Data Science Coursework 2

06026373

2025-05-10

Academic integrity statement

"I, 06026373, certify that this assessed coursework is my own work, unless otherwise acknowledged, and includes no plagiarism. I have not discussed my coursework with anyone else except when seeking clarification with the module lecturer via email or on MS Teams. I have not shared any code underlying my coursework with anyone else prior to submission."

Part 1 - Introduction

In the last decades, a statistic called "expected Goal" (xG), has been developed. It represents the likelihood of a shot being scored and is based on several factor like the distance to the goal, the body part used, the distance to the closest defender, ... It has been constantly improved by new studies during the last decades. However, this likelihood is not just due to luck: the difference in individual competence of each player also plays a significant role. Some players consistently outperform their expected goals, suggesting superior finishing skills, decision-making under pressure, ...

This project aims to see if a player scoring more or less than there expected goal can be due to a personal competence of simply luck. To do this we will construct a Bayesian hierarchical model to try to measure the competence of the players during their shots. Then we will modify our model to also include a defensive competence from the opposing team.

Part 2 - Data

We load the data from the website FBREF, which contains a lot of data about each matchs in several leagues. We will use the data from the Premier League, saison 23/24 and 22/23. This data contains a lot of information, however we will only be concerned about: players id, match id, team id, opposite team id, the number of goal of the player in the match, the number of shots, the expected Goal for the player for the match.

We will only keep players that have scored at least 5 goals, or have at least 5 as xG. With this, we are guarantied to have data on players focusing on goals, while still having enough players to analyse:

```
#> There are
#> 8573
#>  entries, composed of
#> 163
#>  unique players.
```

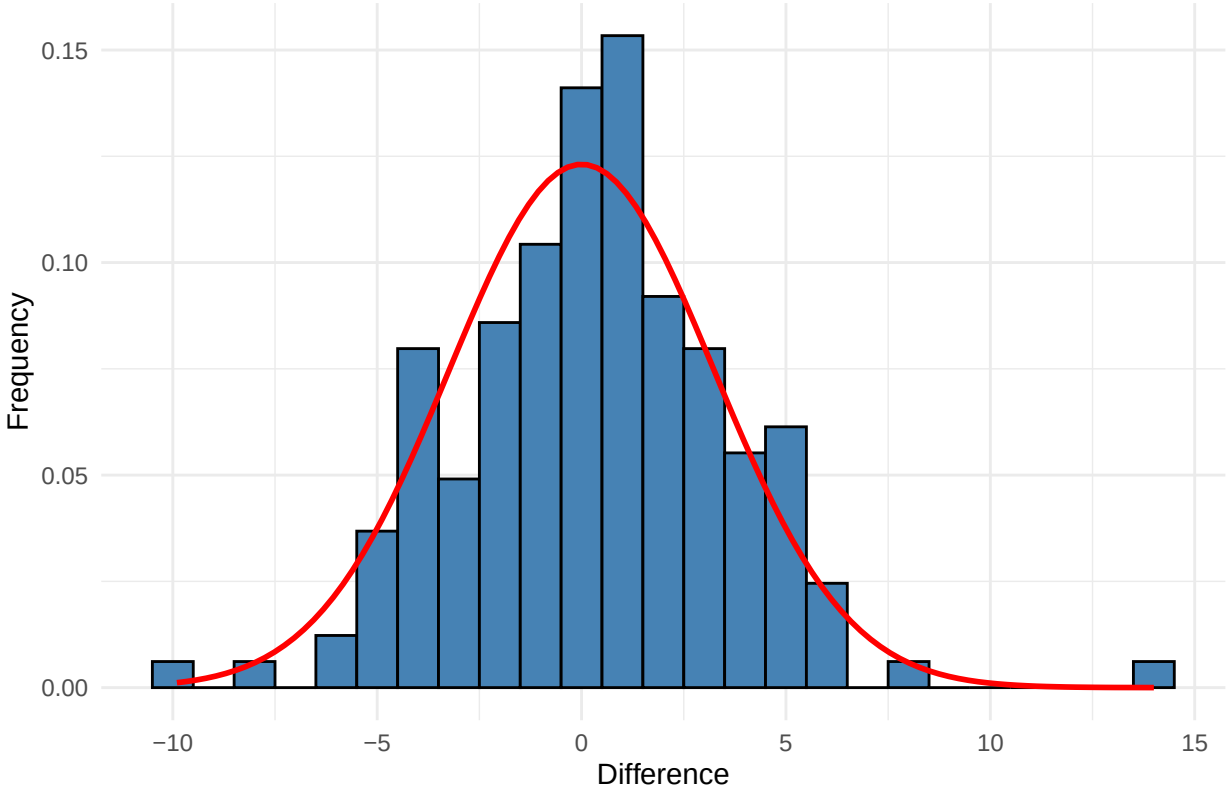
Part 3 - Exploratory data analysis

Before any analysis, we will just check the difference between the number of goal of each players and there expected goal. Normally this value should be close to 0 if the expected goal value is well constructed.

```
#> The mean of the difference between the goal scored and xG is:
#> 0.3509202
#>
#> The variance of the difference between the goal scored and xG is:
#> 10.46684
```

```
#> Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
#> i Please use `linewidth` instead.
#> This warning is displayed once every 8 hours.
#> Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
#> generated.
#> Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
#> i Please use `after_stat(density)` instead.
#> This warning is displayed once every 8 hours.
#> Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
#> generated.
```

Fig.1 − Distribution of the difference between goals and xG



We can see that the mean is closed to 0, this suggest that the expected Goal values is a good tool for this analysis. The data is also closed to a normal law, which confirm the robustness of this statistics. However the variance is only 10.5, which is not a lot as players in the data base played around 50 matchs, so we have only a $\pm$ 0.06 difference. But this is still a lot as there is 38 matchs in a saison, this may result in a difference of 3 goals for the whole season between the players with the best difference, and the worse one.

So it is interesting to see if the difference between the number of goals and the expected goal is due to luck or personal competence.

Modeling players competence

Theoretical model

To quantify the difference between luck and competence, we will use the following Bayesian hierarchical model:

$$Y_{i,j} \sim \text{Bernouilli}(p_{i,j}) \ (1)$$

$$\text{logit}(p_{i,j}) = \text{logit}(\text{xG}_{i,j}) + \theta_i \ (2)$$

$$\theta_i \sim \mathcal{N}(0, \sigma^2) \ (3)$$

$$\sigma \sim \text{Exp}(1) \ (4)$$

With $Y_{i,j} \in 0, 1$ the categorical variable that take the value 1 if the j shot by the i players was a goal, θ_i the competence of the player i, $\text{xG}_{i,j}$ the expected goal of the j shot by the i players.

However we do not have the expected goal statistic for each shot, we only have the expected goal for the whole match. So we will use the following model instead:

$$Z_{i,k} \sim \text{Bin}(n_{i,k}, q_{i,k}) \ (5)$$

$$\text{logit}(q_{i,k}) = \text{logit}\left(\frac{\text{xG}_{i,k}}{n_{i,k}}\right) + \theta_i \ (6)$$

$$\theta_i \sim \mathcal{N}(0, \sigma^2) \ (7)$$

$$\sigma \sim \text{Exp}(1) \ (8)$$

With $Z_{i,k}$ the number of goal scored by the player i on match k, $n_{i,k}$ the number of shot taken by the player i on match k, $\text{xG}_{i,k}$ the expected goal for the whole match k.

```
#> Running MCMC with 4 parallel chains...
#>
#> Chain 1 Iteration:    1 / 5000 [  0%]  (Warmup)
#> Chain 2 Iteration:    1 / 5000 [  0%]  (Warmup)
#> Chain 3 Iteration:    1 / 5000 [  0%]  (Warmup)
#> Chain 3 Informational Message: The current Metropolis proposal is about to be rejected because of the
#> Chain 3 Exception: normal_lpdf: Scale parameter is 0, but must be positive! (in 'C:/Users/mathi/AppDa
#> Chain 3 If this warning occurs sporadically, such as for highly constrained variable types like covar
#> Chain 3 but if this warning occurs often then your model may be either severely ill-conditioned or mi
#> Chain 3
#> Chain 4 Iteration:    1 / 5000 [  0%]  (Warmup)
#> Chain 3 Iteration:  500 / 5000 [ 10%]  (Warmup)
#> Chain 3 Iteration:  501 / 5000 [ 10%]  (Sampling)
#> Chain 4 Iteration:  500 / 5000 [ 10%]  (Warmup)
#> Chain 4 Iteration:  501 / 5000 [ 10%]  (Sampling)
#> Chain 2 Iteration:  500 / 5000 [ 10%]  (Warmup)
#> Chain 2 Iteration:  501 / 5000 [ 10%]  (Sampling)
#> Chain 1 Iteration:  500 / 5000 [ 10%]  (Warmup)
#> Chain 1 Iteration:  501 / 5000 [ 10%]  (Sampling)
#> Chain 4 Iteration: 1000 / 5000 [ 20%]  (Sampling)
#> Chain 3 Iteration: 1000 / 5000 [ 20%]  (Sampling)
#> Chain 1 Iteration: 1000 / 5000 [ 20%]  (Sampling)
#> Chain 4 Iteration: 1500 / 5000 [ 30%]  (Sampling)
#> Chain 3 Iteration: 1500 / 5000 [ 30%]  (Sampling)
#> Chain 4 Iteration: 2000 / 5000 [ 40%]  (Sampling)
#> Chain 2 Iteration: 1000 / 5000 [ 20%]  (Sampling)
```

```
#> Chain 1 Iteration: 1500 / 5000 [ 30%]  (Sampling)
#> Chain 3 Iteration: 2000 / 5000 [ 40%]  (Sampling)
#> Chain 4 Iteration: 2500 / 5000 [ 50%]  (Sampling)
#> Chain 3 Iteration: 2500 / 5000 [ 50%]  (Sampling)
#> Chain 4 Iteration: 3000 / 5000 [ 60%]  (Sampling)
#> Chain 1 Iteration: 2000 / 5000 [ 40%]  (Sampling)
#> Chain 3 Iteration: 3000 / 5000 [ 60%]  (Sampling)
#> Chain 4 Iteration: 3500 / 5000 [ 70%]  (Sampling)
#> Chain 2 Iteration: 1500 / 5000 [ 30%]  (Sampling)
#> Chain 1 Iteration: 2500 / 5000 [ 50%]  (Sampling)
#> Chain 3 Iteration: 3500 / 5000 [ 70%]  (Sampling)
#> Chain 4 Iteration: 4000 / 5000 [ 80%]  (Sampling)
#> Chain 3 Iteration: 4000 / 5000 [ 80%]  (Sampling)
#> Chain 1 Iteration: 3000 / 5000 [ 60%]  (Sampling)
#> Chain 4 Iteration: 4500 / 5000 [ 90%]  (Sampling)
#> Chain 3 Iteration: 4500 / 5000 [ 90%]  (Sampling)
#> Chain 4 Iteration: 5000 / 5000 [100%]  (Sampling)
#> Chain 4 finished in 317.7 seconds.
#> Chain 2 Iteration: 2000 / 5000 [ 40%]  (Sampling)
#> Chain 1 Iteration: 3500 / 5000 [ 70%]  (Sampling)
#> Chain 3 Iteration: 5000 / 5000 [100%]  (Sampling)
#> Chain 3 finished in 338.3 seconds.
#> Chain 1 Iteration: 4000 / 5000 [ 80%]  (Sampling)
#> Chain 2 Iteration: 2500 / 5000 [ 50%]  (Sampling)
#> Chain 1 Iteration: 4500 / 5000 [ 90%]  (Sampling)
#> Chain 1 Iteration: 5000 / 5000 [100%]  (Sampling)
#> Chain 1 finished in 402.1 seconds.
#> Chain 2 Iteration: 3000 / 5000 [ 60%]  (Sampling)
#> Chain 2 Iteration: 3500 / 5000 [ 70%]  (Sampling)
#> Chain 2 Iteration: 4000 / 5000 [ 80%]  (Sampling)
#> Chain 2 Iteration: 4500 / 5000 [ 90%]  (Sampling)
#> Chain 2 Iteration: 5000 / 5000 [100%]  (Sampling)
#> Chain 2 finished in 561.0 seconds.
#>
#> All 4 chains finished successfully.
#> Mean chain execution time: 404.8 seconds.
#> Total execution time: 563.6 seconds.
#> Warning: 4 of 4 chains had an E-BFMI less than 0.3.
#> See https://mc-stan.org/misc/warnings for details.
```

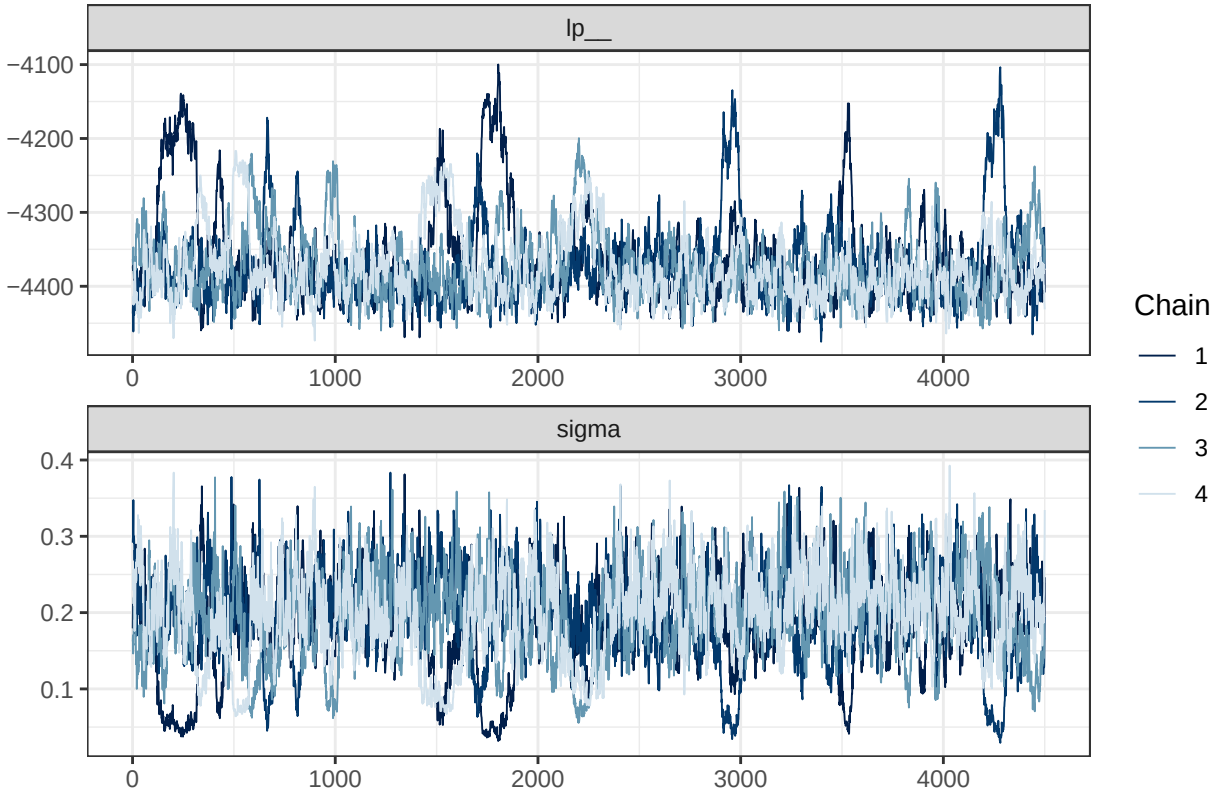
Checking convergence

Before analysing the results, we check if the model converges well.

```
#> The maximum of rhat is:
#> 1.023149
#>
#> The minimum of ess_bulk is:
#> 285.0682
#>
#> The minimum of ess_tail is:
#> 199.9394
```

We have rhat < 1.1 and ess > 100. So the model seems to converge. Let's check y ploting the "worst" variable:

Fig.2 − Values and log−likelihood of the worst performing variable

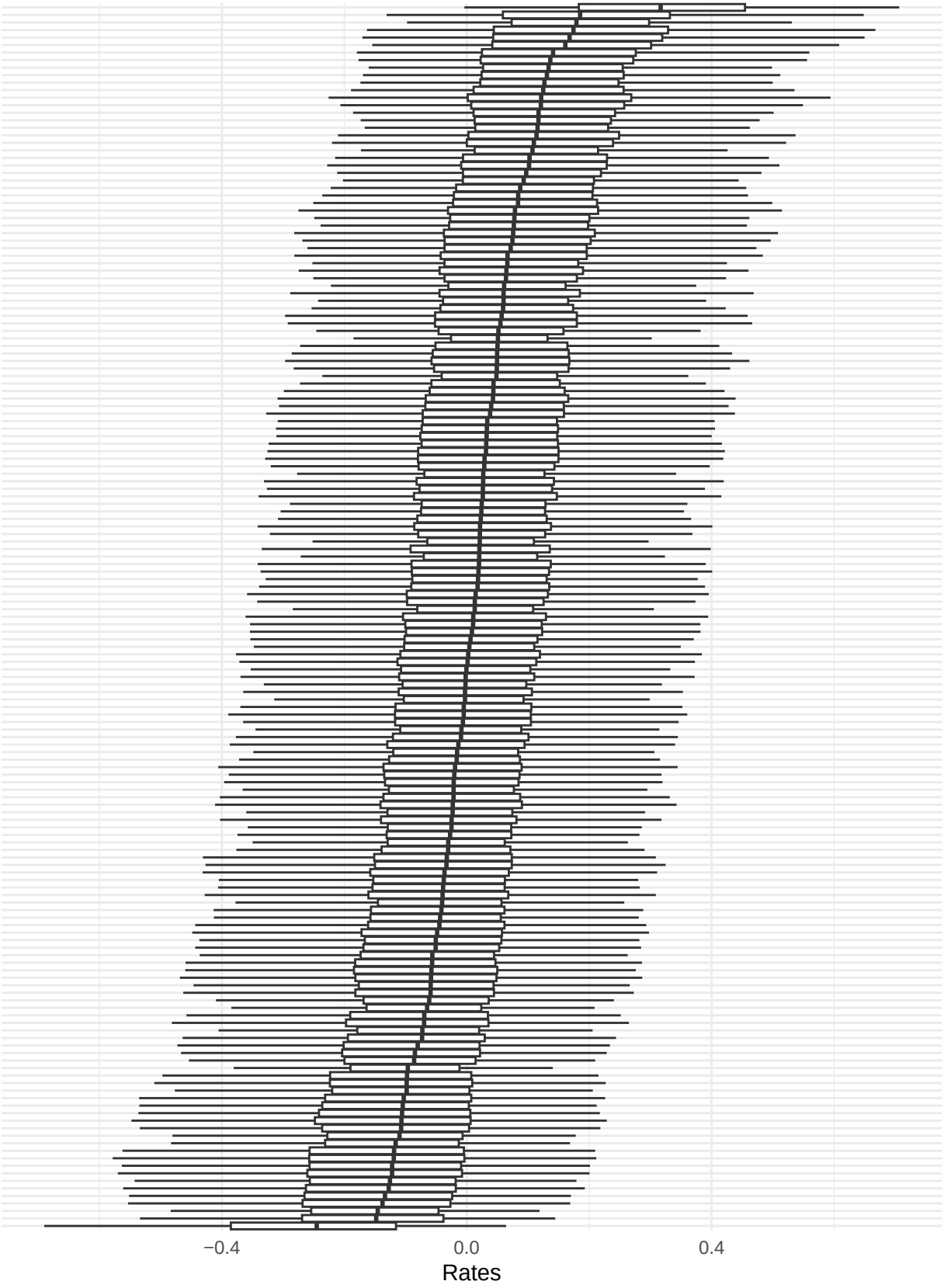


Even if we see some spikes, the variable converges. So our model converges.

Results

We can now plot the results:

Fig.3 − Rates of each players, with 95% confidence intervals



We can clearly that the 95% confidence intervals for the rate cross the 0 value for each players. So it seems that the difference between goals numbers and expected goals is not due to a difference in competence between players.

Adding defensive team competence

Theoretical model

Our first analysis shows that players have no personal competence that distinguish them when shooting. However players of the defending team may also impact the difference between the number of goals and the expected goal value. Here we will model this effect by replacing equation (6) in the previous model with the following equation:

$$\text{logit}(q_{i,j,k}) = \text{logit}\left(\frac{\text{xG}_{i,k}}{n_{i,k}}\right) + \theta_i + \gamma_j \quad (9)$$

With γ_i a parameter for the defending team's competence, which as the following prior:

$$\gamma_j \sim \mathcal{N}(0, \tau^2) \quad (10)$$

$$\tau \sim \text{Exp}(1) \quad (11)$$

We will model the defensive team as only one parameter, even if it is compose of a lot of players with different posts, to keep the model simple and because we cannot know the position of each players at the moment of the shot.

```
#> Running MCMC with 4 parallel chains...
#>
#> Chain 1 Iteration:    1 / 5000 [  0%]  (Warmup)
#> Chain 1 Informational Message: The current Metropolis proposal is about to be rejected because of the
#> Chain 1 Exception: normal_lpdf: Scale parameter is 0, but must be positive! (in 'C:/Users/mathi/AppDa
#> Chain 1 If this warning occurs sporadically, such as for highly constrained variable types like covar
#> Chain 1 but if this warning occurs often then your model may be either severely ill-conditioned or mi
#> Chain 1
#> Chain 2 Iteration:    1 / 5000 [  0%]  (Warmup)
#> Chain 2 Informational Message: The current Metropolis proposal is about to be rejected because of the
#> Chain 2 Exception: normal_lpdf: Scale parameter is 0, but must be positive! (in 'C:/Users/mathi/AppDa
#> Chain 2 If this warning occurs sporadically, such as for highly constrained variable types like covar
#> Chain 2 but if this warning occurs often then your model may be either severely ill-conditioned or mi
#> Chain 2
#> Chain 3 Iteration:    1 / 5000 [  0%]  (Warmup)
#> Chain 3 Informational Message: The current Metropolis proposal is about to be rejected because of the
#> Chain 3 Exception: normal_lpdf: Scale parameter is 0, but must be positive! (in 'C:/Users/mathi/AppDa
#> Chain 3 If this warning occurs sporadically, such as for highly constrained variable types like covar
#> Chain 3 but if this warning occurs often then your model may be either severely ill-conditioned or mi
#> Chain 3
#> Chain 4 Iteration:    1 / 5000 [  0%]  (Warmup)
#> Chain 4 Informational Message: The current Metropolis proposal is about to be rejected because of the
#> Chain 4 Exception: normal_lpdf: Scale parameter is 0, but must be positive! (in 'C:/Users/mathi/AppDa
#> Chain 4 If this warning occurs sporadically, such as for highly constrained variable types like covar
#> Chain 4 but if this warning occurs often then your model may be either severely ill-conditioned or mi
#> Chain 4
#> Chain 4 Iteration:  500 / 5000 [ 10%]  (Warmup)
#> Chain 4 Iteration:  501 / 5000 [ 10%]  (Sampling)
#> Chain 1 Iteration:  500 / 5000 [ 10%]  (Warmup)
#> Chain 1 Iteration:  501 / 5000 [ 10%]  (Sampling)
```

```
#> Chain 3 Iteration:  500 / 5000 [ 10%]  (Warmup)
#> Chain 3 Iteration:  501 / 5000 [ 10%]  (Sampling)
#> Chain 2 Iteration:  500 / 5000 [ 10%]  (Warmup)
#> Chain 2 Iteration:  501 / 5000 [ 10%]  (Sampling)
#> Chain 4 Iteration: 1000 / 5000 [ 20%]  (Sampling)
#> Chain 1 Iteration: 1000 / 5000 [ 20%]  (Sampling)
#> Chain 4 Iteration: 1500 / 5000 [ 30%]  (Sampling)
#> Chain 3 Iteration: 1000 / 5000 [ 20%]  (Sampling)
#> Chain 1 Iteration: 1500 / 5000 [ 30%]  (Sampling)
#> Chain 4 Iteration: 2000 / 5000 [ 40%]  (Sampling)
#> Chain 1 Iteration: 2000 / 5000 [ 40%]  (Sampling)
#> Chain 4 Iteration: 2500 / 5000 [ 50%]  (Sampling)
#> Chain 3 Iteration: 1500 / 5000 [ 30%]  (Sampling)
#> Chain 1 Iteration: 2500 / 5000 [ 50%]  (Sampling)
#> Chain 4 Iteration: 3000 / 5000 [ 60%]  (Sampling)
#> Chain 3 Iteration: 2000 / 5000 [ 40%]  (Sampling)
#> Chain 4 Iteration: 3500 / 5000 [ 70%]  (Sampling)
#> Chain 1 Iteration: 3000 / 5000 [ 60%]  (Sampling)
#> Chain 1 Iteration: 3500 / 5000 [ 70%]  (Sampling)
#> Chain 4 Iteration: 4000 / 5000 [ 80%]  (Sampling)
#> Chain 3 Iteration: 2500 / 5000 [ 50%]  (Sampling)
#> Chain 1 Iteration: 4000 / 5000 [ 80%]  (Sampling)
#> Chain 4 Iteration: 4500 / 5000 [ 90%]  (Sampling)
#> Chain 2 Iteration: 1000 / 5000 [ 20%]  (Sampling)
#> Chain 4 Iteration: 5000 / 5000 [100%]  (Sampling)
#> Chain 4 finished in 320.8 seconds.
#> Chain 1 Iteration: 4500 / 5000 [ 90%]  (Sampling)
#> Chain 3 Iteration: 3000 / 5000 [ 60%]  (Sampling)
#> Chain 1 Iteration: 5000 / 5000 [100%]  (Sampling)
#> Chain 1 finished in 356.3 seconds.
#> Chain 3 Iteration: 3500 / 5000 [ 70%]  (Sampling)
#> Chain 3 Iteration: 4000 / 5000 [ 80%]  (Sampling)
#> Chain 3 Iteration: 4500 / 5000 [ 90%]  (Sampling)
#> Chain 2 Iteration: 1500 / 5000 [ 30%]  (Sampling)
#> Chain 3 Iteration: 5000 / 5000 [100%]  (Sampling)
#> Chain 3 finished in 475.6 seconds.
#> Chain 2 Iteration: 2000 / 5000 [ 40%]  (Sampling)
#> Chain 2 Iteration: 2500 / 5000 [ 50%]  (Sampling)
#> Chain 2 Iteration: 3000 / 5000 [ 60%]  (Sampling)
#> Chain 2 Iteration: 3500 / 5000 [ 70%]  (Sampling)
#> Chain 2 Iteration: 4000 / 5000 [ 80%]  (Sampling)
#> Chain 2 Iteration: 4500 / 5000 [ 90%]  (Sampling)
#> Chain 2 Iteration: 5000 / 5000 [100%]  (Sampling)
#> Chain 2 finished in 1337.3 seconds.
#>
#> All 4 chains finished successfully.
#> Mean chain execution time: 622.5 seconds.
#> Total execution time: 1339.7 seconds.
#> Warning: 3 of 18000 (0.0%) transitions ended with a divergence.
#> See https://mc-stan.org/misc/warnings for details.
#> Warning: 4 of 4 chains had an E-BFMI less than 0.3.
#> See https://mc-stan.org/misc/warnings for details.
```

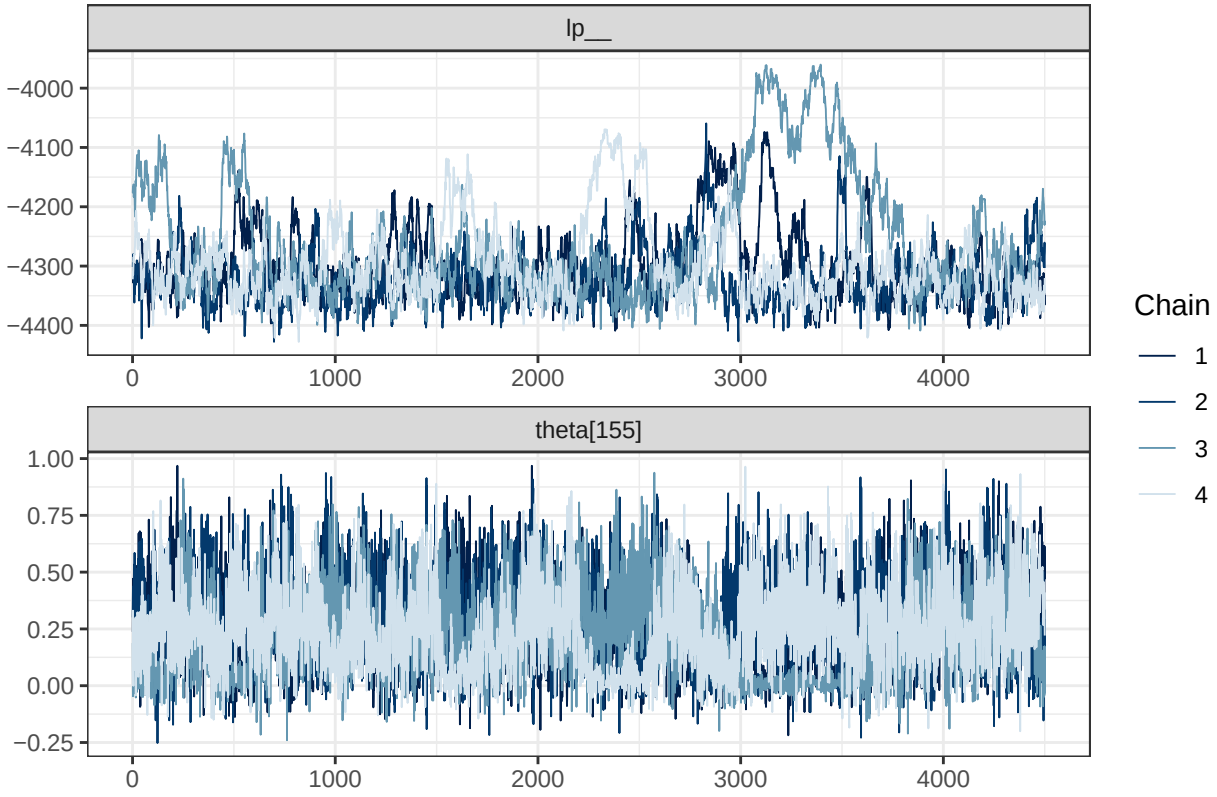
Checking convergence

Before analysing the results, we check if the model converges well.

```
#> The maximum of rhat is:
#> 1.023157
#>
#> The minimum of ess_bulk is:
#> 150.0695
#>
#> The minimum of ess_tail is:
#> 1428.279
```

We have rhat < 1.1 and ess > 100. So the model seems to converge. Let's check y ploting the "worst" variable:

Fig.4 − Values and log−likelihood of the worst performing variable



Even if we see some spikes, the variable converges. So our model converges.

Results

We can now plot the results:

Fig.5 − Rates of each players, with 95% confidence intervals

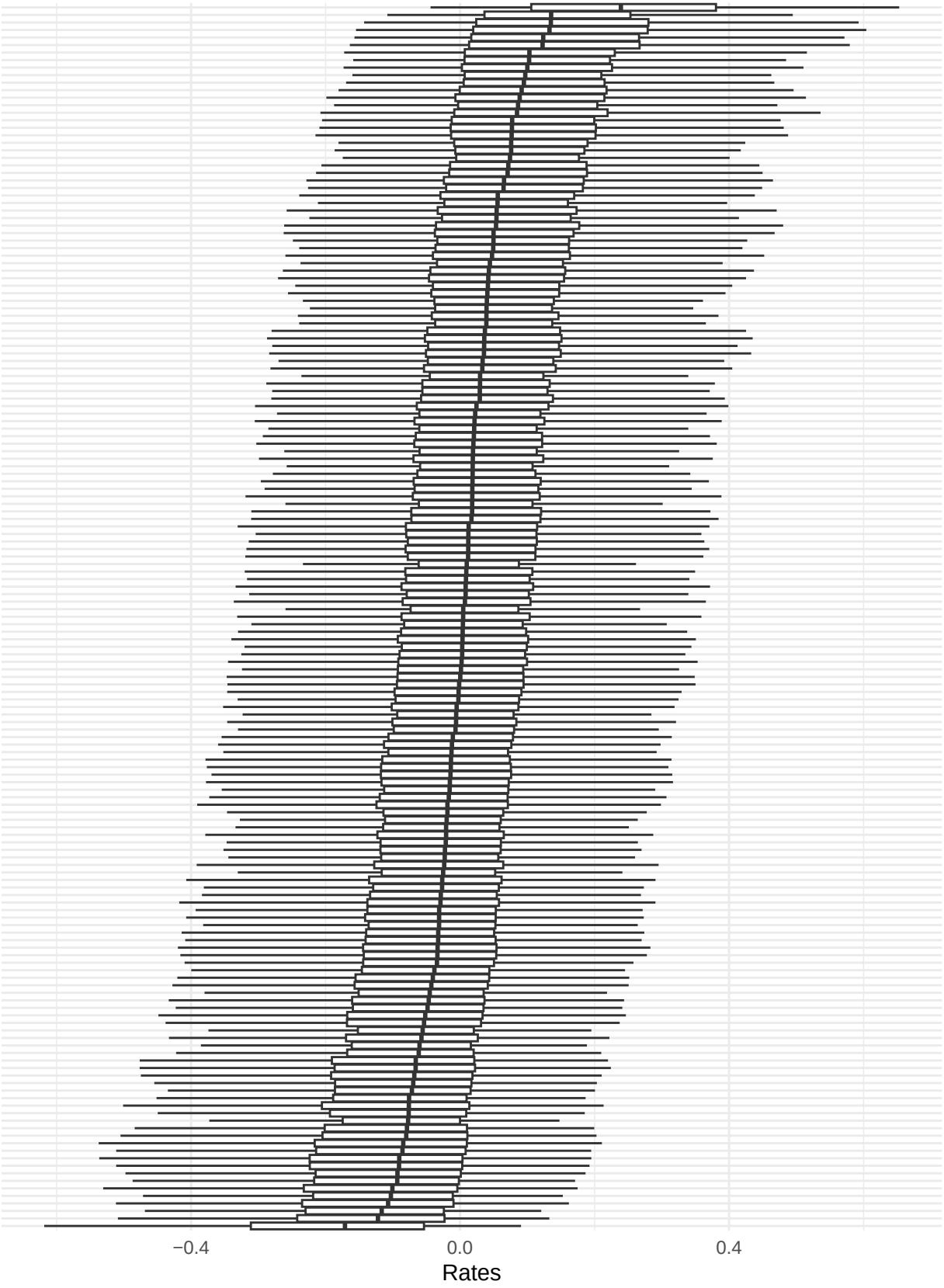
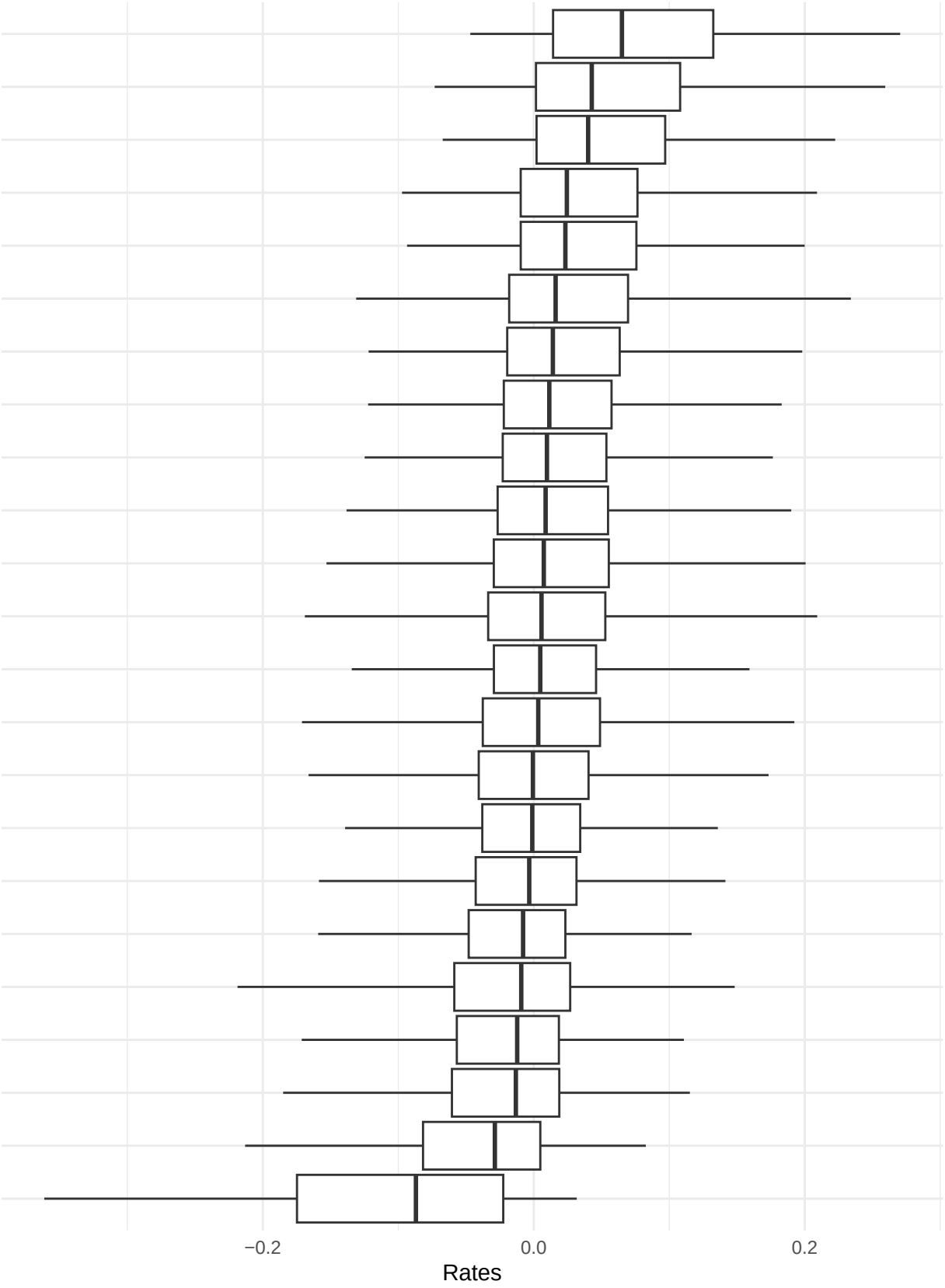


Fig.5 − Rates of each teams, with 95% confidence intervals



The results here confirm our first analysis, the competence of the player still does not have an effect. Furthermore the competence of the defensive team also has no influence: the confidence intervals all cross the rate=0.

Conclusion

In this report, we have modeled with a Bayesian hierarchical model the competence of players and teams in the Premier League when a shot is made, comparing the expected goal statistic to the real goals count. We have found that there is no significant rate for the players and for the teams. So this statistic is not really useful to determine the competence of a player to shot.

However we can also see that we have large confidence interval, compared to the effect measured. If we had more data we might have be able to notice an effect. For this we will need to collect data on a longer period a time, but this is not useful for coach that would like to select the best players, or improve the competences of their players, or to collect data from other matchs, or even training sessions. Furthermore the expected goal statistics has been copied for several other statistics, like expected pass. The latter is more common and so might be more statistically significant and yields better results in a similar analysis.

Finally, it may be interesting to question the usefulness of the expected goal statistic for this analysis. For example take two players: one is competent in shooting, while the other one is good at passing defenders. If they receive the ball at the same position with a defender in front of them, the first on may try to shot directly while the second one will pass the defender first, which may increasing the probability of scoring. But in our analysis the first player will probably be considered as better. So it would be more interesting to have a statistic constructed from the point the player receive the ball, and not were they shot.

Code appendix

```
knitr::opts_chunk$set(
  collapse = TRUE,
  comment = "#>"
)
include_solutions <- TRUE
require(rmarkdown)
require(knitr)
require(kableExtra)
require(data.table)
require(dplyr)
require(ggplot2)
require(cmdstanr)
require(bayesplot)
################
# Part 2 - Data#
################

data.dir <- 'data/'
out.dir <- 'output/'
# Import the data
file_2324 <- file.path(data.dir, 'Premier_League2324.csv')
file_2223 <- file.path(data.dir, 'Premier_League2223.csv')
df_2324 <- as.data.table(read.csv(file_2324))
df_2223 <- as.data.table(read.csv(file_2223))

# Take only the relevant column:
```

```r
# player.tag, team.tag, id_team_A, id_team_B, match.tag, Goals, Shots.Total, xG..Expected.Goals, # Minu
df_2324 <- df_2324[, .('Player' = player.tag,
                       'Team' = team.tag,
                       'Team_A' = id_team_A,
                       'Team_B' = id_team_B,
                       'Match' = match.tag,
                       'Goals' = Goals,
                       'Shots' = Shots.Total,
                       'xG' = xG..Expected.Goals,
                       'Minutes' = Minutes
                        )]
df_2223 <- df_2223[, .('Player' = player.tag,
                       'Team' = team.tag,
                       'Team_A' = id_team_A,
                       'Team_B' = id_team_B,
                       'Match' = match.tag,
                       'Goals' = Goals,
                       'Shots' = Shots.Total,
                       'xG' = xG..Expected.Goals,
                       'Minutes' = Minutes
                        )]

# Combine both datasets
df <- bind_rows(df_2223, df_2324)

# Create an Opposite_Team column
df <- df %>%
  mutate(
    Opposite_Team = ifelse(Team == Team_A, Team_B, Team_A)
  )
# Create a dataset with only players scoring at least 5 goals or with xG >=5
df_combined <- df %>%
  group_by(Player) %>%
  summarise(
    total_minutes = sum(Minutes, na.rm = TRUE),
    total_goals = sum(Goals, na.rm = TRUE),
    total_xG = sum(xG, na.rm = TRUE)
  ) %>%
  filter(total_goals >= 5 |
           total_xG >= 5)

# Filter the original datasets
df <- df %>%
  filter(Player %in% df_combined$Player)

cat('There are ')
cat(nrow(df))
cat(' entries, composed of ')
cat(length(unique(df$Player)))
cat(' unique players.')
#######################################
# Part 3 - Exploratory data analysis #
#######################################
```

```r
df_combined$diff <- df_combined$total_goals - df_combined$total_xG

cat('The mean of the difference between the goal scored and xG is: ')
cat(mean(df_combined$diff))
cat('\nThe variance of the difference between the goal scored and xG is: ')
cat(var(df_combined$diff))
# Create a graphique to plot the difference
ggplot(df_combined, aes(x = diff)) +
  geom_histogram(aes(y = ..density..), binwidth = 1,
                 fill = "steelblue", color = "black") +
  stat_function(fun = dnorm, args = list(mean = 0, sd = sqrt(10.5)), color = "red", size = 1) +
  labs(
    title = "Fig.1 - Distribution of the difference between goals and xG",
    x = "Difference",
    y = "Frequency"
  ) +
  theme_minimal()
###################################
# Part 4 - Multi-models analysis #
###################################

# Stan code
competence_txt <- "
data {
  int<lower=1> N;  // number of entries
  int<lower=1> P;  // number of players
  array[N] int<lower=0> Z;  // number of goals scored in the match
  array[N] int<lower=0> S;  // number of shots in the match
  vector[N] mean_xG;  //mean of the expected goals of the shots in the match = xG / S
  array[N] int<lower=1> player_id;
}
parameters {
  vector[P] theta;
  real<lower=0> sigma;
}
transformed parameters {
  vector[N] q;

  q = inv_logit(logit(mean_xG) + theta[player_id]);
}
model {
  // Likelihood
  Z ~ binomial(S, q);
  // Priors
  theta ~ normal(0, sigma);
  sigma ~ exponential(1);
}
"
# data preparation
df$xG_mean <- df$xG / df$Shots
df <- df %>%
  filter(!is.nan(xG_mean)) # Remove lines with no shots
df <- df %>%
```

```r
  filter(xG_mean <= 1) # Remove lines where xG > shots
df <- df %>%
  mutate(xG_mean = ifelse(xG_mean == 1, 0.999,
                   ifelse(xG_mean == 0, 0.001, xG_mean))) # to avoid problem with inv_logit
df <- df %>%
  mutate(Shots = pmax(Shots, Goals))
df$player_id = as.integer(factor(df$Player)) # Create id for the players

# compile the model
competence_filename <- cmdstanr::write_stan_file(
  gsub('\t',' ', competence_txt ),
  dir = tempdir(),
  basename = NULL,
  force_overwrite = FALSE,
  hash_salt = ""
)
competence_compiled <- cmdstanr::cmdstan_model(competence_filename)

# data
stan_data <- list(
  N = nrow(df),
  P = length(unique(df$player_id)),
  Z = df$Goals,
  S = df$Shots,
  mean_xG = df$xG_mean,
  player_id = df$player_id
)

# sample
competence_fit <- competence_compiled$sample(
  data = stan_data,
  seed = 6373,
  chains = 4,
  parallel_chains = 4,
  iter_warmup = 500,
  iter_sampling = 4500,
  refresh = 500
)

# Check the model
check <- competence_fit$summary(
  variables = c('theta','sigma'),
  posterior::default_summary_measures(),
  posterior::default_convergence_measures()
)

cat('The maximum of rhat is: ')
cat(max(check$rhat))
cat('\nThe minimum of ess_bulk is: ')
cat(min(check$ess_bulk))
cat('\nThe minimum of ess_tail is: ')
cat(min(check$ess_tail))
```

```r
# parameter with lowest ess_bulk
competence_worst_var <- check$variable[which.min(check$ess_bulk)]

# extract samples
competence_po <- competence_fit$draws(
  variables = c("lp__",competence_worst_var),
  inc_warmup = FALSE,
  format = "draws_array"
  )

# make trace plot
p <- bayesplot:::mcmc_trace(competence_po,
                            pars = c("lp__",competence_worst_var),
                            facet_args = list(nrow = 2)
                            )
p + theme_bw() + ggtitle('Fig.2 - Values and log-likelihood of the worst performing variable')

competence_po <- competence_fit$draws(
  variables = c("theta"),
  inc_warmup = FALSE,
  format = "draws_df"
  )

competence_po <- as.data.table(competence_po)
competence_po <- data.table::melt(competence_po, id.vars = c('.chain','.iteration','.draw') )

competence_po_summary <-
  competence_po[,
            list( summary_value = quantile(value, prob = c(0.025, 0.25, 0.5, 0.75, 0.975)),
                  summary_name = c('q_lower','iqr_lower','median','iqr_upper','q_upper')
                  ),
            by = 'variable'
            ]
# Create a table with the results
results <- competence_po_summary[grepl("^theta\\[", variable)]
results[, player_id := as.integer(gsub("theta\\[([0-9]+)\\]", "\\1", variable))]
results <- dcast(results, player_id ~ summary_name, value.var = "summary_value")
# Plot the result
ggplot(data = results, aes(x = reorder(player_id, median),
                           ymin = q_lower, lower = iqr_lower, middle = median,
                           upper = iqr_upper, ymax = q_upper)) +
  geom_boxplot(stat = "identity") +
  coord_flip() +
  labs(x = NULL, y = "Rates",
       title = "Fig.3 - Rates of each players, with 95% confidence intervals") +
  theme_minimal() +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
team_txt <- "
data {
  int<lower=1> N;  // number of entries
  int<lower=1> P;  // number of players
  int<lower=1> Team;  //number of teams
```

```
  array[N] int<lower=0> Z;  // number of goals scored in the match
  array[N] int<lower=0> S;  // number of shots in the match
  vector[N] mean_xG;  //mean of the expected goals of the shots in the match = xG / S
  array[N] int<lower=1> player_id;
  array[N] int<lower=1> team_id;
}
parameters {
  vector[P] theta;
  vector[Team] gamma;
  real<lower=0> sigma;
  real<lower=0> tau;
}
transformed parameters {
  vector[N] q;

  q = inv_logit(logit(mean_xG) + theta[player_id] + gamma[team_id]);
}
model {
  // Likelihood
  Z ~ binomial(S, q);
  // Priors
  theta ~ normal(0, sigma);
  sigma ~ exponential(1);
  gamma ~ normal(0, tau);
  tau ~ exponential(1);
}
"

# data preparation: add a column for team_id
df$team_id <- as.integer(factor(df$Team))
# compile the model
team_filename <- cmdstanr::write_stan_file(
  gsub('\t',' ', team_txt ),
  dir = tempdir(),
  basename = NULL,
  force_overwrite = FALSE,
  hash_salt = ""
)
team_compiled <- cmdstanr::cmdstan_model(team_filename)

# data
stan_data <- list(
  N = nrow(df),
  P = length(unique(df$player_id)),
  Team = length(unique(df$team_id)),
  Z = df$Goals,
  S = df$Shots,
  mean_xG = df$xG_mean,
  player_id = df$player_id,
  team_id = df$team_id
)

# sample
```

```r
team_fit <- team_compiled$sample(
  data = stan_data,
  seed = 6373,
  chains = 4,
  parallel_chains = 4,
  iter_warmup = 500,
  iter_sampling = 4500,
  refresh = 500
)

# Check the model
check <- team_fit$summary(
  variables = c('theta', 'gamma'),
  posterior::default_summary_measures(),
  posterior::default_convergence_measures()
)

cat('The maximum of rhat is: ')
cat(max(check$rhat))
cat('\nThe minimum of ess_bulk is: ')
cat(min(check$ess_bulk))
cat('\nThe minimum of ess_tail is: ')
cat(min(check$ess_tail))

# parameter with lowest ess_bulk
team_worst_var <- check$variable[which.min(check$ess_bulk)]

# extract samples
team_po <- team_fit$draws(
  variables = c("lp__",team_worst_var),
  inc_warmup = FALSE,
  format = "draws_array"
  )

# make trace plot
p <- bayesplot:::mcmc_trace(team_po,
                            pars = c("lp__",team_worst_var),
                            facet_args = list(nrow = 2)
                            )
p + theme_bw() + ggtitle('Fig.4 - Values and log-likelihood of the worst performing variable')

team_po_player <- team_fit$draws(
  variables = c("theta"),
  inc_warmup = FALSE,
  format = "draws_df"
  )

team_po_player <- as.data.table(team_po_player)
team_po_player <- data.table::melt(team_po_player, id.vars = c('.chain','.iteration','.draw') )

team_po_player_summary <-
  team_po_player[,
            list( summary_value = quantile(value, prob = c(0.025, 0.25, 0.5, 0.75, 0.975)),
```

```
                  summary_name = c('q_lower','iqr_lower','median','iqr_upper','q_upper')
                  ),
            by = 'variable'
            ]
# Create a table with the results
results_player <- team_po_player_summary[grepl("^theta\\[", variable)]
results_player[, player_id := as.integer(gsub("theta\\[([0-9]+)\\]", "\\1", variable))]
results_player <- dcast(results_player, player_id ~ summary_name, value.var = "summary_value")
# Plot the result
ggplot(data = results_player, aes(x = reorder(player_id, median),
                          ymin = q_lower, lower = iqr_lower, middle = median,
                          upper = iqr_upper, ymax = q_upper)) +
  geom_boxplot(stat = "identity") +
  coord_flip() +
  labs(x = NULL, y = "Rates",
       title = "Fig.5 - Rates of each players, with 95% confidence intervals") +
  theme_minimal() +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
team_po_team <- team_fit$draws(
  variables = c("gamma"),
  inc_warmup = FALSE,
  format = "draws_df"
  )

team_po_team <- as.data.table(team_po_team)
team_po_team <- data.table::melt(team_po_team, id.vars = c('.chain','.iteration','.draw') )

team_po_team_summary <-
  team_po_team[,
            list( summary_value = quantile(value, prob = c(0.025, 0.25, 0.5, 0.75, 0.975)),
                  summary_name = c('q_lower','iqr_lower','median','iqr_upper','q_upper')
                  ),
            by = 'variable'
            ]
# Create a table with the results
results_team <- team_po_team_summary[grepl("^gamma\\[", variable)]
results_team[, team_id := as.integer(gsub("gamma\\[([0-9]+)\\]", "\\1", variable))]
results_team <- dcast(results_team, team_id ~ summary_name, value.var = "summary_value")
# Plot the result
ggplot(data = results_team, aes(x = reorder(team_id, median),
                          ymin = q_lower, lower = iqr_lower, middle = median,
                          upper = iqr_upper, ymax = q_upper)) +
  geom_boxplot(stat = "identity") +
  coord_flip() +
  labs(x = NULL, y = "Rates",
       title = "Fig.5 - Rates of each teams, with 95% confidence intervals") +
  theme_minimal() +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
# Ouput the results
df_player <- unique(df[, .(Player, player_id)])
df_team <- unique(df[, .(Team, team_id)])
```

```
results <- merge(results, df_player, by='player_id')
results$player_id <- NULL
results_player <- merge(results_player, df_player, by='player_id')
results_player$player_id <- NULL
results_team <- merge(results_team, df_team, by='team_id')
results_team$team_id <- NULL
setcolorder(results, c("Player", "q_lower", "iqr_lower", "median", "iqr_upper", "q_upper"))
setcolorder(results_player,
            c("Player", "q_lower", "iqr_lower", "median", "iqr_upper", "q_upper"))
setcolorder(results_team,
            c("Team", "q_lower", "iqr_lower", "median", "iqr_upper", "q_upper"))
fwrite(results, file.path(out.dir, 'results_players_only.csv'))
fwrite(results_player, file.path(out.dir, 'results_players_players_and_team.csv'))
fwrite(results_team, file.path(out.dir, 'results_team_players_and_team.csv'))
```