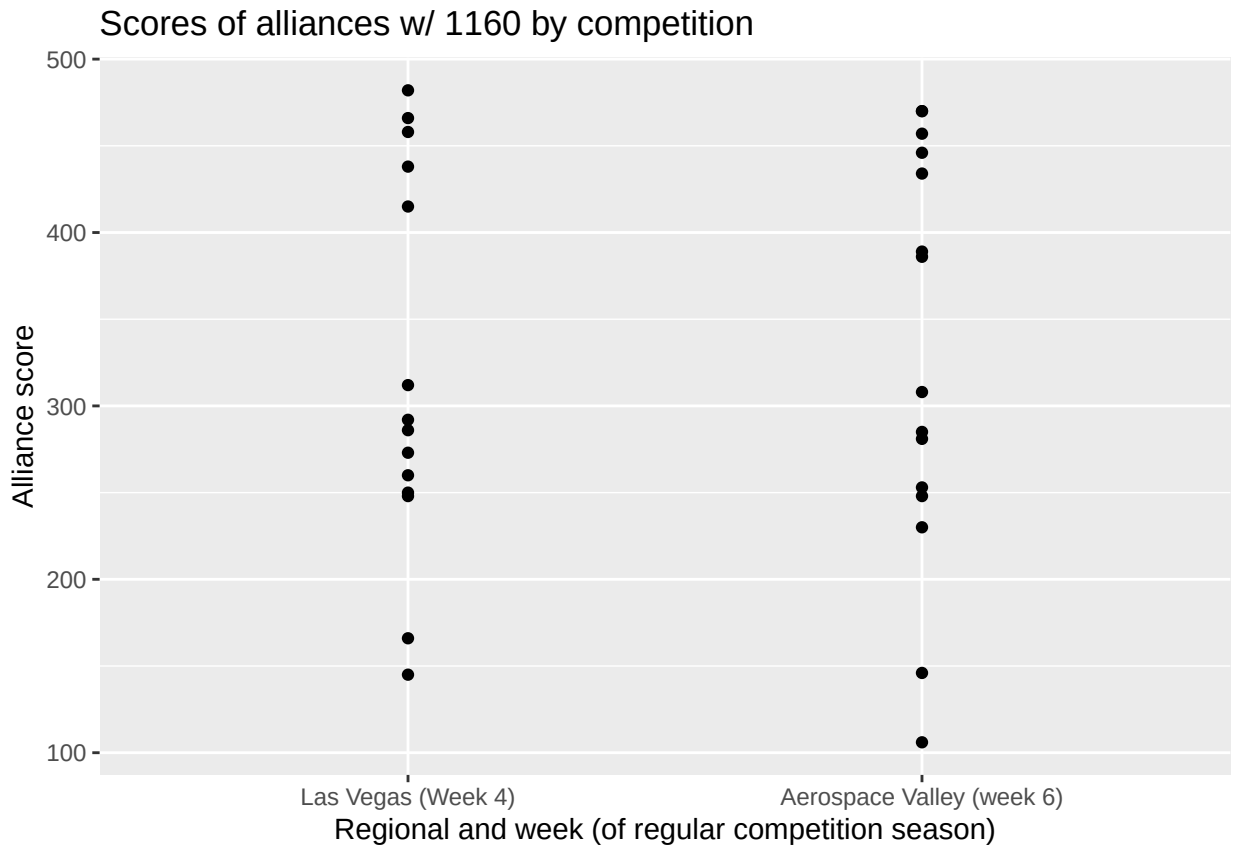



```r
ggplot(data = frc1160_official) +
  geom_jitter(width = 0.05,
```

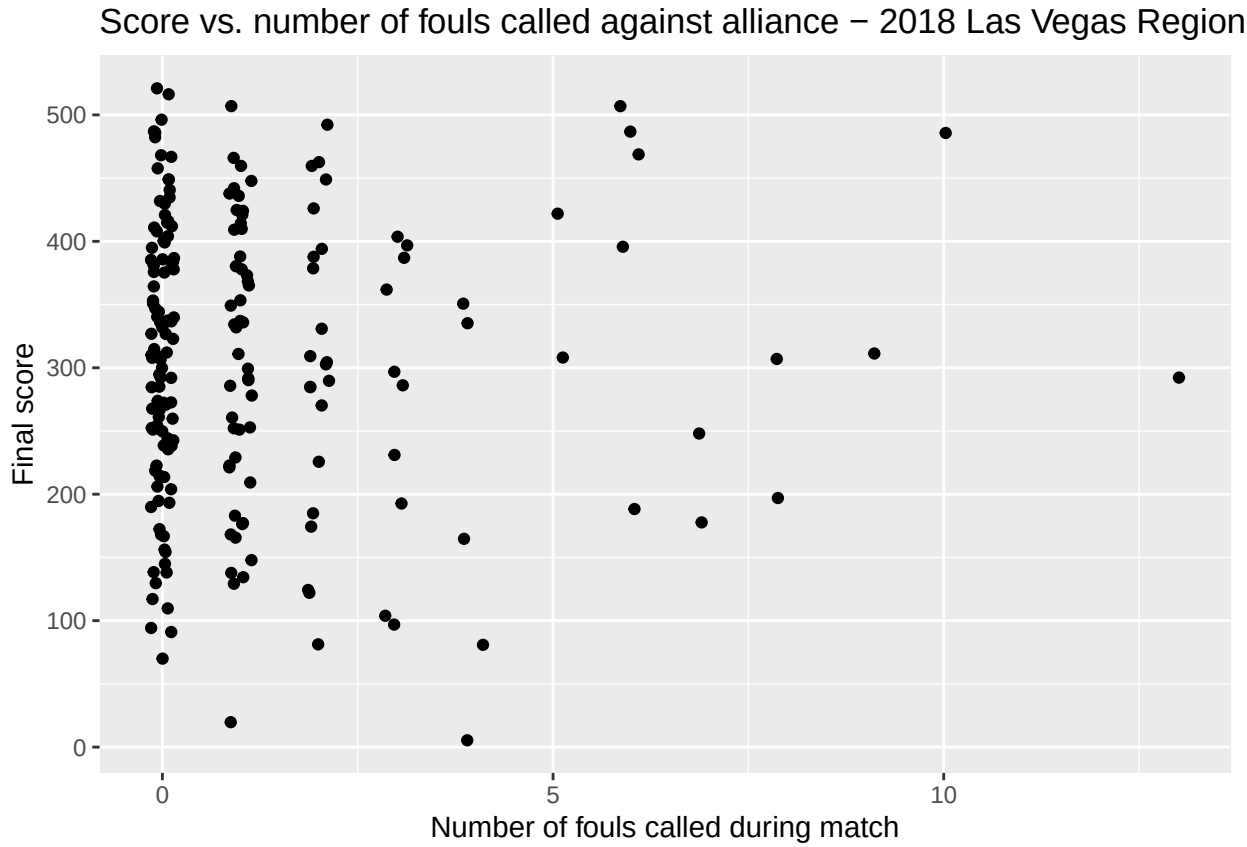
```
            mapping = aes(x = as.logical(alliance_autorun_all), y = alliance_score)) +
  labs(
    x = paste("All robots on the alliance had autorun routine"),
    y = paste("Alliance score"),
    title = paste("Scores of alliances w/ 1160 by whether all bots had autorun routine")
  )
```




```
# I would've plotted this data as a histogram (or two), but we really don't have enough data
ggplot(data = frc1160_official) +
  geom_point(mapping = aes(x = as.logical(event_key == "2018caav"),
                           y = alliance_score)) +
  scale_x_discrete(labels = c("Las Vegas (Week 4)", "Aerospace Valley (week 6)")) +
  labs(
    x = paste("Regional and week (of regular competition season)"),
    y = paste("Alliance score"),
    title = paste("Scores of alliances w/ 1160 by competition")
  )
```




```
# next, we look at all match data from the 2018 Las Vegas regional
lv <- read.csv("las_vegas_2018.csv") %>% as_tibble(.)

# more preliminary graphs -- foul count
ggplot(data = lv) +
  geom_jitter(width = 0.15,
              mapping = aes(x = score_breakdown_red_foulCount, y = alliances_red_score)) +
  geom_jitter(width = 0.15,
              mapping = aes(x = score_breakdown_blue_foulCount, y = alliances_blue_score)) +
  labs(
    x = paste("Number of fouls called during match"),
    y = paste("Final score"),
    title = paste("Score vs. number of fouls called against alliance - 2018 Las Vegas Regional")
  )
```




```
teams <- data.frame(team = unique(c(unique(lv$alliances_blue_team_keys_0),
                           unique(lv$alliances_blue_team_keys_1),
                           unique(lv$alliances_blue_team_keys_2),
                           unique(lv$alliances_red_team_keys_0),
                           unique(lv$alliances_red_team_keys_1),
                           unique(lv$alliances_red_team_keys_2))))

teams_blue <- data.frame(team = unique(c(unique(lv$alliances_blue_team_keys_0),
                                         unique(lv$alliances_blue_team_keys_1),
                                         unique(lv$alliances_blue_team_keys_2))))
lv_mod <- lv
for (i in teams_blue$team) {
  varname <- as.name(paste(i))
  lv_mod <- mutate(lv_mod, !! varname :=
                     as.integer(alliances_blue_team_keys_0 == varname |
                                alliances_blue_team_keys_1 == varname |
                                alliances_blue_team_keys_2 == varname))
}

f <- paste(teams_blue$team[order(teams_blue)][1:20], collapse = ' + ')
f <- paste("alliances_blue_score", f, sep = ' ~ ')
f <- as.formula(f)

fit <- stan_glm(formula = f, data = lv_mod, refresh = 0, cores = 10)

teams_red <- data.frame(team = unique(c(unique(lv$alliances_red_team_keys_0),
```

```
                                        unique(lv$alliances_red_team_keys_1),
                                        unique(lv$alliances_red_team_keys_2))))
for (i in teams_red$team) {
  varname <- as.name(paste(i))
  lv_mod <- mutate(lv_mod, !! varname :=
                     as.integer(alliances_red_team_keys_0 == varname |
                                alliances_red_team_keys_1 == varname |
                                alliances_red_team_keys_2 == varname))
}

f2 <- paste(teams_red$team[order(teams_red)][1:10], collapse = ' + ')
f2 <- paste("alliances_red_score", f2, sep = ' ~ ')
f2 <- as.formula(f2)

fit2 <- stan_glm(formula = f2, data = lv_mod, refresh = 0, cores = 10)
print(fit2)
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      alliances_red_score ~ frc1160 + frc1388 + frc2429 + frc2485 +
##     frc2543 + frc2647 + frc2659 + frc2710 + frc3009 + frc3011
##  observations: 103
##  predictors:   11
## ------
##             Median MAD_SD
## (Intercept)  287.9   14.3
## frc1160      -18.3   50.3
## frc1388       42.0   48.9
## frc2429      -47.6   43.2
## frc2485      142.3   43.7
## frc2543       53.1   43.4
## frc2647     -105.5   46.0
## frc2659       21.4   46.6
## frc2710        5.0   37.5
## frc3009       55.6   46.7
## frc3011     -115.9   43.6
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 108.1    7.9
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

```
f3 <- paste(teams_red$team[order(teams_red)][11:20], collapse = ' + ')
f3 <- paste("alliances_red_score", f3, sep = ' ~ ')
f3 <- as.formula(f3)

fit3 <- stan_glm(formula = f3, data = lv_mod, refresh = 0, cores = 10)
print(fit3)
```

```
## stan_glm
##  family:       gaussian [identity]
```

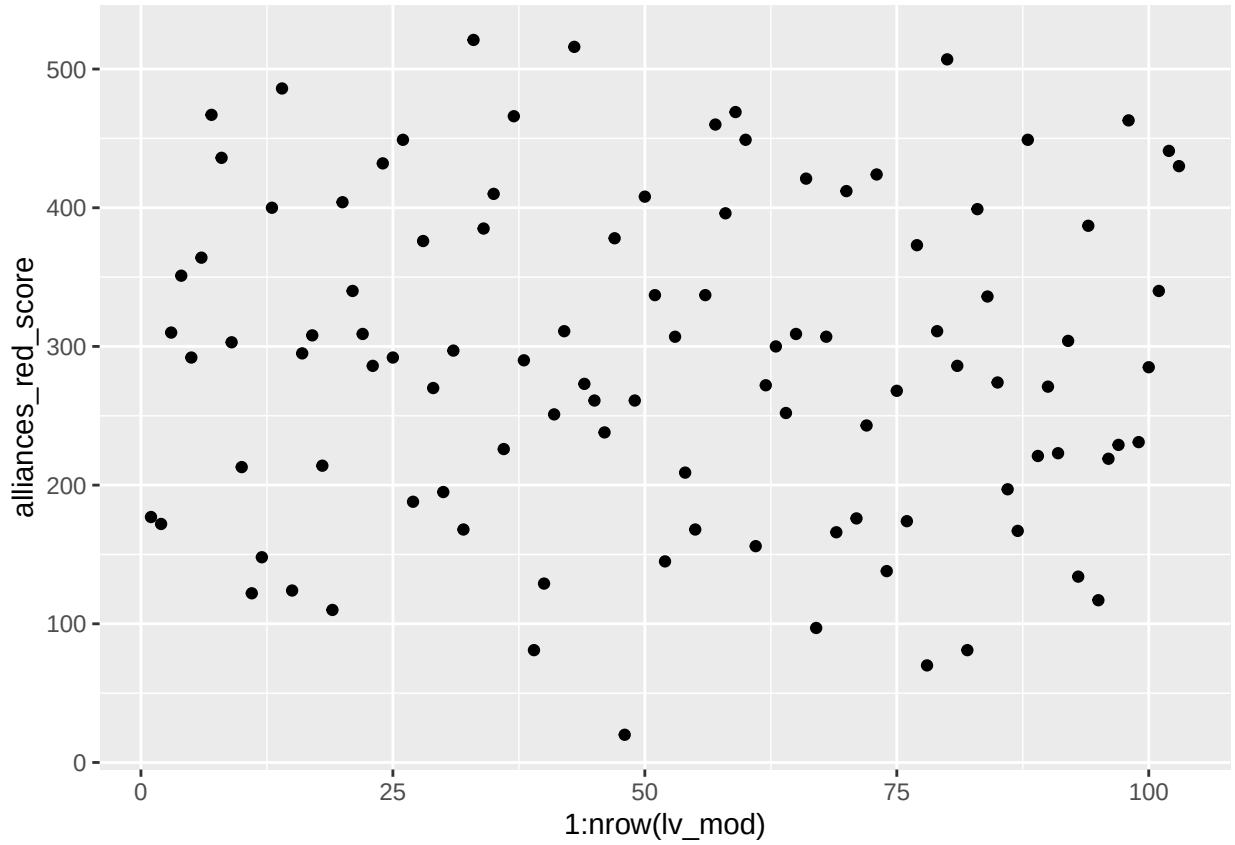
```
##  formula:      alliances_red_score ~ frc3021 + frc3255 + frc3495 + frc3577 +
##     frc3965 + frc399 + frc4 + frc4486 + frc4501 + frc4738
##  observations: 103
##  predictors:   11
## ------
##             Median MAD_SD
## (Intercept)  274.3   15.7
## frc3021       -5.8   40.0
## frc3255      106.0   41.8
## frc3495       27.9   42.1
## frc3577      -42.7   48.8
## frc3965       85.5   37.0
## frc399        15.1   37.2
## frc4          19.1   43.0
## frc4486       11.2   47.1
## frc4501     -117.4   47.0
## frc4738       34.8   42.1
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 110.8    8.1
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

```
f4 <- paste(teams_red$team[order(teams_red)][21:30], collapse = ' + ')
f4 <- paste("alliances_red_score", f4, sep = ' ~ ')
f4 <- as.formula(f4)

fit4 <- stan_glm(formula = f4, data = lv_mod, refresh = 0, cores = 10)
print(fit4)
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      alliances_red_score ~ frc4792 + frc5012 + frc5025 + frc5049 +
##     frc5059 + frc5285 + frc5429 + frc585 + frc5851 + frc5875
##  observations: 103
##  predictors:   11
## ------
##             Median MAD_SD
## (Intercept) 299.0   15.3
## frc4792     -86.9   48.9
## frc5012      58.4   48.1
## frc5025     -62.3   49.3
## frc5049     -44.6   51.8
## frc5059      54.3   54.2
## frc5285      44.4   39.8
## frc5429     -18.7   50.0
## frc585        2.5   48.8
## frc5851     -36.3   51.0
## frc5875     -93.3   50.0
##
## Auxiliary parameter(s):
##       Median MAD_SD
```

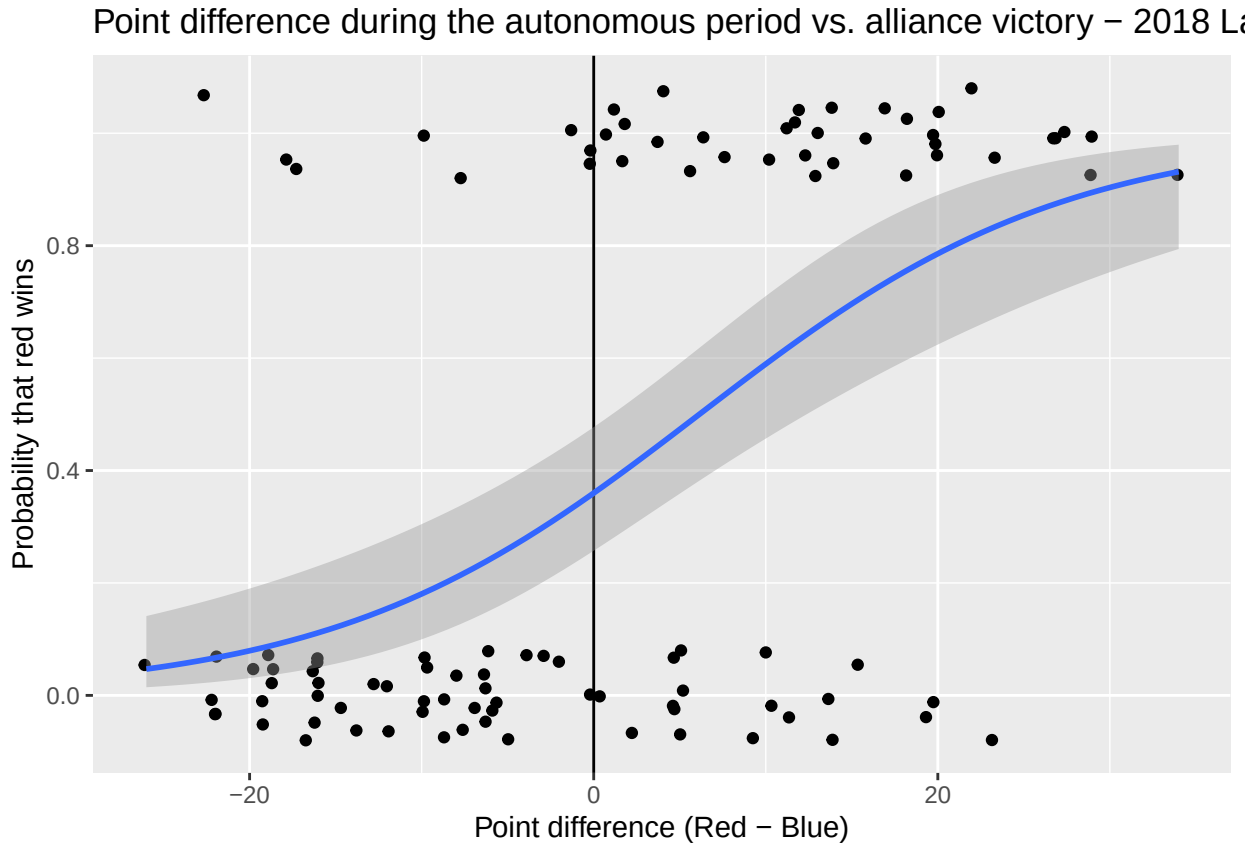
```
## sigma 114.2    8.5
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

```
# potentially unused - errors in predictions

lv_mod$predicted_red_score_1 <- colMeans(posterior_predict(fit, data = lv_mod))
lv_mod$error_red_score_1 <- lv_mod$alliances_red_score - lv_mod$predicted_red_score_1

lv_mod$predicted_red_score_2 <- colMeans(posterior_predict(fit2, data = lv_mod))
lv_mod$error_red_score_2 <- lv_mod$alliances_red_score - lv_mod$predicted_red_score_2

lv_mod$predicted_red_score_3 <- colMeans(posterior_predict(fit3, data = lv_mod))
lv_mod$error_red_score_3 <- lv_mod$alliances_red_score - lv_mod$predicted_red_score_3

lv_mod$predicted_red_score_4 <- colMeans(posterior_predict(fit4, data = lv_mod))
lv_mod$error_red_score_4 <- lv_mod$alliances_red_score - lv_mod$predicted_red_score_4

ggplot(data = lv_mod) +
  geom_point(mapping = aes(x = 1:nrow(lv_mod), y = alliances_red_score))
```



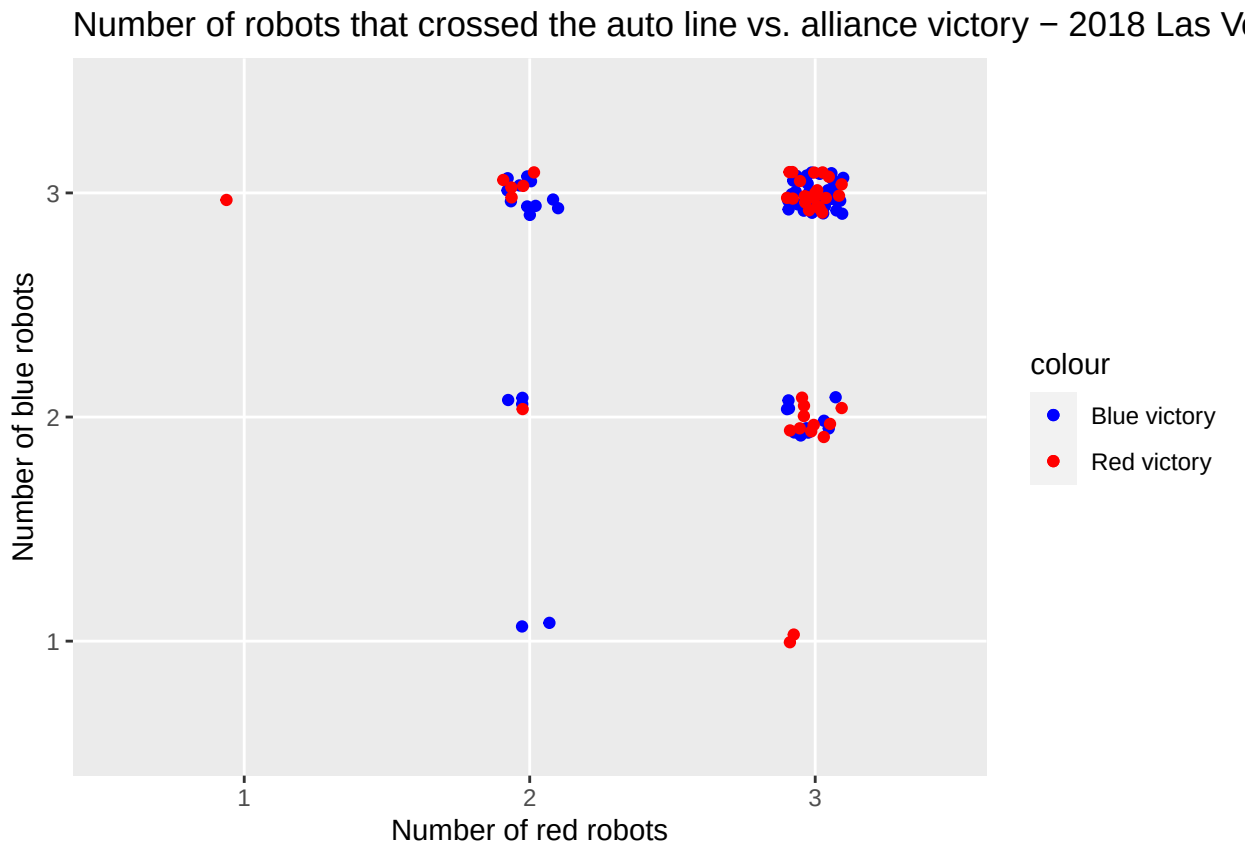

```
# correlation between difference in autoPoints (red - blue) and victory probability?
lv_mod %<>% mutate(difference_auto = score_breakdown_red_autoPoints -
                     score_breakdown_blue_autoPoints,
                   red_win = as.integer(winning_alliance == "red"),
```

```
                  red_autorun = as.integer(score_breakdown_red_autoRobot1 == 'AutoRun') +
                    as.integer(score_breakdown_red_autoRobot2 == 'AutoRun') +
                    as.integer(score_breakdown_red_autoRobot3 == 'AutoRun'),
                  blue_autorun = as.integer(score_breakdown_blue_autoRobot1 == 'AutoRun') +
                    as.integer(score_breakdown_blue_autoRobot2 == 'AutoRun') +
                    as.integer(score_breakdown_blue_autoRobot3 == 'AutoRun'),
                  difference_autorun = red_autorun - blue_autorun,
                  red_climb = as.integer(score_breakdown_red_endgameRobot1 == 'Climbing') +
                    as.integer(score_breakdown_red_endgameRobot2 == 'Climbing') +
                    as.integer(score_breakdown_red_endgameRobot3 == 'Climbing'),
                  blue_climb = as.integer(score_breakdown_blue_endgameRobot1 == 'Climbing') +
                    as.integer(score_breakdown_blue_endgameRobot2 == 'Climbing') +
                    as.integer(score_breakdown_blue_endgameRobot3 == 'Climbing'),
                  difference_climb = red_climb - blue_climb)

fit5 <- stan_glm(red_win ~ difference_auto, family = binomial(link = 'logit'),
                 data = lv_mod, refresh = 0)
lv_mod$fit5_prob <- colMeans(posterior_predict(fit5))
lv_mod$fit5_pred <- round(lv_mod$fit5_prob)

ggplot(data = lv_mod) +
  geom_jitter(height = 0.08,
              mapping = aes(x = difference_auto, y = red_win)) +
  #geom_line(mapping = aes(x = difference_auto, y = fit5_prob)) +
  geom_vline(xintercept = 0) +
  geom_smooth(formula = y ~ x,
              mapping = aes(x = difference_auto, y = red_win),
              method = "glm", method.args = list(family = 'binomial'), se = TRUE) +
  labs(
    x = paste("Point difference (Red - Blue)"),
    y = paste("Probability that red wins"),
    title = paste("Point difference during the autonomous period vs. alliance victory - 2018 Las Vegas
  )
```

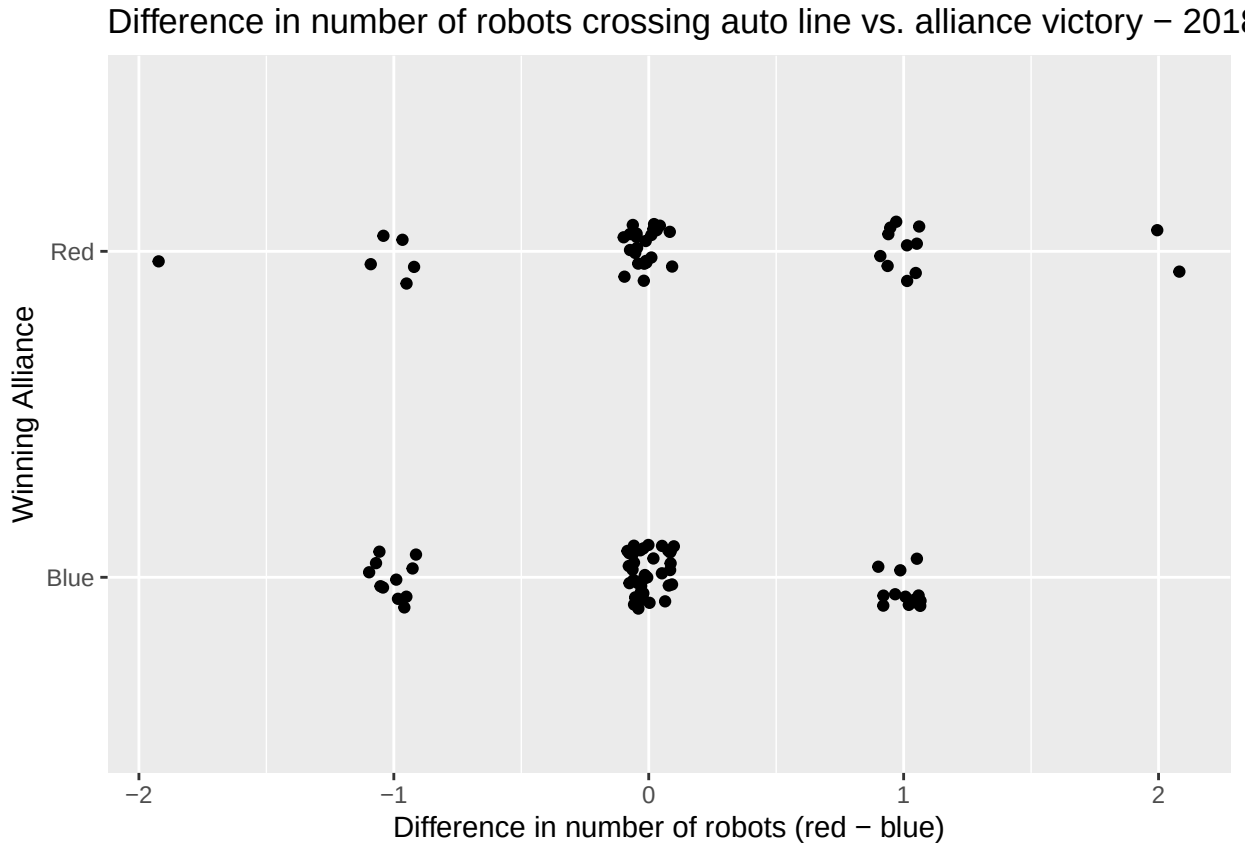



```
# okay, now let's try adding more predictors

fit6 <- stan_glm(red_win ~ difference_auto + difference_autorun + difference_climb,
                 family = binomial(link = 'logit'), data = lv_mod, refresh = 0)

# version without differences
fit7 <- stan_glm(red_win ~ difference_auto + red_autorun + blue_autorun + red_climb +
                   blue_climb,
                 family = binomial(link = 'logit'), data = lv_mod, refresh = 0)

# blue, red autorun vs. victor - exploratory plot
ggplot() +
  geom_jitter(width = 0.1, height = 0.1,
              data = filter(lv_mod, winning_alliance == 'blue'),
              mapping = aes(x = factor(red_autorun),
                            y = factor(blue_autorun),
                            color = "Blue victory")) +
  geom_jitter(width = 0.1, height = 0.1,
              data = filter(lv_mod, winning_alliance == 'red'),
              mapping = aes(x = factor(red_autorun),
                            y = factor(blue_autorun),
                            color = "Red victory")) +
  labs(
    x = paste("Number of red robots"),
    y = paste("Number of blue robots"),
    title = paste("Number of robots that crossed the auto line vs. alliance victory - 2018 Las Vegas Re
```

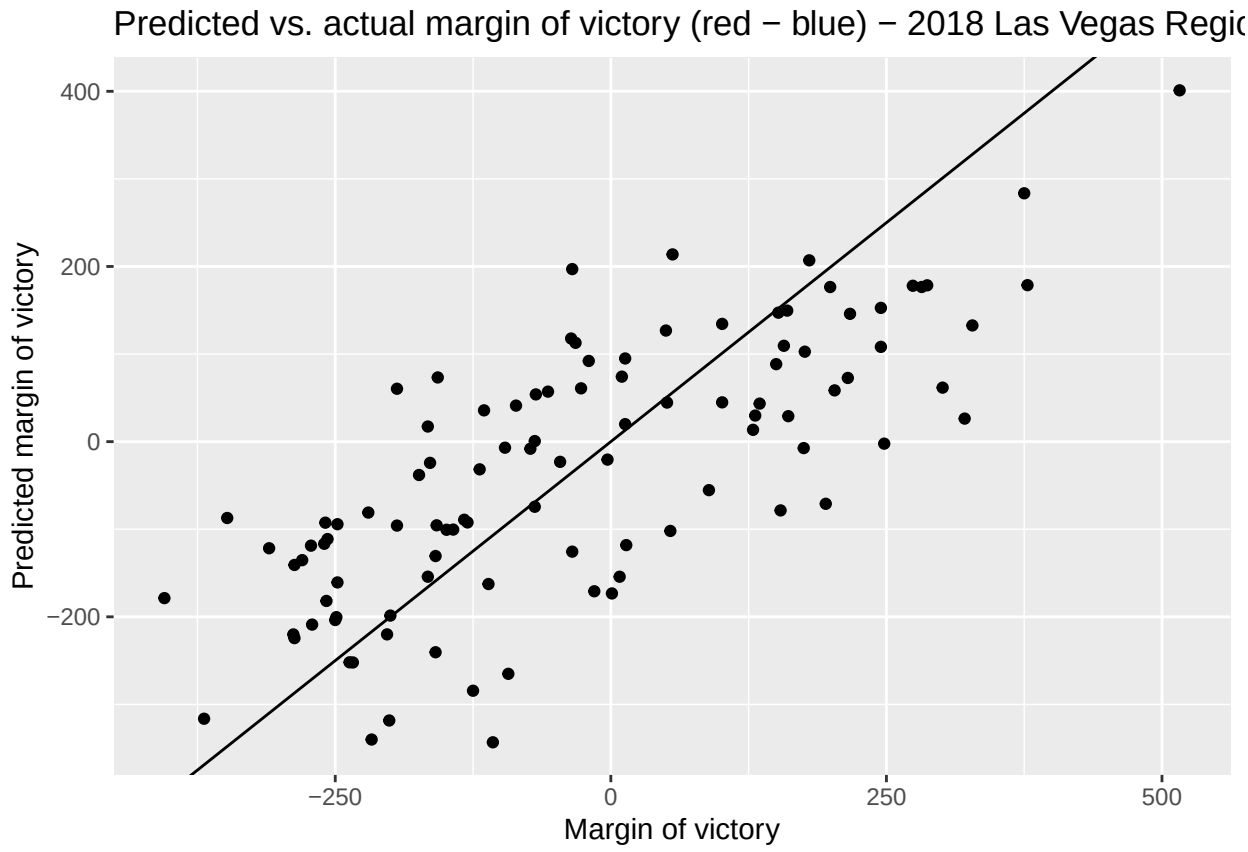
```
  ) +
  scale_color_manual(values = c("blue", "red"))
```



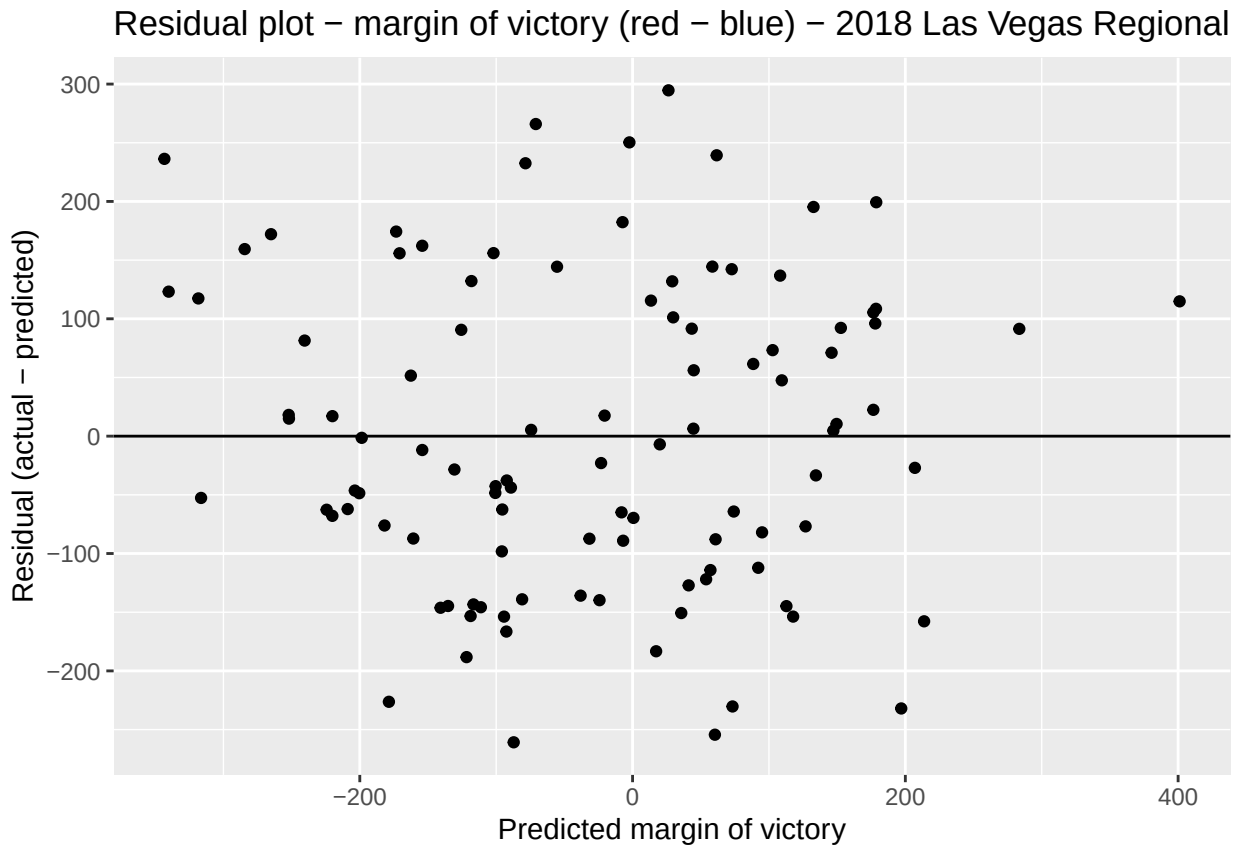

```
# maybe the difference thereof will reveal more insights
ggplot() +
  geom_jitter(width = 0.1, height = 0.1,
              data = filter(lv_mod, winning_alliance == 'red'),
              mapping = aes(x = difference_autorun, y = TRUE)) +
  geom_jitter(width = 0.1, height = 0.1,
              data = filter(lv_mod, winning_alliance == 'blue'),
              mapping = aes(x = difference_autorun, y = FALSE)) +
  labs(
    x = paste("Difference in number of robots (red - blue)"),
    y = paste("Winning Alliance"),
    title = paste("Difference in number of robots crossing auto line vs. alliance victory - 2018 Las Veg
  ) +
  scale_color_manual(values = c("blue", "red")) +
  scale_y_discrete(labels = c("Blue", "Red"))
```




```
lv_mod %<>% mutate(blue_cubes = score_breakdown_blue_vaultPoints / 5,
                   red_cubes = score_breakdown_red_vaultPoints / 5,
                   difference_score = score_breakdown_red_totalPoints -
                     score_breakdown_blue_totalPoints)
fit8 <- stan_glm(red_win ~ red_autorun + blue_autorun +
                   red_climb + blue_climb + blue_cubes + red_cubes,
                 family = binomial(link = 'logit'), data = lv_mod, refresh = 0)
# not a logistic regression - margin of victory
fit9 <- stan_glm(difference_score ~ red_autorun + blue_autorun +
                   red_climb + blue_climb + blue_cubes + red_cubes,
                 data = lv_mod, refresh = 0)
lv_mod$difference_score_predicted <- colMeans(posterior_predict(fit9))
ggplot(data = lv_mod) +
  geom_point(mapping = aes(x = difference_score, y = difference_score_predicted)) +
  geom_abline(slope = 1, intercept = 0) +
  labs(
    x = paste("Margin of victory"),
    y = paste("Predicted margin of victory"),
    title = paste("Predicted vs. actual margin of victory (red - blue) - 2018 Las Vegas Regional")
  )
```

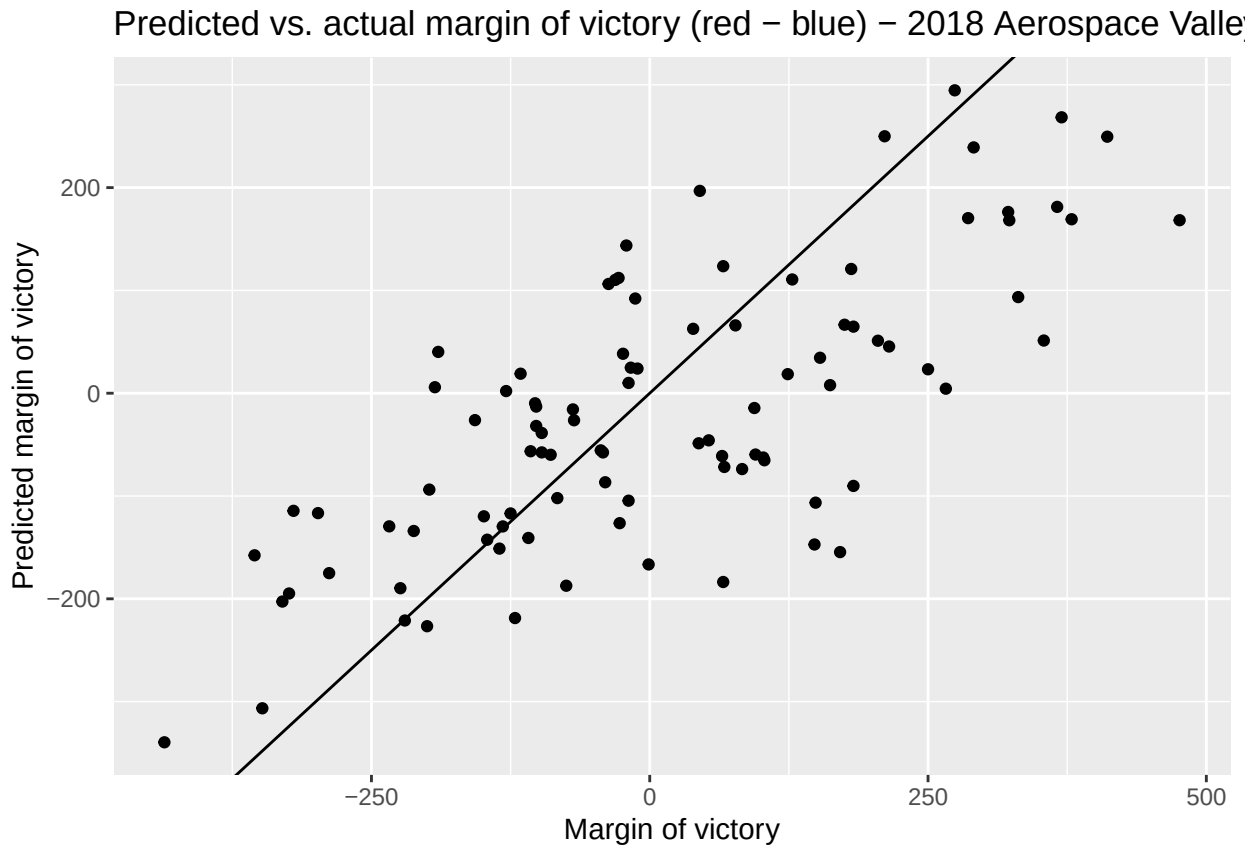



```
ggplot(data = lv_mod) +
  geom_point(mapping = aes(x = difference_score_predicted,
                           y = difference_score - difference_score_predicted)) +
  geom_abline(slope = 0, intercept = 0) +
  labs(
    x = paste("Predicted margin of victory"),
    y = paste("Residual (actual - predicted)"),
    title = paste("Residual plot - margin of victory (red - blue) - 2018 Las Vegas Regional")
  )
```

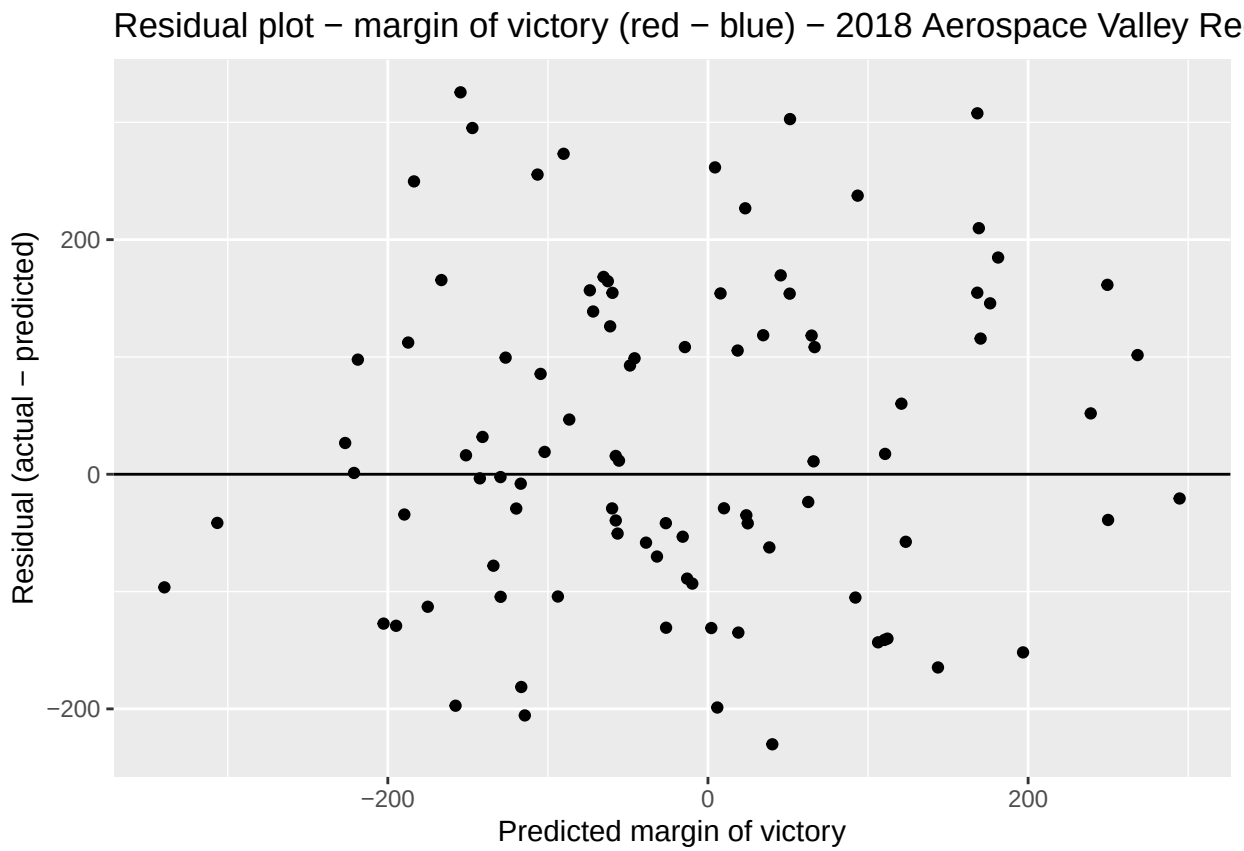



```
# logistic version of fit9
fit10 <- stan_glm(red_win ~ red_autorun + blue_autorun +
                    red_climb + blue_climb + blue_cubes + red_cubes,
                  family = binomial(link = 'logit'), data = lv_mod, refresh = 0)

# okay, so now we have a decently promising model. let's try it on some fresh data
# this is from the 2018 aerospace valley regional
av <- read.csv("aerospace_valley_2018.csv") %>% as_tibble(.)
av %<>% mutate(red_autorun = as.integer(score_breakdown_red_autoRobot1 == 'AutoRun') +
                 as.integer(score_breakdown_red_autoRobot2 == 'AutoRun') +
                 as.integer(score_breakdown_red_autoRobot3 == 'AutoRun'),
               blue_autorun = as.integer(score_breakdown_blue_autoRobot1 == 'AutoRun') +
                 as.integer(score_breakdown_blue_autoRobot2 == 'AutoRun') +
                 as.integer(score_breakdown_blue_autoRobot3 == 'AutoRun'),
               red_climb = as.integer(score_breakdown_red_endgameRobot1 == 'Climbing') +
                 as.integer(score_breakdown_red_endgameRobot2 == 'Climbing') +
                 as.integer(score_breakdown_red_endgameRobot3 == 'Climbing'),
               blue_climb = as.integer(score_breakdown_blue_endgameRobot1 == 'Climbing') +
                 as.integer(score_breakdown_blue_endgameRobot2 == 'Climbing') +
                 as.integer(score_breakdown_blue_endgameRobot3 == 'Climbing'),
               blue_cubes = score_breakdown_blue_vaultPoints / 5,
               red_cubes = score_breakdown_red_vaultPoints / 5,
               difference_score = score_breakdown_red_totalPoints -
                 score_breakdown_blue_totalPoints)
av$difference_score_predicted <- colMeans(posterior_predict(fit9, newdata = av))
```

```
ggplot(data = av) +
  geom_point(mapping = aes(x = difference_score, y = difference_score_predicted)) +
  geom_abline(slope = 1, intercept = 0) +
  labs(
    x = paste("Margin of victory"),
    y = paste("Predicted margin of victory"),
    title = paste("Predicted vs. actual margin of victory (red - blue) - 2018 Aerospace Valley Regional
  )
```

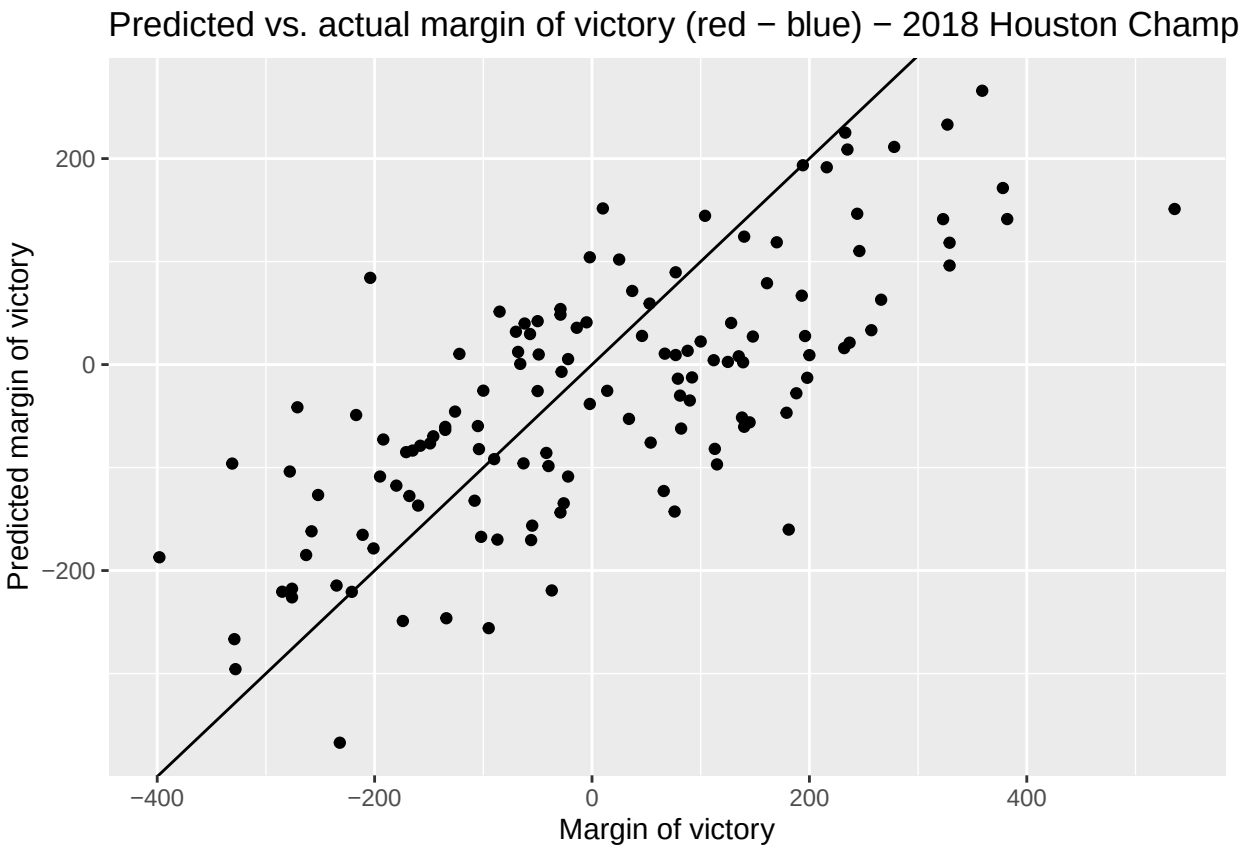



```
ggplot(data = av) +
  geom_point(mapping = aes(x = difference_score_predicted,
                           y = difference_score - difference_score_predicted)) +
  geom_abline(slope = 0, intercept = 0) +
  labs(
    x = paste("Predicted margin of victory"),
    y = paste("Residual (actual - predicted)"),
    title = paste("Residual plot - margin of victory (red - blue) - 2018 Aerospace Valley Regional")
  )
```

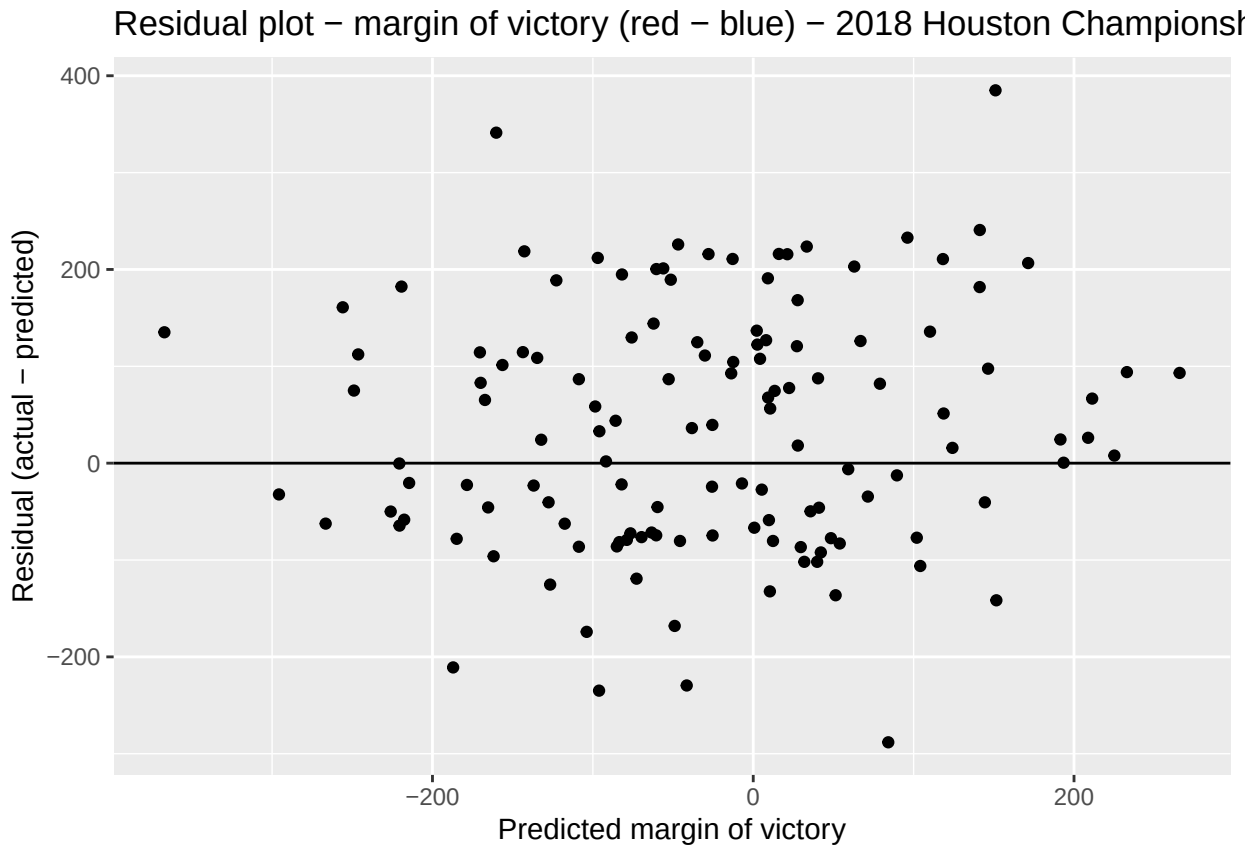



```
# another way of assessing fit-- accuracy metric
av %<>% mutate(accurate = as.integer(sign(difference_score) == sign(difference_score_predicted)))
av_accuracy <- sum(av$accurate) / nrow(av)

lv_mod %<>% mutate(accurate = as.integer(sign(difference_score) == sign(difference_score_predicted)))
lv_accuracy <- sum(lv_mod$accurate) / nrow(lv)

# more fresh data - hopper divison, houston champs 2018
hp <- read.csv("houston_hopper_2018.csv") %>% as_tibble(.)
hp %<>% mutate(red_autorun = as.integer(score_breakdown_red_autoRobot1 == 'AutoRun') +
                 as.integer(score_breakdown_red_autoRobot2 == 'AutoRun') +
                 as.integer(score_breakdown_red_autoRobot3 == 'AutoRun'),
               blue_autorun = as.integer(score_breakdown_blue_autoRobot1 == 'AutoRun') +
                 as.integer(score_breakdown_blue_autoRobot2 == 'AutoRun') +
                 as.integer(score_breakdown_blue_autoRobot3 == 'AutoRun'),
               red_climb = as.integer(score_breakdown_red_endgameRobot1 == 'Climbing') +
                 as.integer(score_breakdown_red_endgameRobot2 == 'Climbing') +
                 as.integer(score_breakdown_red_endgameRobot3 == 'Climbing'),
               blue_climb = as.integer(score_breakdown_blue_endgameRobot1 == 'Climbing') +
                 as.integer(score_breakdown_blue_endgameRobot2 == 'Climbing') +
                 as.integer(score_breakdown_blue_endgameRobot3 == 'Climbing'),
               blue_cubes = score_breakdown_blue_vaultPoints / 5,
               red_cubes = score_breakdown_red_vaultPoints / 5,
               difference_score = score_breakdown_red_totalPoints -
                 score_breakdown_blue_totalPoints)
hp$difference_score_predicted <- colMeans(posterior_predict(fit9, newdata = hp))
```

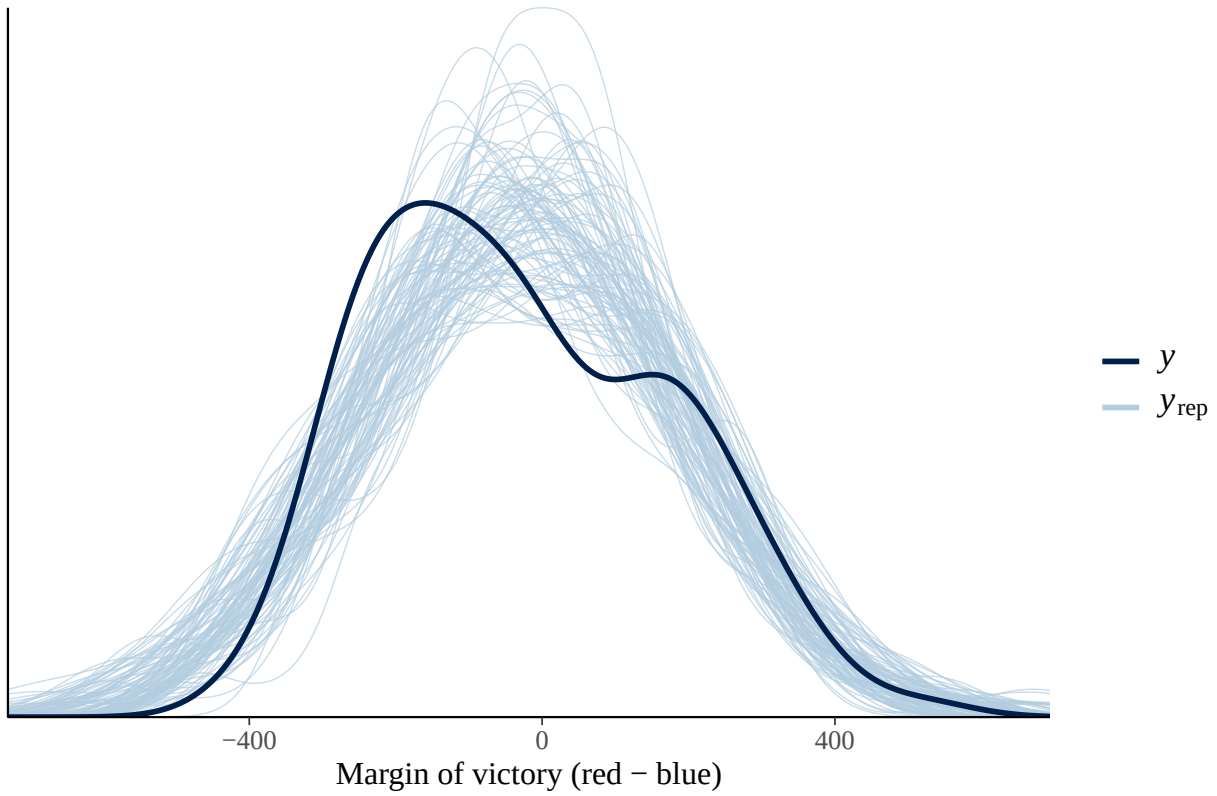
```
ggplot(data = hp) +
  geom_point(mapping = aes(x = difference_score, y = difference_score_predicted)) +
  geom_abline(slope = 1, intercept = 0) +
  labs(
    x = paste("Margin of victory"),
    y = paste("Predicted margin of victory"),
    title = paste("Predicted vs. actual margin of victory (red - blue) - 2018 Houston Championships, Ho
  )
```



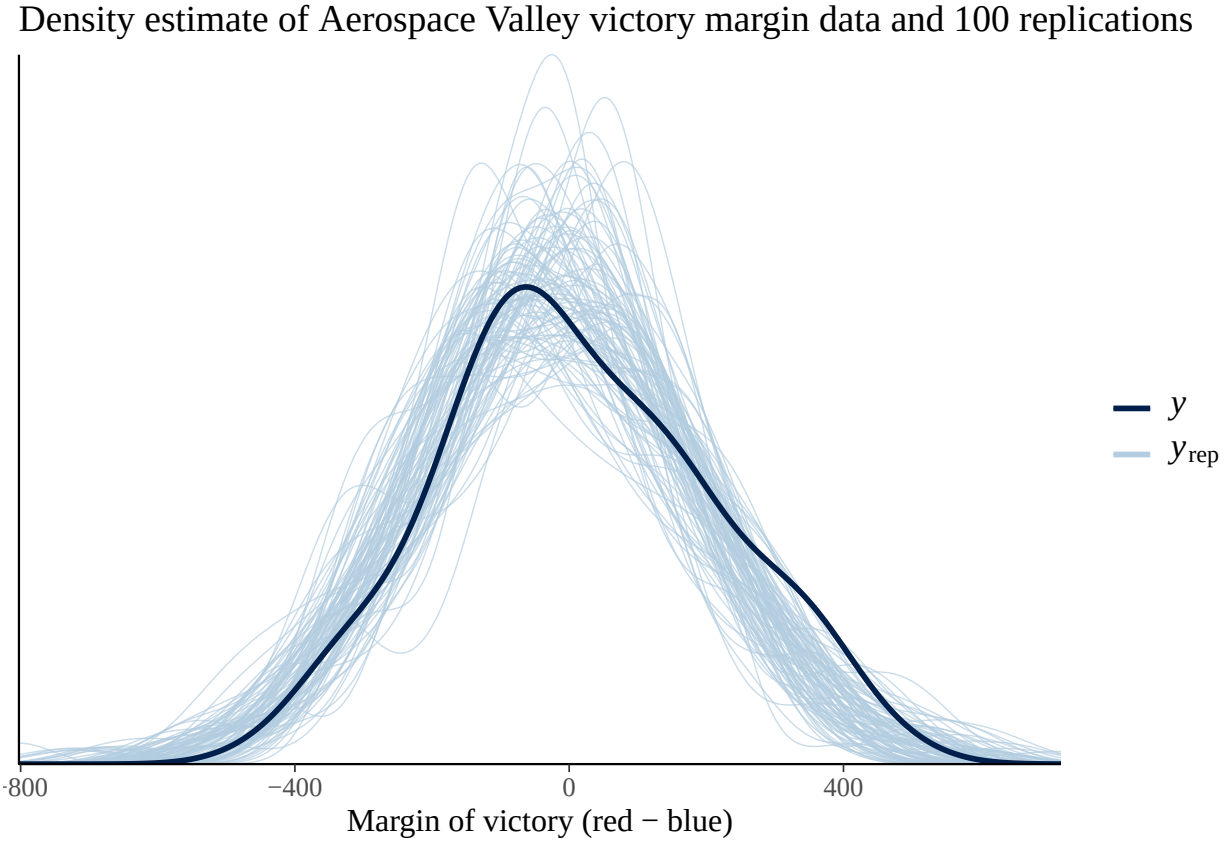

```
ggplot(data = hp) +
  geom_point(mapping = aes(x = difference_score_predicted,
                           y = difference_score - difference_score_predicted)) +
  geom_abline(slope = 0, intercept = 0) +
  labs(
    x = paste("Predicted margin of victory"),
    y = paste("Residual (actual - predicted)"),
    title = paste("Residual plot - margin of victory (red - blue) - 2018 Houston Championships, Hopper
  )
```




```
# hopper divison - accuracy
hp %<>% mutate(accurate = as.integer(sign(difference_score) == sign(difference_score_predicted)))
hp_accuracy <- sum(hp$accurate) / nrow(hp)

# posterior predictive checking - las vegas - distribution of victory margins
lv_margin_rep <- posterior_predict(fit9)
ppc_dens_overlay(lv_mod$difference_score, lv_margin_rep[1:100,]) +
  scale_y_continuous(breaks = NULL) +
  labs(title = paste("Density estimate of Las Vegas victory margin data and 100 replications"),
       x = paste("Margin of victory (red - blue)"))
```




```
# AV
av_margin_rep <- posterior_predict(fit9, newdata = av)
ppc_dens_overlay(av$difference_score, av_margin_rep[1:100,]) +
  scale_y_continuous(breaks = NULL) +
  labs(title = paste("Density estimate of Aerospace Valley victory margin data and 100 replications"),
       x = paste("Margin of victory (red - blue)"))
```




```
# Hopper
hp_margin_rep <- posterior_predict(fit9, newdata = hp)
ppc_dens_overlay(hp$difference_score, hp_margin_rep[1:100,]) +
  scale_y_continuous(breaks = NULL) +
  labs(title = paste("Density estimate of Hopper Division victory margin data and 100 replications"),
       x = paste("Margin of victory (red - blue)"))
```

Density estimate of Hopper Division victory margin data and 100 replications