homework8

Homework 8

(1) simulat the fake data

prepartion

```
print_file <- function(file) {
  cat(paste(readLines(file), "\n", sep=""), sep="")
}
library(rstan)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: StanHeaders
```

```
## rstan (Version 2.17.4, GitRev: 2e1f913d3ca3)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
```

```
options(mc.cores = parallel::detectCores())
rstan_options(auto_write = TRUE)
library(plyr)
```

```
## read the data and save them
library(plyr)
# Linear map of points to a score between -1 and 1
url_csv <- "http://www.football-data.co.uk/mmz4281/1516/E0.csv"
 # Data downloaded from football-data.co.uk
mydat <- read.csv(url(url_csv))
epl <- c()
# teams are assigned IDs 1, 2, ...:
epl$home_team <- as.numeric(mydat$HomeTeam)
epl$away_team <- as.numeric(mydat$AwayTeam)
epl$team_names <- levels(mydat$HomeTeam)
epl$home_goals <- mydat$FTHG # full time home goals
epl$away_goals <- mydat$FTAG # full time away goals
epl$score_diff <- epl$home_goals - epl$away_goals
# Points from last season are read and mapped to a score
epl$ngames <- length(epl$score_diff)
epl$nteams <- length(unique(epl$home_team))
epl$nweeks <- floor(2*epl$ngames /epl$nteams)
# The following code computes the week for each team in their games:
epl$home_week <- c()
epl$away_week <- c()
for (g in 1:epl$ngames) {
    epl$home_week[g] <- sum(epl$home_team[1:g] == epl$home_team[g]) + sum(epl$away_team[
1:g] == epl$home_team[g])
    epl$away_week[g] <- sum(epl$away_team[1:g] == epl$away_team[g]) + sum(epl$home_team[
1:g] == epl$away_team[g])
}
epl$bet_home <- mydat$B365H # Betting odds for home team win
epl$bet_draw <- mydat$B365D # Betting odds for draw
epl$bet_away <- mydat$B365A # Betting odds for away team win

# epl$prev_perf you can use estimates from the Team Abilities After Week One
epl$prev_perf <- c(0.5,1.2,0,0.6,-0.2,0,-0.2,-0.1,-0.3,0.4,0.6,-0.15,-0.4,
            0.01,-0.5,-1.5,-0.25,-0.8,-0.6,-0.5)

saveRDS(epl,'epl_data.rds')
```

modeling

```
fake_data <- stan_model('fakedata.stan')
data <- list(nteams = epl$nteams,
             ngames = epl$ngames,
             nweeks = epl$nweeks,
             home_week = epl$home_week,
             away_week = epl$away_week,
             home_team = epl$home_team,
             away_team = epl$away_team,
             prev_perf = epl$prev_perf)
simulations <- sampling(fake_data,data=data,algorithm = 'Fixed_param', seed = 123)
```

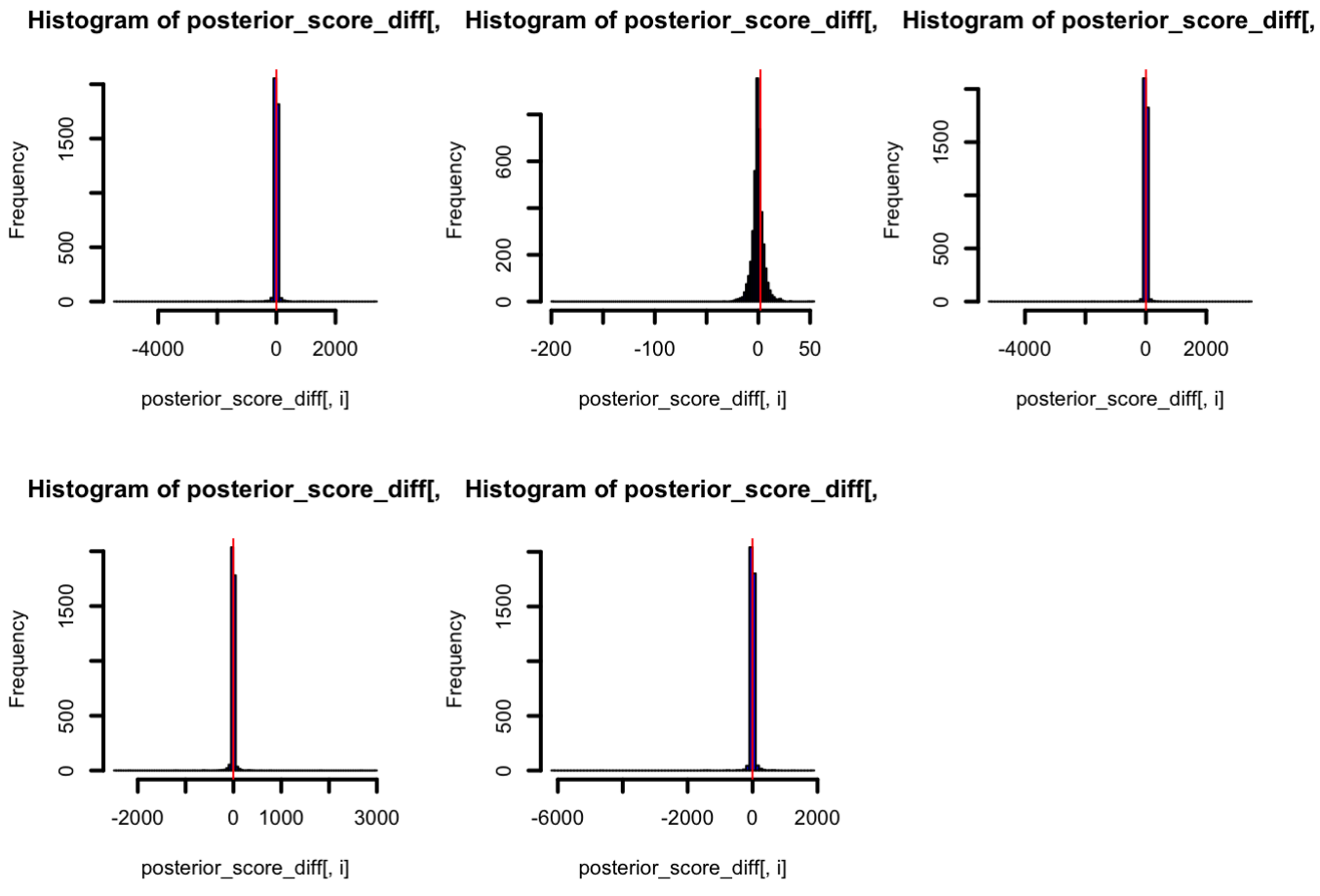
```
print_file('fakedata.stan')
```

```
## /*
## This is used to simulate the fake data.
## */
##
## data{
##   int<lower=2> nteams; // number of teams
##   int<lower=1> ngames; // number of games
##   int<lower=1> nweeks; // number of weeks
##   int<lower=1> home_week[ngames]; // week number for the home team
##   int<lower=1> away_week[ngames]; // week number for the away team
##   int<lower=1, upper=nteams> home_team[ngames]; // home team ID (1, ..., 20)
##   int<lower=1, upper=nteams> away_team[ngames]; // away team ID (1, ..., 20)
##   vector[nteams] prev_perf;// a score between -1 and 1
## }
## model{
## }
## generated quantities{
##   vector[ngames] score_diff; // home_goals - away_goals
##   real<lower=0> nu;
##   real<lower=0> tau_a;
##   real b_prev;
##   real<lower=0> sigma_a0;
##   real<lower=0> sigma_y;
##   vector[ngames] a_diff;
##   matrix[nweeks, nteams] a; // team abilities
##   row_vector<lower=0>[nteams] sigma_a; // game-to-game variation
##   row_vector<lower=0>[nteams] sigma_a_raw; // game-to-game variation
##   matrix[nweeks,nteams] eta_a;         // random component
##   real b_home; // the effect of hosting the game in mean of score_diff dist.
##
##   // priors
##   nu = fabs(gamma_rng(2,0.1));
##   b_prev = normal_rng(0,1);
##   sigma_a0 = fabs(normal_rng(0,1));
##   sigma_y = fabs(normal_rng(0,5));
##   b_home = normal_rng(0,1);
##   for (i in 1:nteams){
##     sigma_a_raw[i] = fabs(normal_rng(0,1));
##   }
##   tau_a = fabs(cauchy_rng(0,1));
##   for (i in 1:nweeks){
##     for (j in 1:nteams){
##       eta_a[i,j] = normal_rng(0,1);
##     }
##   }
##
##   // simulation the abilities matrix
##   for (j in 1:nteams){
##     a[1,j] = b_prev * prev_perf[j] + sigma_a0 * eta_a[1,j]; //initial abilities(at we
ek 1)
##   }
##   for(i in 1:nteams){
##       sigma_a[i] = fabs(tau_a * sigma_a_raw[i]);
```

```
##   }
##   for (w in 2:nweeks){
##     for (j in 1:nteams){
##     a[w,j] = a[(w-1),j] + sigma_a[j] * eta_a[w,j];  //evolution of abilities
##     }
##   }
##   // simulate the game results
##   for (g in 1:ngames){
##     a_diff[g] = a[home_week[g],home_team[g]] - a[away_week[g],away_team[g]];
##   }
##   for (g in 1:ngames)
##   score_diff[g] = round(student_t_rng(nu,a_diff[g] + b_home,sigma_y));
## }
```

check the coverage

```
## randomly pick five games to show the coverage
set.seed(4)
games <- sample(1:epl$ngames,5,replace = F)
posterior_score_diff <- as.matrix(simulations, pars = c('score_diff'))[,games]
true_score_diff <- epl$score_diff[games]
par(mfrow=c(2,3))
for (i in 1:5){
  hist(posterior_score_diff[,i],col = "blue", lwd = 2,breaks  = 100)
  abline(v=true_score_diff[i],col="red")
}
```



The coverage is not bad at least based on the random samples that i pick

(2) summary of the model

1 strenght

a. the author take the number of week into consideration which helps we to evaluate the ability change over the whole season.
b. use the historical data as a prior which incorporate more useful information

2 weekness

a. the set of some prior is not so reasonable, whether the prior is week or not is depand on the scale of the data. And these need more explaination.

b. the evalutaion of team ability is independ not from a overall change of the whole association. I personally perfer to use the hierarchical

c. the simulated data sometime would be very strange, since the likelihood based on the t distritbution. If you use the t distribution in the prior it tends to give a robust estimation but I am not sure whether it is reasonable to use such a heavy tail distribution for the likelihood.

(2) fit the model

```
fit_model <- stan_model('homework8.stan')
```

It takes too much time to run the whole 38 weeks. For simplicity, just run the 5 weeks to explain the main idea.

```
nsamples <- 1500
for (w in 1:5) {
  epl_w <- epl
  idx <- c(1:(w*10))
  epl_w$home_team <- epl$home_team[idx]
  epl_w$away_team <- epl$away_team[idx]
  epl_w$home_goals <- epl$home_goals[idx]
  epl_w$away_goals <- epl$away_goals[idx]
  epl_w$score_diff <- epl$score_diff[idx]
  epl_w$home_week <- epl$home_week[idx]
  epl_w$away_week <- epl$away_week[idx]
  epl_w$ngames <- w*10
  epl_w$nweeks <- max(c(epl_w$home_week, epl_w$away_week))
  fit <- sampling(fit_model, chains = 4, iter = (nsamples/2), data = epl_w)
  saveRDS(fit, paste("fit_", w, ".rds", sep=""))
}
```

check the model:

it have a lot of time steamps to pick and evaluate the result. But as we know, the time series tends to be more and more unstable. Thus, it would be reasonable to pick the last time steamps to check whether the model result is good.

```
# there are only 50 games in the first 5 weeks
test_result <- readRDS('fit_5.rds')
set.seed(4)
games <- sample(1:50,5,replace = F)
library(bayesplot)
```

```
## This is bayesplot version 1.6.0
```

```
## - Online documentation and vignettes at mc-stan.org/bayesplot
```

```
## - bayesplot theme set to bayesplot::theme_default()
```

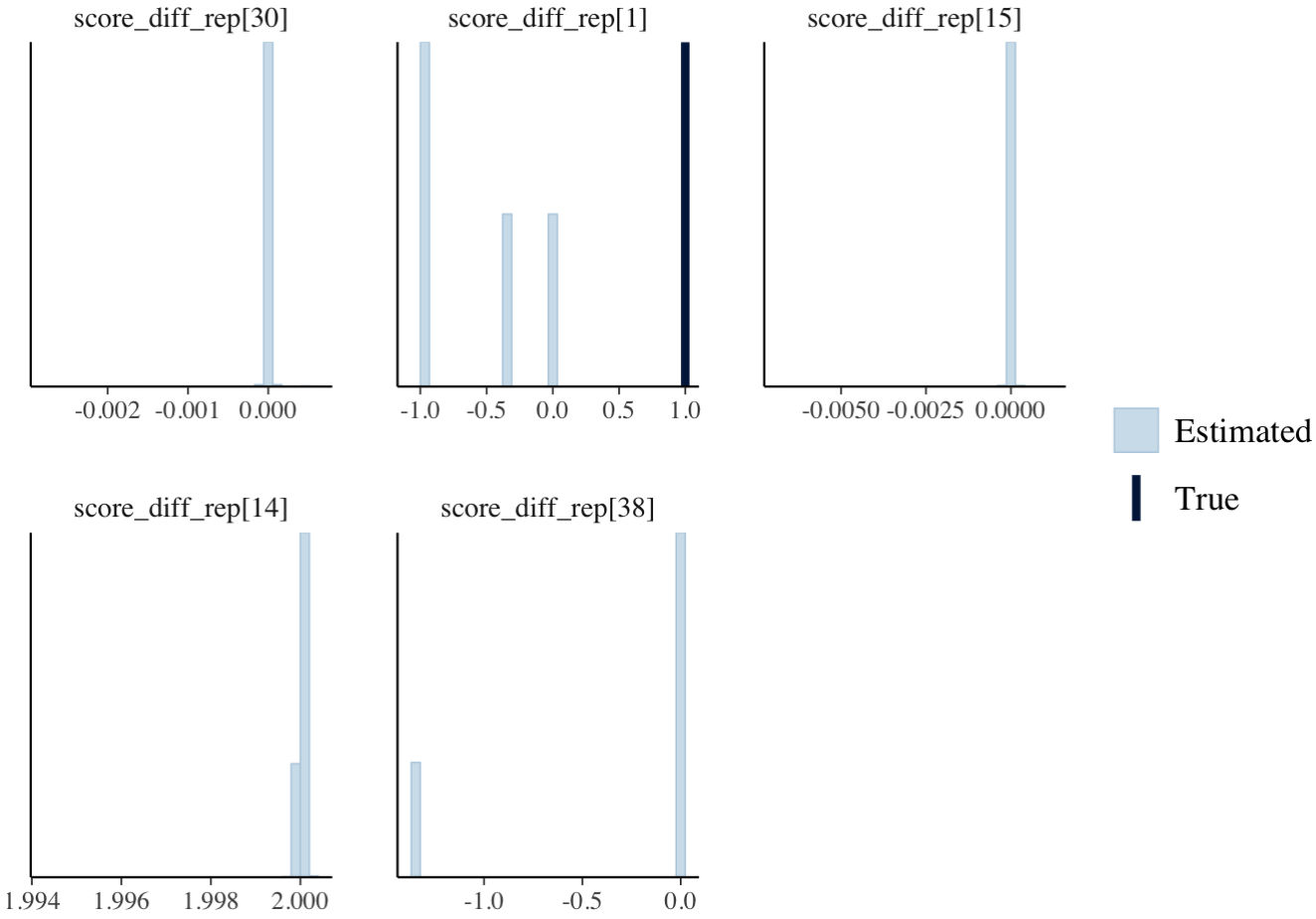
```
##    * Does _not_ affect other ggplot2 plots
```

```
##    * See ?bayesplot_theme_set for details on theme setting
```

```
posterior_score_diff_rep <- as.matrix(test_result, pars = c('score_diff_rep'))[,games]
true_score_diff <- epl$score_diff[40:50][games]
mcmc_recover_hist(posterior_score_diff_rep, true = true_score_diff)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

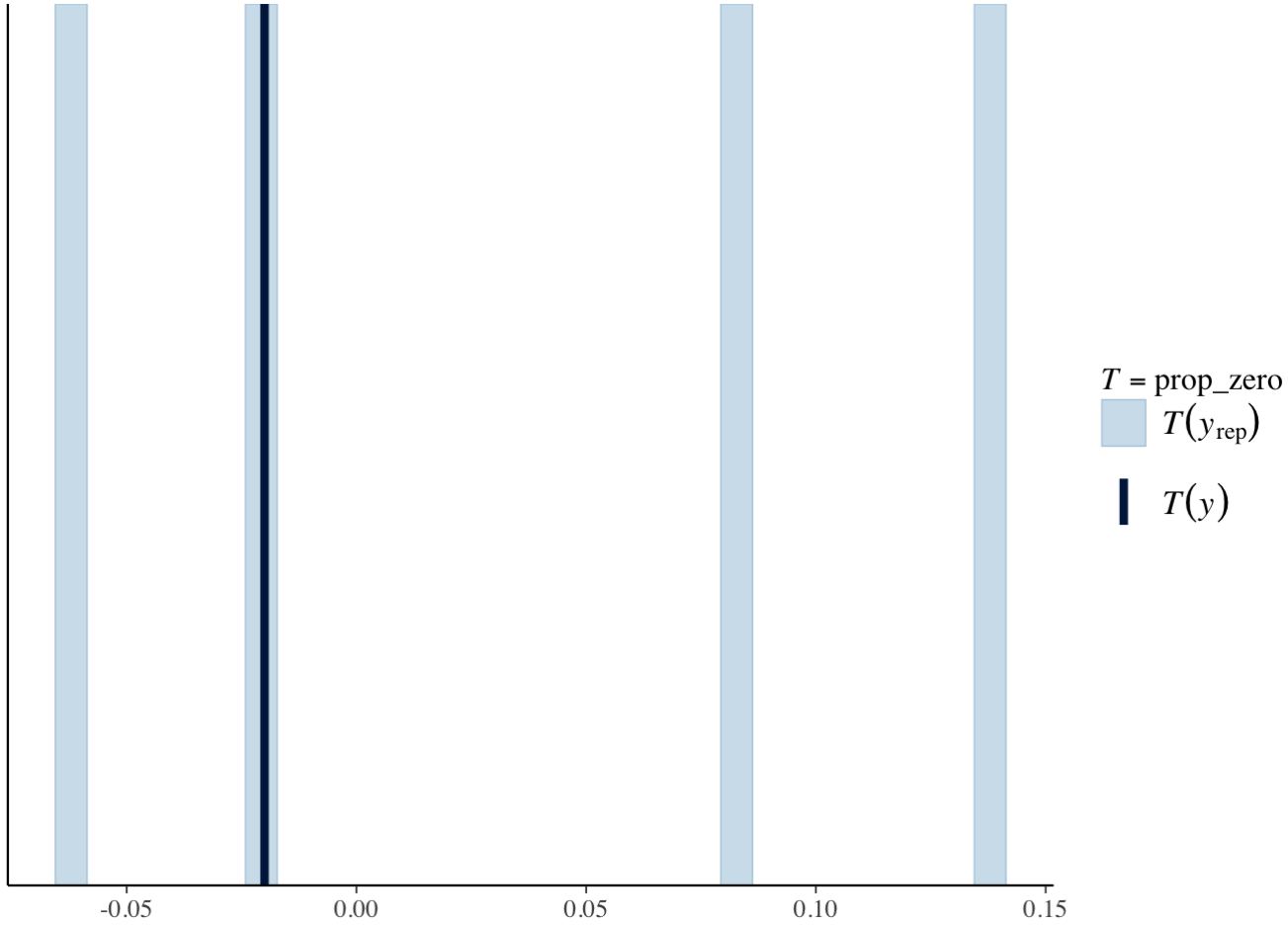
```
## Warning: Removed 4 rows containing missing values (geom_vline).
```



As we can see the result is not good.

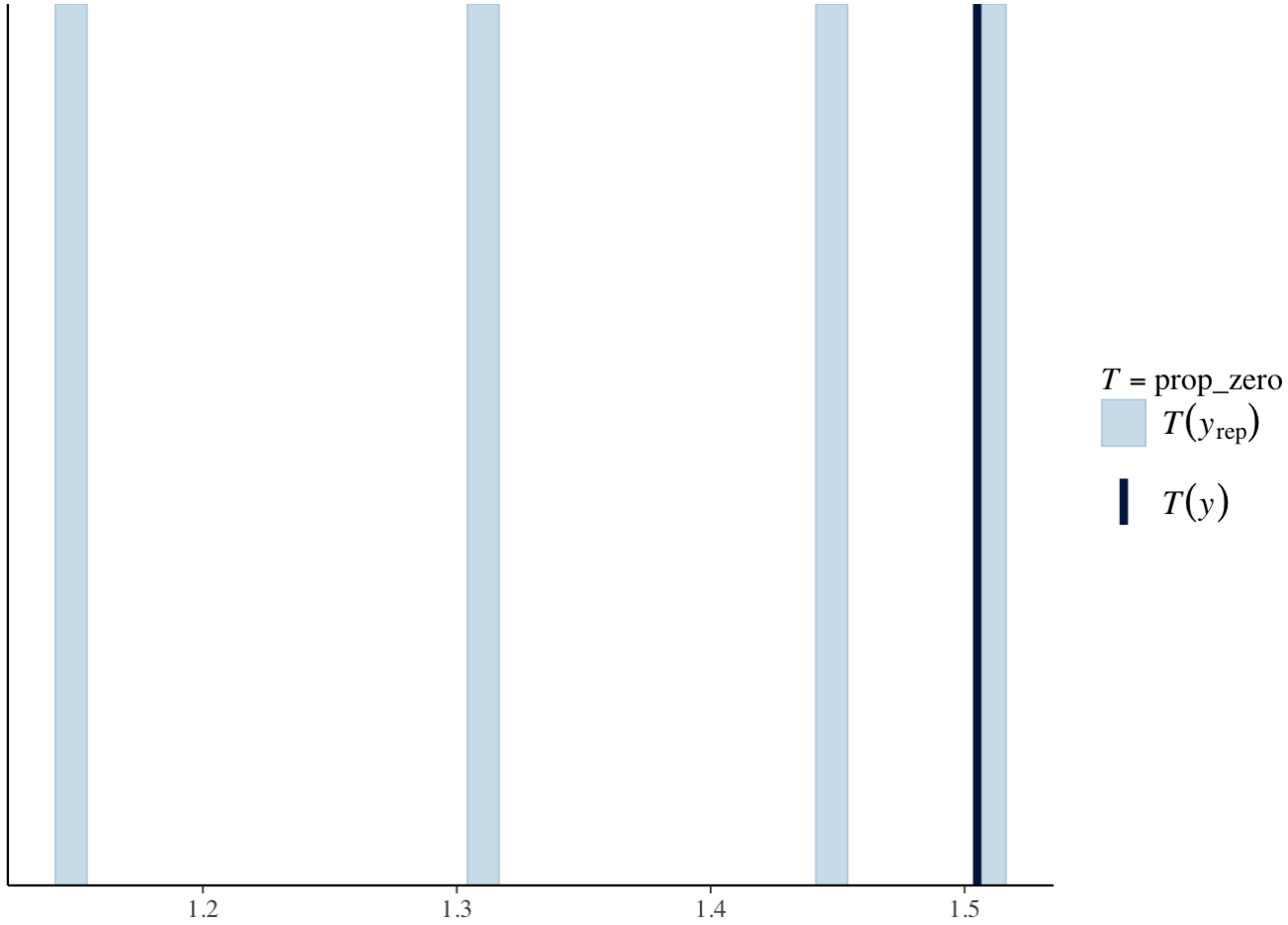
```
prop_zero <- function(x) mean(x)
ppc_stat(y = epl$score_diff[1:50], yrep = round(as.matrix(test_result, pars = c('score_d
iff_rep'))), stat = "prop_zero")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

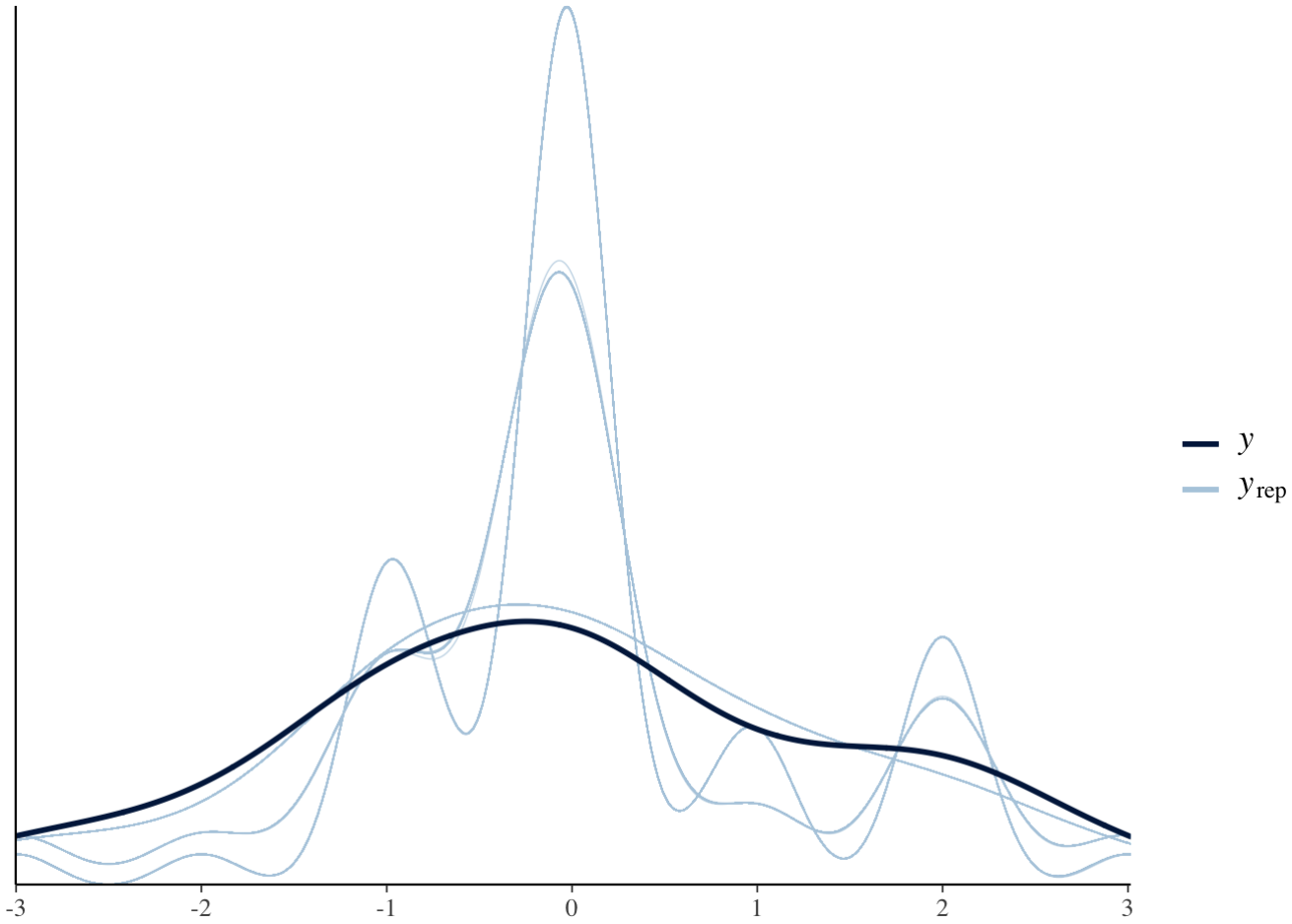



```
prop_zero <- function(x) sd(x)
ppc_stat(y = epl$score_diff[1:50], yrep = round(as.matrix(test_result, pars = c('score_d
iff_rep'))), stat = "prop_zero")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ppc_dens_overlay(y = epl$score_diff[1:50], as.matrix(test_result, pars = c('score_diff_r
ep')))
```



As we can see the result is very unrobust

extend the model

My idea:

1. I will fix some unreasonable distribution setting problem. Make a more robust prior and better likelihood distribution.
2. I will redo the model wih my new code.

1. based on the rstan textbook chapter 3 world cup example, i will choose the score difference follows the t distribution with df = 7 rather than a random variable.
2. more robust priors

```
fit_model <- stan_model('homework8.stan')
print_file('homework8.stan')
```

```
## /*
## This is used to simulate the fit the real data
## */
##
## data{
##   int<lower=2> nteams; // number of teams
##   int<lower=1> ngames; // number of games
##   int<lower=1> nweeks; // number of weeks
##   int<lower=1> home_week[ngames]; // week number for the home team
##   int<lower=1> away_week[ngames]; // week number for the away team
##   int<lower=1, upper=nteams> home_team[ngames]; // home team ID (1, ..., 20)
##   int<lower=1, upper=nteams> away_team[ngames]; // away team ID (1, ..., 20)
##   vector[ngames] score_diff;    // home_goals - away_goals
##   vector[nteams] prev_perf;// a score between -1 and 1
## }
## parameters{
##   real b_home; // the effect of hosting the game in mean of score_diff dist.
##   real b_prev;
##   real<lower=0> sigma_a0;
##   real<lower=0> tau_a;
##   real<lower=1> nu;
##   real<lower=0> sigma_y;
##   row_vector<lower=0>[nteams] sigma_a_raw; // game-to-game variation
##   matrix[nweeks,nteams] eta_a;         // random component
## }
## transformed parameters{
##   matrix[nweeks, nteams] a; // team abilities
##   row_vector<lower=0>[nteams] sigma_a; // game-to-game variation
##   // simulation the abilities matrix
##   for (j in 1:nteams){
##     a[1,j] = b_prev * prev_perf[j] + sigma_a0 * eta_a[1,j]; //initial abilities(at we
ek 1)
##   }
##   for(i in 1:nteams){
##       sigma_a[i] = fabs(tau_a * sigma_a_raw[i]);
##   }
##   for (w in 2:nweeks){
##     a[w,] = a[w-1,] + sigma_a .* eta_a[w,];
##   }
## }
## model{
##   vector[ngames] a_diff;
##   // priors
##   nu ~ gamma(2,0.1);
##   b_prev ~ normal(0,1);
##   sigma_a0 ~ normal(0,1);
##   sigma_y ~ normal(0,5);
##   b_home ~ normal(0,1);
##   sigma_a_raw ~ normal(0,1);
##   tau_a ~ cauchy(0,1);
##   to_vector(eta_a) ~ normal(0,1);
##   for (g in 1:ngames){
##     a_diff[g] = a[home_week[g],home_team[g]] - a[away_week[g],away_team[g]];
```

```
##   }
##   for (g in 1:ngames){
##     score_diff ~ student_t(nu,a_diff + b_home,sigma_y);
##   }
## }
## generated quantities{
##   vector[ngames] score_diff_rep;
##   for (g in 1:ngames){
##         score_diff_rep[g] = student_t_rng(nu, a[home_week[g],home_team[g]] - a[away_w
eek[g],away_team[g]]+b_home, sigma_y);
##   }
## }
```

```
fit_model <- stan_model('new.stan')
for (w in 1:5) {
  epl_w <- epl
  idx <- c(1:(w*10))
  epl_w$home_team <- epl$home_team[idx]
  epl_w$away_team <- epl$away_team[idx]
  epl_w$home_goals <- epl$home_goals[idx]
  epl_w$away_goals <- epl$away_goals[idx]
  epl_w$score_diff <- epl$score_diff[idx]
  epl_w$home_week <- epl$home_week[idx]
  epl_w$away_week <- epl$away_week[idx]
  epl_w$ngames <- w*10
  epl_w$nweeks <- max(c(epl_w$home_week, epl_w$away_week))
  fit <- sampling(fit_model, chains = 4, iter = (nsamples/2), data = epl_w)
  saveRDS(fit, paste("fit_new_", w, ".rds", sep=""))
}


test_result <- readRDS('fit_new_1.rds')
print(test_result,pars = 'a')
```

```
## Inference for Stan model: new.
## 4 chains, each with iter=750; warmup=375; thin=1;
## post-warmup draws per chain=375, total post-warmup draws=1500.
##
##          mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff     Rhat
## a[1,1]  -0.28    0.20 0.28 -0.63 -0.51 -0.27 -0.05  0.05     2  5997.29
## a[1,2]   0.21    0.19 0.27 -0.17  0.04  0.23  0.41  0.57     2  5315.95
## a[1,3]  -0.36    0.25 0.35 -0.78 -0.63 -0.39 -0.12  0.12     2  5705.78
## a[1,4]   0.09    0.07 0.11 -0.07  0.05  0.11  0.15  0.22     2  2634.03
## a[1,5]   0.50    0.35 0.49  0.00  0.05  0.40  0.85  1.20     2 16236.00
## a[1,6]   0.12    0.09 0.13  0.00  0.05  0.07  0.14  0.33     2  2485.48
## a[1,7]   1.41    0.29 0.41  0.77  1.23  1.50  1.68  1.89     2  8944.92
## a[1,8]   0.29    0.14 0.20  0.01  0.20  0.29  0.39  0.59     2  3522.59
## a[1,9]   0.28    0.09 0.13  0.14  0.16  0.27  0.39  0.45     2  3813.99
## a[1,10]  1.03    0.12 0.18  0.73  0.98  1.10  1.15  1.17     2  2268.32
## a[1,11]  0.23    0.13 0.18  0.03  0.14  0.18  0.27  0.52     2  5817.87
## a[1,12] -0.49    0.18 0.26 -0.83 -0.64 -0.51 -0.36 -0.10     2  9058.57
## a[1,13] -0.19    0.10 0.15 -0.41 -0.29 -0.16 -0.07 -0.04     2  6390.38
## a[1,14] -0.28    0.10 0.15 -0.50 -0.35 -0.26 -0.19 -0.10     2  3470.35
## a[1,15] -0.43    0.23 0.32 -0.88 -0.65 -0.36 -0.13 -0.11     2  6100.09
## a[1,16] -0.33    0.12 0.17 -0.55 -0.48 -0.32 -0.18 -0.14     2  3460.28
## a[1,17] -0.40    0.15 0.22 -0.62 -0.54 -0.46 -0.32 -0.05     2  2733.46
## a[1,18] -0.31    0.13 0.19 -0.52 -0.46 -0.34 -0.19 -0.03     2  3824.55
## a[1,19] -1.03    0.21 0.29 -1.45 -1.21 -0.98 -0.79 -0.70     2 11033.28
## a[1,20]  0.71    0.22 0.32  0.37  0.50  0.62  0.83  1.22     2  6193.22
##
## Samples were drawn using NUTS(diag_e) at Sat Oct 27 18:41:58 2018.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```