Predicting Player's Goal Contribution in UEFA Champions League

— Project Report —
Advanced Bayesian Data Analysis

Quang Huy Nguyen
Thanh Nam Nguyen
Ali Majedi

March 14, 2025

TU Dortmund University

1 Introduction

The UEFA Champions League 2025 is underway, and given its significance and popularity, matches are watched by millions of people, and its data are available publicly for anyone to access. The purpose of this report is to demonstrate the capabilities of two different Bayesian models in predicting a player's goal contribution based on their stats. Classical algorithms and machine learning-based models have been employed to tackle such problems, but this report focuses on Bayesian models in this report.

An advantage of Bayesian models is their ability to generate a probability distribution, unlike classical methods that provide point estimates. In the context of predicting a player's goal contribution, a Bayesian model can leverage prior distributions to provide probabilistic estimates for each player's contribution. These models are also better able to handle limited or unbalanced data, both of which are present in this dataset. This report will compare two Bayesian models: one that includes a player's team as a feature and another that does not. It will analyze the models' ability to predict goal contributions, evaluating their effectiveness and interpretability in the context of UEFA Champions League data.

To support our analysis, we will first examine the available dataset and the processing steps required to prepare the data for the model, such as data reduction, cleaning, and transformation. In this section, the reasoning behind the selected variables will be explored. Next, we will analyze the two models used to tackle this challenge, along with the priors, and explain some of the mathematics behind them. Following that, we will clarify key aspects of the code and functions used in this project. Afterward, we will conduct a deeper analysis of the results, focusing on the convergence of the model and other metrics comparing the two. Finally, a conclusion will summarize the findings of the report.

2 Data

2.1 Dataset Overview

The dataset used in training the models in this report was collected from the Kaggle dataset "UEFA Champions League 2025 — Players data" (Wilkins, 2024). The dataset is extensive, containing information on 676 players from 71 nationalities across 35 clubs and is updating on a weekly basis as long as the matches are ongoing. For this report, the data was collected in December 2024. The dataset initially contained 10 tables: Player, Key Stats, Attacking, Attempts, Distribution, Defending, Disciplinary, Goals, Goalkeeping, and Team data.

2.2 Data Processing

Three tables, Defending Data, Disciplinary Data, and Team Data were discarded as the information they contained was not relevant to this project. The remaining tables were all merged using player ID as the key. The resulting table contained 45 variables that will be explored in the following sections. The processing and analysis were mostly done in Python while the model training and testing were performed in R. In Python, libraries such as Pandas, Matplotlib, Seaborn, and NumPy were used to structure the data in a way that is quick and easy to process and visualize.

The values in the dataset were not scaled or normalized because the model did not converge as well and performed worse when the values were converted to a zero mean and unit variance. There are some outliers in goal contribution which will be examined in more detail in the coming sections, but it was included in the data used to train the model as no noticeable change in the model's performance was noticed when they was omitted.

One of the reasons why goal contribution was chosen as the target variable, which is the sum of goals and assists was to increase the number of non-zero values in the target variable. We can see in Figure 1 how the values are distributed, with 68% being zero.

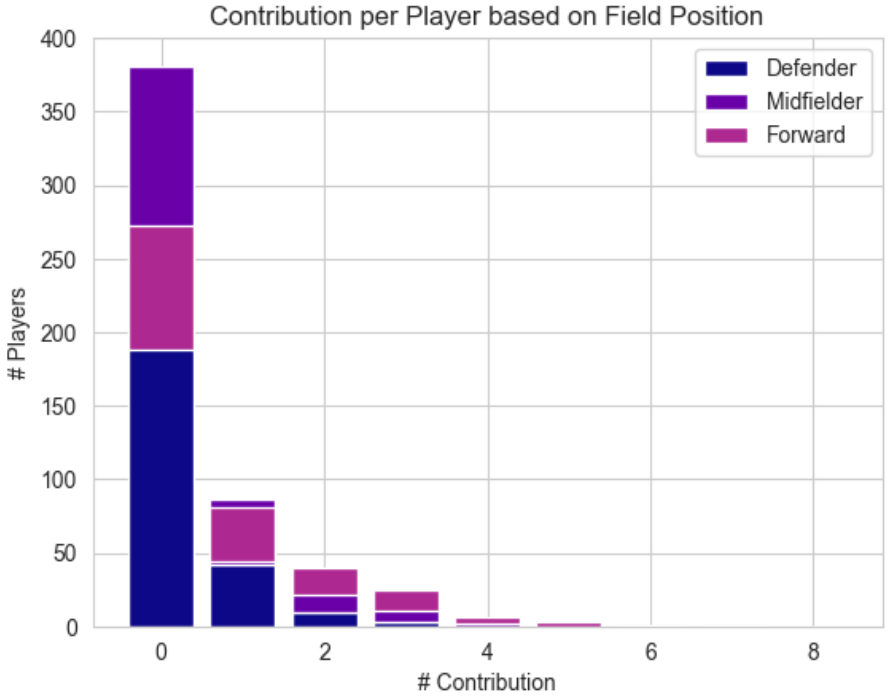

Figure 1: Player goal contribution chart showing the distribution among all the players based on their position.

2.3 Covariate Selection Breakdown

From the initial 45 variables in the dataset, we filtered it down to six essential variables for our models. To keep the model interpretable and efficient, many variables were ultimately excluded and only the following were retained at the end:

- Goal Contribution (Goals + Assists) (response variable)
- Player ID
- Team ID
- Player Field Position
- Minutes Played
- Attempts on Target

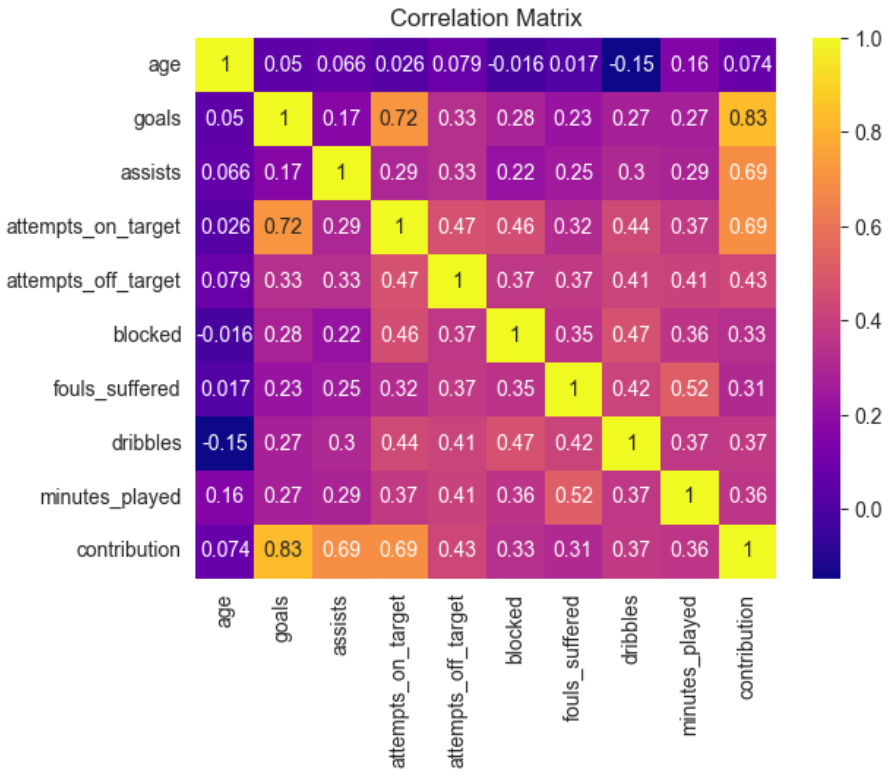

Figure 2: Correlation matrix showing the relationship between the variables in the dataset.

Initially, the variables were selected using intuition as well as the covariance matrix seen in Figure 2. As expected, goals and assists have a very high correlation, and attempts on target also came very close. Other variables such as attempts off target, blocked, fouls suffered, and... also has a positive correlation. However, after training the model despite having a positive and good correlation, it's effect on the outcome of the model did not justify complicating the model and these variables were omitted from the final model presented in this report.

Position is another important variable. Although 40% of the players are midfielders, goal contribution for this role is much less than the defenders (36%) and the forwards (24%) (Figure 1. However, as expected, most of the defenders who contributed had only a single goal or assist.

Attempts on target values have a very high positive correlation with goal contribution, as seen in Figure 2. This correlation is further illustrated in Figure 6 where an increase in attempts also results in a higher goal contribution.

The final variable considered is minutes played. Although the correlation is lower, as shown in Figure7, players with higher goal contribution tend to have a higher average minutes played (excluding contributions 6 and 8, which are outliers).

3 Models

Two types of models are used to predict a player's goal contribution: Player Model and Team Model. Each type has two variants: Linear Model and Nonlinear Model. In total, there are four models, all of which are modeled using the Bayesian regression model function `brm` from **brms** (Bürkner, 2017) package in R. The required parameters of `brm` are model, data, family distribution, and prior. Other parameters required for results analysis include `control = list(adapt_delta = 0.99)`, which increases the step size adaptation to reduce divergences, and `save_pars = save_pars(all = TRUE)` to ensure that all model parameters are saved, which can be useful for posterior analysis and model comparison.

3.1 Player Model

The Player Model predicts the player's goal contribution based on individual player characteristics such as total attempts, minutes played, and field position. For the Player Model, there are two variants of the model: Player Linear Model and Player Nonlinear Model, the coefficients in these models indicate how much each covariate influences the goal contribution. Both the linear and the nonlinear models are negative binomial regression models (`family = negbinomial()`), the link functions and the linear predictors

are as follows:

$$y_i \sim NegBin(E[y_i], \phi)$$

y_i: Goal Contribution.

$E[y_i]$: Mean (expected value) of y_i.

ϕ: The dispersion parameter.

- Player Linear Model (`model_indiv_lin` in R Script):

$$log(E[y_i]) = \beta_0 + \beta_1 \cdot attempts\_on\_target_i + \beta_2 \cdot minutes\_played_i + b_{1,field\_position_i}$$

- Player Nonlinear Model (`model_indiv_nonlin` in R Script):

$$log(E[y_i]) = \beta_0 + f_1(attempts\_on\_target_i) + f_2(minutes\_played_i) + b_{1,field\_position_i}$$

The log link function $log(\mu) = X\beta$ is the most commonly used link function for a negative binomial model because it ensures that $E[y_i]$ remains positive, which is required in this case since goal contribution is a count data. The linear predictor of the linear model contains β_0 as the intercept (also in the nonlinear model too), β_1 and β_2 as the regression coefficients for the covariates attempts on target and minutes played. However, in the nonlinear models, these two variables are included in the smoothing spline functions f with 5 basis functions (`s(...,k=5)` in R Script). The smoothing spline function allows for nonlinear relationships between covariates and the response variable by minimizing the error and ensuring the smoothness of the regression curve. Mathematically, the smoothing spline minimizes the following objective function:

$$\sum_i (y_i - f(x_i))^2 + \lambda \int f''(x)^2 dx$$

λ: hyperparameter to control the strength of the penalty.

The first term $\sum_i (y_i - f(x_i))^2$ represents the sum of squared errors (SSE) between y_i and $f(x_i)$, ensuring that the function $f(x_i)$ fits the data well. The second term has various penalty terms to choose from, but for smoothing splines, the common choice is $\int f''(x)^2 dx$, as it aims for the cubic spline to improve the smoothness of $f(x)$. Using the spline approach, the covariate space is divided into regions, and each region is modeled by a basis function; the number of basis functions determines the flexibility of the smoothing spline. With 5 basis functions, the function can capture moderate curvature in the data while maintaining smoothness and preventing overfitting that occurs when too many basis functions are used.

The mean and the variance of the response variable goal contribution are calculated in R and it shows that the variance of goal contribution (1.007407) is quite larger than

its mean (0.5384615), this case is called overdispersion. To deal with the overdispersion the negative binomial regression is used as a solution, instead of the original plan to use the Poisson regression with the equality between mean and variance. Negative binomial regression adds an additional dispersion parameter ϕ to modify the variance structure:

$$Var(y_i) = \mu_i + \frac{\mu_i^2}{\phi}$$

The additional parameter ϕ allows the regression to capture extra variability in the count data and adjust the variance accordingly to better fit the data.

3.2 Team Model

The Team Model predicts the player's goal contribution based on individual player characteristics, but also incorporates team-level effects, allowing for the possibility that player's that the player's goal contribution may vary depending on the team they play for. Like the Player Model, the Team Model has two variants: Team Linear Model and Team Nonlinear Model. The models are negative binomial regression models and have very similar link functions and linear predictors to the Player Model:

$$y_i \sim NegBin(\mu_i, \phi)$$

y_i: Goal Contribution.

μ_i: Mean (expected value) of y_i.

ϕ: The dispersion parameter.

- Team Linear Model (`model_team_lin` in R Script):

$$log(E[y_i]) = \beta_0 + \beta_1 \cdot attempts\_on\_target_i + \beta_2 \cdot minutes\_played_i + b_{1,field\_position_i} + b_{2,team_i}$$

- Team Nonlinear Model (`model_team_nonlin` in R Script):

$$log(E[y_i]) = \beta_0 + f_1(attempts\_on\_target_i) + f_2(minutes\_played_i) + b_{1,field\_position_i} + b_{2,team_i}$$

The only difference between the linear predictors of the Player Model and the Team Model is the addition of the random intercept for the player's team in the Team Model. Using the random intercept it allows different groups (field positions, teams) to have different baseline levels. Each field position and team gets its own intercept, drawn from a normal distribution with a mean of 0 and some variance reflecting differences between field positions or teams:

$$b_{1,field\_position_i} \sim \mathcal{N}(0, \sigma^2_{field\_positon})$$
$$b_{2,team_i} \sim \mathcal{N}(0, \sigma^2_{team})$$

This means that players in the same field position or on the same team will have some similarity in their goal contributions. The use of random effects is necessary for an accurate model, because in reality some field positions, such as forward or midfielder, obviously have a better goal contribution than defender, and the level of the participating teams is also very different. Both the Player Model and the Team Model capture the field position effects (`1|field_position` in R Script), but only the Team Model captures the team effects (`1|team` in R Script).

4 Priors

One of the most important parts of developing the Bayesian model is the prior, as it affects how the model learns from the data. In this project, the priors for the intercept, regression coefficient, random effects standard deviation (SD), spline smoothness parameter, and dispersion parameter are set using the function `set_prior()` in R. Each model uses a different set of priors, these sets are distinguished by their names in the R script.

The intercept is assumed to be normally distributed with a mean of 0 and a variance of 10, which means that the prior for the intercept is broad and allows for large variations. This can be considered as weakly informative.

$$\beta_0 \sim \mathcal{N}(0, 10)$$

```
set_prior("normal(0, 10)", class = "Intercept")
```

The prior for regression coefficient is applied to β_1 and β_2 in the linear models. This prior also follows a normal distribution with a mean of 0 and a slightly smaller variance of 5. This reflects the belief that **attempts on target** and **minutes played** won't have an extremely large effect, but still allows for flexibility.

$$\beta \sim \mathcal{N}(0, 5)$$

```
set_prior("normal(0, 5)", class = "b")
```

The half-normal distribution with the same mean and variance as above is applied to the random effect SD prior, here the half-normal prior is used to allow for moderate variation in **goal contribution** across teams or field positions. While the prior for field position effect SD is applied to all models, the prior for random team effect SD is applied only to the Team Model.

$$\sigma_{field\_position} \sim |\mathcal{N}(0, 5)|$$

```
set_prior("normal(0, 5)", class = "sd", group = "field_position", lb=0)
```

$$\sigma_{team} \sim |\mathcal{N}(0, 5)|$$

```
set_prior("normal(0, 5)", class = "sd", group = "team", lb=0)
```

Since there are spline functions in the nonlinear models, the prior for the spline smoothness parameter is required. The parameter is assumed to be half-normally distributed with a mean of 0 and a variance of 5. This prior controls how much flexibility the spline functions f1(attempts_on_target) and f2(minutes_played) are allowed to have, $\mathcal{N}(0, 5)$ promotes the smoothness of the spline while still preventing overfitting and avoiding underfitting.

$$\sigma_{sds} \sim |\mathcal{N}(0, 5)|$$

```
set_prior("normal(0, 5)", class = "sds", lb=0)
```

Another prior that is applied to all the models is the prior for the dispersion parameter. Given the negative binomial likelihood, the dispersion parameter must be strictly positive, so the half-normal distribution is the appropriate choice for this prior. The prior distribution with a mean of 0 and a variance of 5 may be weakly informative, but this matches the flexibility of the other priors and additionally prevents overly large dispersion estimates.

$$\phi \sim |\mathcal{N}(0, 5)|$$

```
set_prior("normal(0, 5)", class = "shape", lb=0)
```

Overall, these priors are weakly informative, the reason is due to the lack of expert knowledge. However, the choice to use weakly informative priors fits well with these models because it provides regularization while still allowing the data to drive the inference. This approach is well suited to situations where prior knowledge is uncertain, or where the aim is to allow the data to dominate the model while still avoiding extreme predictions that might arise from noise or outliers.

5 Results Analysis

Our models was fitted using Hamiltonian Monte Carlo via the R package `brms`. The sampling process used 4 chains, each with 2000 iterations (1000 warm-up and 1000 sampling iterations), resulting in a total of 4000 post-warm-up draws. The output of the summary function for our models can be found at the Appendix B.

The results of the summaries indicate that all models yield a negative intercept parameter, meaning that when all covariates are zero, the log goal contribution is negative, and the predicted goal contribution is less than 1. This outcome is expected, as a player's contribution to a goal is contingent on their participation in the game. With regard to `attempts_on_target`, the coefficients estimated by non-linear models are significantly larger than by linear models (16.69 and 16.23 compared to 0.28 and 0.26). This suggests that linear models may not capture the non-linear relationship between this covariate and the response variable.. The same is true for `minutes_played`, and even more extreme, where the estimated coefficients are 0 for both linear models, and are 4.63 and 4.73 for non-linear models. For the random effects, nonlinear models have slightly smaller estimates

for both `field_position` and `team`, suggesting that the inclusion of smooth terms reduces some of the variability attributed to field position. Furthermore, the `shape` parameter values for the non-linear models (8.64 and 10.17) implies that they handle overdispersion in the data better compared to linear models (5.03 and 7.87). Particularly, the team non-linear model has the largest shape value, suggesting that including both non-linear terms and `team` random effect better account for overdispersion. Figure 3 visualize the non-linear relationship between the covariates and goal contribution.

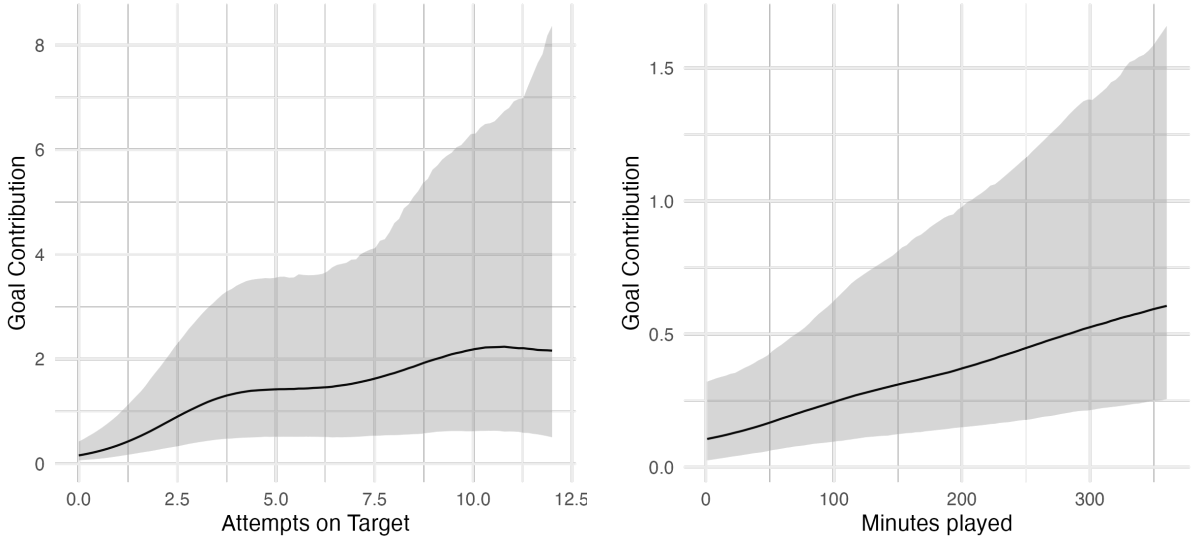

Figure 3: Spline effect from Team Non-linear model.

5.1 Convergence diagnostics

Table 1 summarizes the convergence diagnostics for the four Bayesian models fitted in this project: the Player Linear model, Player Non-linear model, Linear Team model, and Nonlinear Team model. The diagnostics include the R-hat statistic, Bulk Effective Sample Size (Bulk_ESS), and Tail Effective Sample Size (Tail_ESS) for key parameters in each model. Numbers and model in bold indicates the best result.

All models showed good convergence results, with R-hat values close to 1.00 for all parameters. This indicates that the Markov Chain Monte Carlo chains mixed well and explored the posterior distribution effectively.

The Bulk_ESS measures the effective number of independent samples for estimating the posterior mean and central credible intervals. Most Bulk_ESS values were above 1000, indicating efficient sampling for the majority of parameters. However, there were some exceptions. The Player Non-linear model had slightly lower Bulk_ESS values for certain parameters, such as the intercept (Bulk_ESS = 603) and the standard deviation of

field_position (Bulk_ESS = 504). This is likely due to the increased complexity of the model, which includes smooth terms for attempts_on_target and minutes_played. The Player Linear model and Linear Team model showed moderate Bulk_ESS values, with most parameters exceeding 1000. The Nonlinear Team model demonstrated the highest sampling efficiency, with Bulk_ESS values consistently above 1000 (e.g., 3657 for s(attempts_on_target) and 5865 for the shape parameter).

Model	**Parameter**	**Rhat**	**Bulk_ESS**	**Tail_ESS**
Player Linear	Intercept	1.00	945	492
	attempts_on_target	1.00	1496	1619
	minutes_played	1.00	3103	2946
	sd(field_position) (3 levels)	1.00	607	652
	shape	1.00	1774	2123
Player Non-linear	Intercept	1.01	603	216
	s(attempts_on_target)	1.00	1930	2607
	s(minutes_played)	1.00	1361	2316
	sd(field_position) (3 levels)	1.00	504	281
	shape	1.00	3858	2868
Team linear	Intercept	1.01	798	601
	attempts_on_target	1.00	**3284**	**2852**
	minutes_played	1.00	**4655**	**3213**
	sd(field_position) (3 levels)	1.00	652	751
	sd(team) (35 levels)	1.00	1237	**2109**
	shape	1.00	4201	**2871**
Team Non-linear	Intercept	1.00	**1521**	**1062**
	s(attempts_on_target)	1.00	**3657**	**3191**
	s(minutes_played)	1.00	**1934**	**2318**
	sd(field_position) (3 levels)	1.00	**1038**	**1077**
	sd(team) (35 levels)	1.00	**1383**	1330
	shape	1.00	**5865**	2586

Table 1: Convergence Diagnostics Summary

The Tail_ESS measures the effective number of independent samples for estimating the tails of the posterior distribution (e.g., 95% credible intervals). Most Tail_ESS values were above 1000, indicating reliable estimation of the posterior tails. However, the Player Nonlinear model had a Tail_ESS of 216 for the intercept, which is below the ideal threshold of 1000. This suggests that the tails of the posterior distribution for this parameter are less well-estimated, likely due to the model's complexity. The Player Linear model also had a relatively low Tail_ESS for the intercept (Tail_ESS = 492), but this is still acceptable for inference. The Nonlinear Team model again performed best, with Tail_ESS values consistently above 1000 (e.g., 3191 for s(attempts_on_target) and 2586 for the shape parameter).

The Nonlinear Team model stands out as the best-performing model in terms of convergence and sampling efficiency. It has the highest ESS values and excellent R-hat statistics, making it the most reliable model for inference. The Player Non-linear model, while showing good convergence (R-hat ≈ 1.00), had lower ESS values for some parameters, particularly the intercept and random effects. This suggests that the model's complexity may have reduced sampling efficiency for certain parameters. The Player Linear and Linear Team models showed moderate sampling efficiency, with most ESS values above 1000. These models are simpler, and therefore may not capture the data's complexity as effectively as the nonlinear models.

5.2 Posterior predictive checks

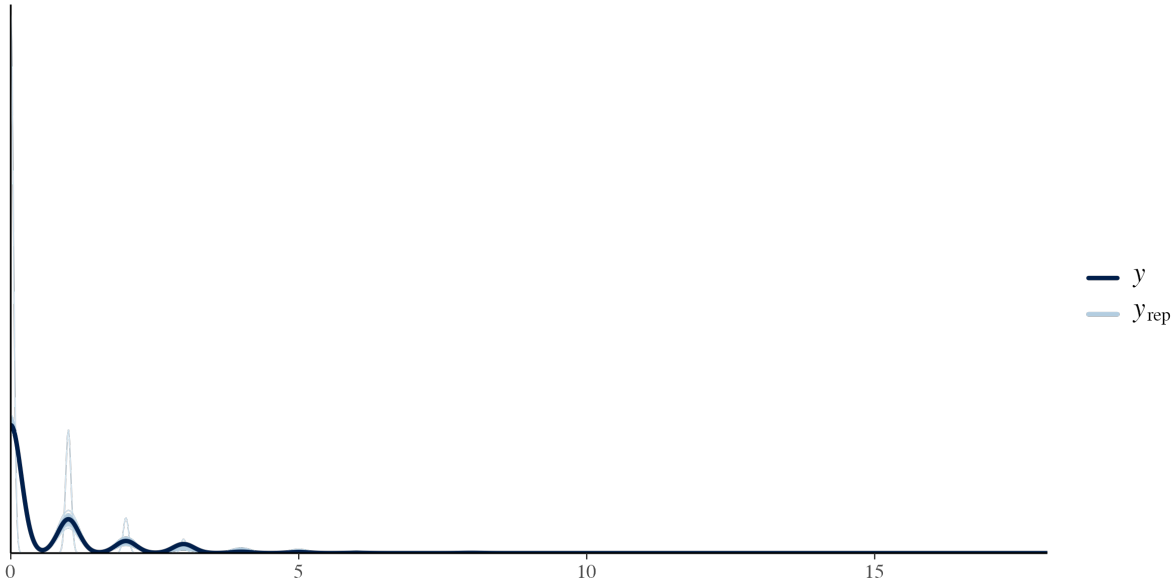

Figure 4: Team non-linear model PPC plot

Posterior predictive checks (PPCs) were conducted to assess how well each model captures the observed data. For each model, 100 replicated datasets ($yrep$) were generated from the posterior predictive distribution and compared to the observed data (y). We use the function `pp_check()` with `type = "dens_overlay"` to generate overlaid density plot for comparison between replicated and observed data. Figure 4 shows a resulting plot from the team non-linear model. PPC plots from other models are displayed in Appendix C because they do not fit in the report text. Overall, the replicated datasets from all models cover the range of the observed data (0–8) and even exceed it. The replicated data extends up to 20 for Nonlinear Team Model, up to 30 for Player Non-linear Model, up to 60 for Linear Team Model, and up to over 90 for Player Linear Model. This indicates that all models are capable of generating data within and beyond the observed range,

but the nonlinear models (especially the team model) show better control over the upper bounds.

All models capture the shape of the observed distribution reasonably well. The solid line representing the observed data lies within the shaded region (replicated datasets) for $y = 0, 1, 2$ in all models, indicating a good fit for these values. For larger values, the fit is not as good. For $y = 3$ specifically, the solid line (observed data) is higher than the shaded area (replicated datasets) in all models, but the discrepancy is smaller for the nonlinear models compared to the linear models. This suggests that while all models struggle to fully capture the frequency of higher values ($y \geq 3$), the nonlinear models perform better in this regard. This also indicates that the nonlinear team model has the best fit to the observed data overall, and is most suitable for inference and prediction.

5.3 Model Comparison

Models	**elpd_diff**	**se_diff**
Team Non-linear	0.0	0.0
Player Non-linear	-11.6	4.2
Team Linear	-26.2	6.2
Player Linear	-39.9	7.0

Table 2: Leave-One-Out Cross-Validation (LOO-CV) comparison between our models

Table 2 shows the output of the function `loo_compare()`, which measures the out-of-sample predictive ability of our models. The Team Non-linear shows to have the best predictive ability with the highest expected log predictive density (`elpd`). It significantly outperforms the other models, as all the `elpd_diff` are greater than twice their `se_diff`. The second, third, and last place go to Player Non-linear, Team Linear, and Player Linear respectively. This implies two points: non-linear terms yield a much better fit to the observed data compared to linear terms; and "*team*" random effect allows the model to capture the complexity of the data more effectively. The comparison result also strongly support our findings in Subsection 5.2.

5.4 Prior Sensitivity Analysis and Prior Predictive Checks

We conducted a prior sensitivity analysis by refitting the Team Non-linear Model with wide priors and narrow priors. For the intercept, we use `normal(0, 20)` and `normal(0, 1)` as wide and narrow prior respectively. For the others, we use `normal(0, 10)` and `normal(0, 1)` as wide and narrow priors respectively. Forest plots were generated using `mcmc_intervals()` to compare the posterior estimates for our parameters across the original, wide, and narrow prior models. All the generated plots can be found in Appendix D. The plots show that the posterior estimates for `b_Intercept`, `sattempts_on_target_1`,

and `sminutes_played_1` are quite similar across models with different priors. This means that there is little to no sensitivity, and the originally chosen priors are good enough. However, there is a big difference between the posterior estimates for the shape parameter. The narrow prior produces a small and short-ranged estimation while the original and wide priors produces much bigger and long-ranged value. This suggest that the shape parameter is not well informed by the data and strongly influenced by the prior.

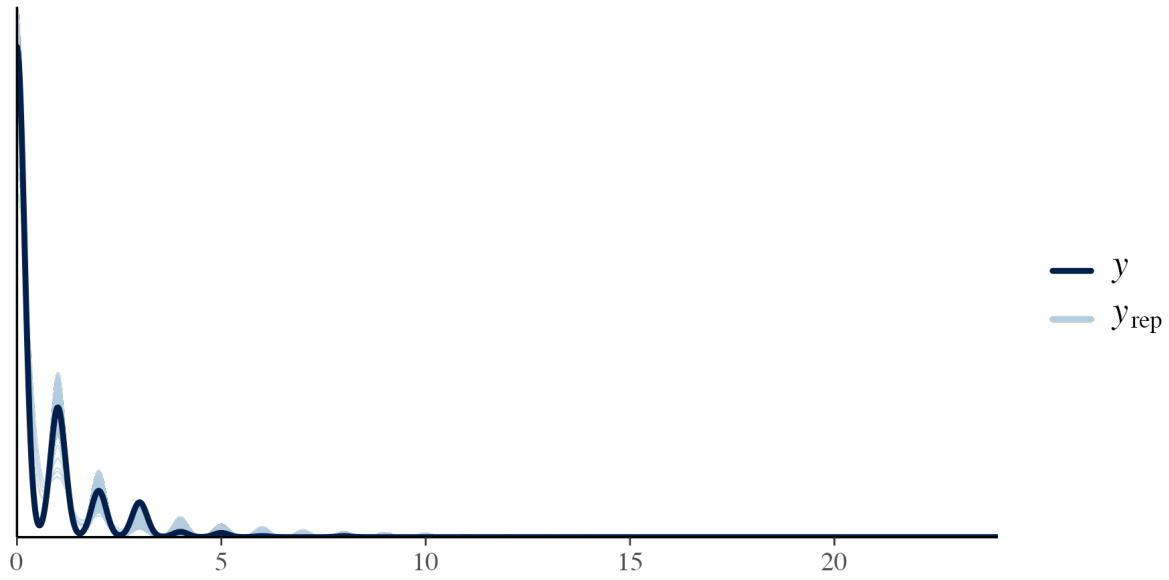

Figure 5: Prior predictive check for team non-linear model.

We also conduct prior predictive checks to assess if the chosen priors for team non-linear model are reasonable. We use the function `posterior_predict` with `sample_prior = "only"` to get the prior prediction, and Figure 5 shows the output density overlay plot from the `pp_check()` function. It is clear that the simulated data match the heavily right-skewed shape of the observed data. The simulated data closely matches the observed data for low values ($y = 0, 1, 2$). This means that the priors have the ability to capture the most common data points. However, the model slightly underestimated the occurrence of $y = 3$, as the solid line is on the upper edge of the shaded area. Moreover, while the maximum value of the observed data is 8, values up to 25 were generated by the simulated data. This indicates that the priors are not informative for higher values, and therefore could not handle overdispersion well. To address these issues, we need further experiments to pick a more informative prior.

6 Conclusion

In conclusion, this report demonstrates the use of Bayesian models to predict player goal contributions (goals + assists) in the UEFA Champions League 2025 dataset in Kaggle based on their performance. The key covariates considered include attempts on target, player position, minutes played, and team. There are two type of models that was used to predict: Player Model and Team Model, and each of them has two variants: Linear Model and Nonlinear Model. All four models are negative binomial regression models and use different set of priors. The priors follow either a normal or half-normal distribution with a mean of 0 and a variance of 5 or 10. After fitting, the estimated parameters show that `attempts_on_target` and `minutes_played` both have positive effect on the log of goal contribution. It is also shown that including non-linear terms and `team` random effect better accounts for overdispersion. We conducted convergence diagnostics, and found out that the Team Non-linear model have the best convergence out of all tested models. We also did posterior predictive checks, and it shows that Team Non-Linear model has the best fit. We compare the predictive ability of our models using leave-one-out cross-validation, and Team Non-liniear also comes out to be the best-performing model. Our prior sensitivity analysis and prior predictive checks implies that the prior for the `shape` parameter is quite sensitive and that the priors are not informative enough to handle overdispersion. To tackle this issue, we need to conduct more experiments to find out a better prior, or maybe gather more data.

During the course of this project, we have learned how to process raw data, implement Bayesian models and analyse the results. One limitation encountered in this report is the unbalanced and incomplete dataset. This is due to the dataset being retrieved three months into a nine-month event, which means it doesn't fully reflect the final stats. As new players cannot join mid-event, the goal contribution distribution is skewed and would likely become more balanced as the event progresses. Additionally, an improvement could be made by incorporating historical data from previous seasons or leagues outside of UEFA, although this is beyond the scope of this project. However, we would love try that in our future project.

A Box plots

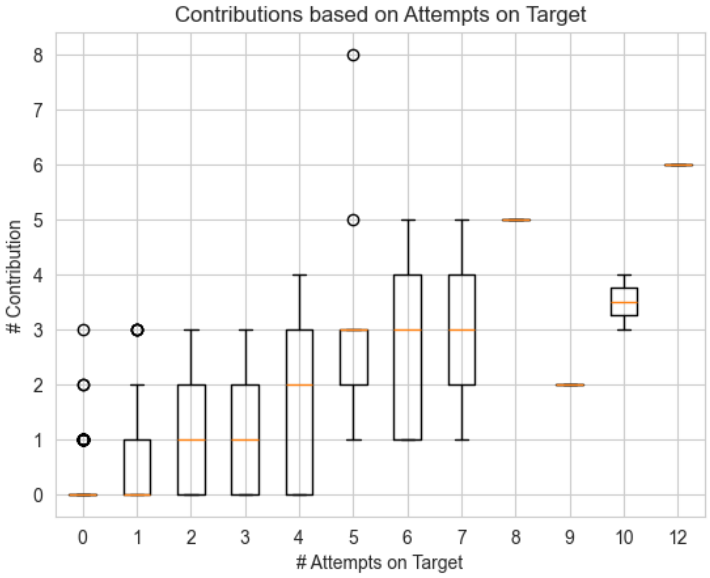

Figure 6: Box plot to depict the distribution of attempts on target with regard to goal contribution variable.

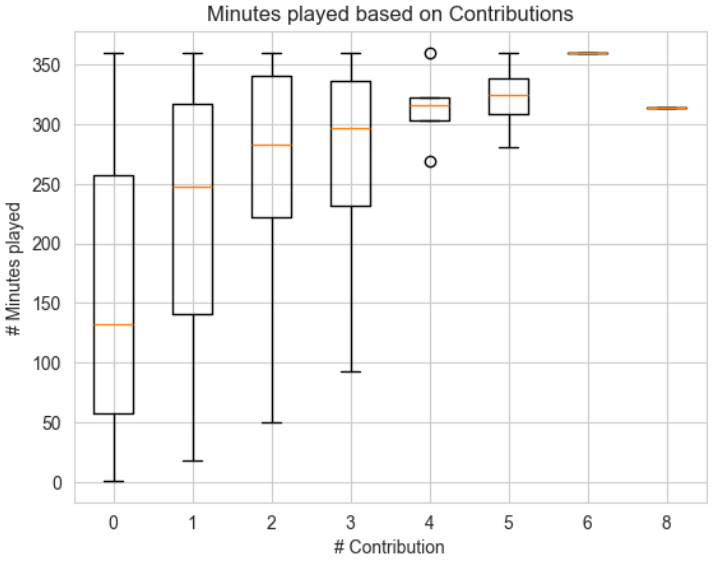

Figure 7: Box plot to depict the distribution of goal contribution with regard to minutes played variable.

B Models summaries

```
> summary(model_team_nonlin)
 Family: negbinomial
  Links: mu = log; shape = identity
Formula: goal_contribution ~ s(attempts_on_target, k = 5) + s(minutes_played, k = 5) + (1 | field_position) + (1 | team)
   Data: data1 (Number of observations: 676)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Smoothing Spline Hyperparameters:
                         Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sds(sattempts_on_target_1)     7.28      2.31     3.68    12.67 1.00     3884     2946
sds(sminutes_played_1)         1.84      1.68     0.07     6.22 1.00     1361     1716

Multilevel Hyperparameters:
~field_position (Number of levels: 3)
              Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sd(Intercept)     0.54      0.81     0.02     2.80 1.00     1038     1077

~team (Number of levels: 35)
              Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sd(Intercept)     0.38      0.10     0.18     0.59 1.00     1383     1330

Regression Coefficients:
                     Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept               -1.35      0.47    -2.27    -0.37 1.00     1521     1062
sattempts_on_target_1   16.23      4.78     6.56    25.88 1.00     3657     3191
sminutes_played_1        4.73      2.31     1.40    10.38 1.00     1934     2318

Further Distributional Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
shape    10.17      2.81     5.49    16.33 1.00     5865     2586

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

Figure 8: Team Non-linear model summary

```
> summary(model_team_lin)
 Family: negbinomial
  Links: mu = log; shape = identity
Formula: goal_contribution ~ attempts_on_target + minutes_played + (1 | field_position) + (1 | team)
   Data: data1 (Number of observations: 676)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Multilevel Hyperparameters:
~field_position (Number of levels: 3)
              Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sd(Intercept)     1.01      1.17     0.15     4.51 1.00      652      751

~team (Number of levels: 35)
              Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sd(Intercept)     0.43      0.10     0.26     0.65 1.00     1237     2109

Regression Coefficients:
                   Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept             -2.33      0.78    -3.91    -0.65 1.01      798      601
attempts_on_target     0.26      0.03     0.20     0.33 1.00     3284     2852
minutes_played         0.00      0.00     0.00     0.01 1.00     4655     3213

Further Distributional Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
shape     7.87      2.67     3.75    14.07 1.00     4201     2871

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

Figure 9: Team Linear model summary

```
> summary(model_indiv_nonlin)
 Family: negbinomial
  Links: mu = log; shape = identity
Formula: goal_contribution ~ s(attempts_on_target, k = 5) + s(minutes_played, k = 5) + (1 | field_position)
   Data: data1 (Number of observations: 676)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Smoothing Spline Hyperparameters:
                          Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sds(sattempts_on_target_1)     7.18      2.25     3.57    12.25 1.00     2091     2472
sds(sminutes_played_1)         1.86      1.65     0.08     6.20 1.00     1176     1553

Multilevel Hyperparameters:
~field_position (Number of levels: 3)
              Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sd(Intercept)     0.55      0.74     0.02     3.06 1.01      504      281

Regression Coefficients:
                     Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept               -1.31      0.50    -2.45    -0.45 1.01      603      216
sattempts_on_target_1   16.69      4.54     7.85    25.20 1.00     1930     2607
sminutes_played_1        4.63      2.37     1.31    10.36 1.01     1361     2316

Further Distributional Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
shape     8.64      2.75     4.32    14.69 1.00     3858     2868

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

Figure 10: Player Non-linear model summary

```
> summary(model_indiv_lin)
 Family: negbinomial
  Links: mu = log; shape = identity
Formula: goal_contribution ~ attempts_on_target + minutes_played + (1 | field_position)
   Data: data1 (Number of observations: 676)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Multilevel Hyperparameters:
~field_position (Number of levels: 3)
              Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sd(Intercept)     0.99      1.14     0.13     4.48 1.00      607      652

Regression Coefficients:
                   Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept             -2.32      0.84    -4.53    -0.88 1.00      945      492
attempts_on_target     0.28      0.04     0.22     0.36 1.00     1496     1619
minutes_played         0.00      0.00     0.00     0.01 1.00     3103     2946

Further Distributional Parameters:
      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
shape     5.03      1.95     2.42     9.76 1.00     1774     2123

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

Figure 11: Player Linear model summary

C Posterior Predictive Checks plots

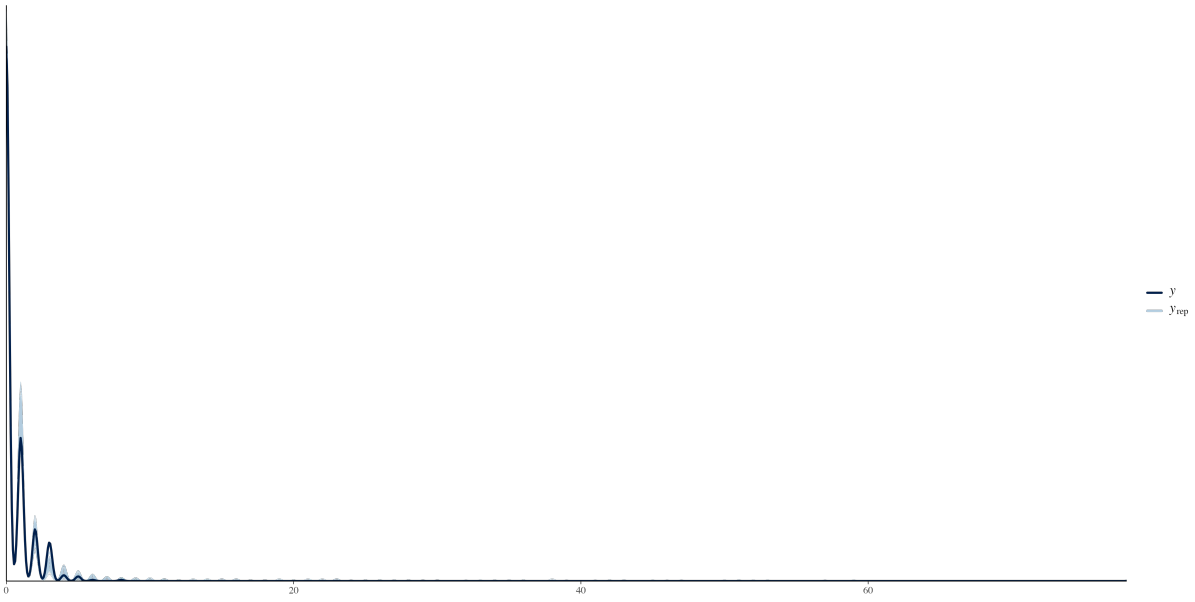

Figure 12: Player Linear model PPC plot

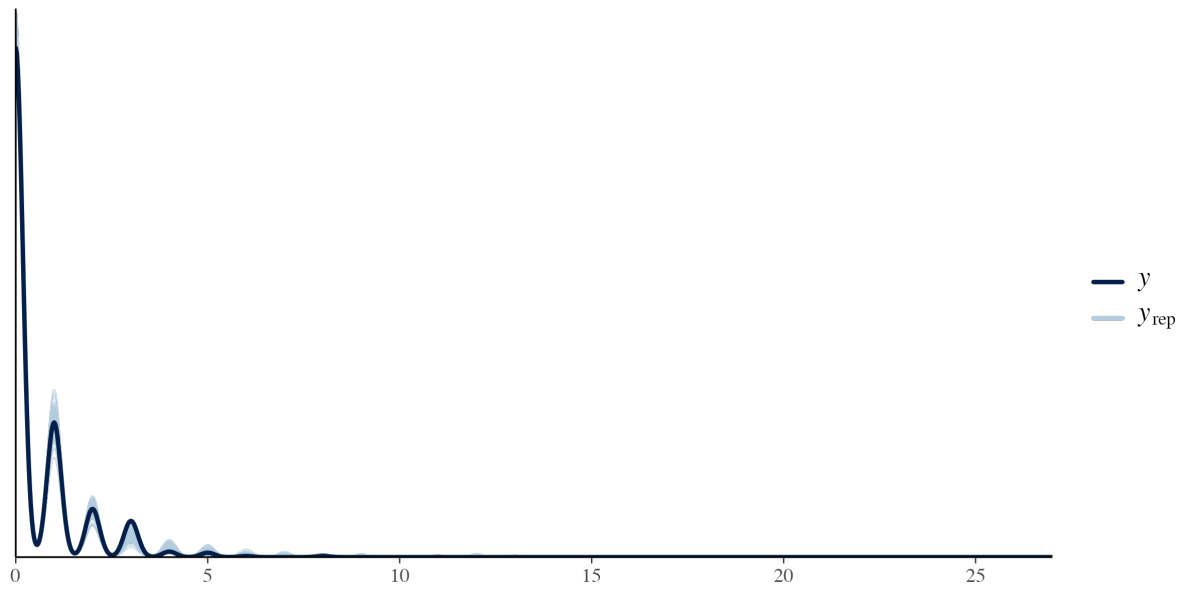

Figure 13: Player Non-inear model PPC plot

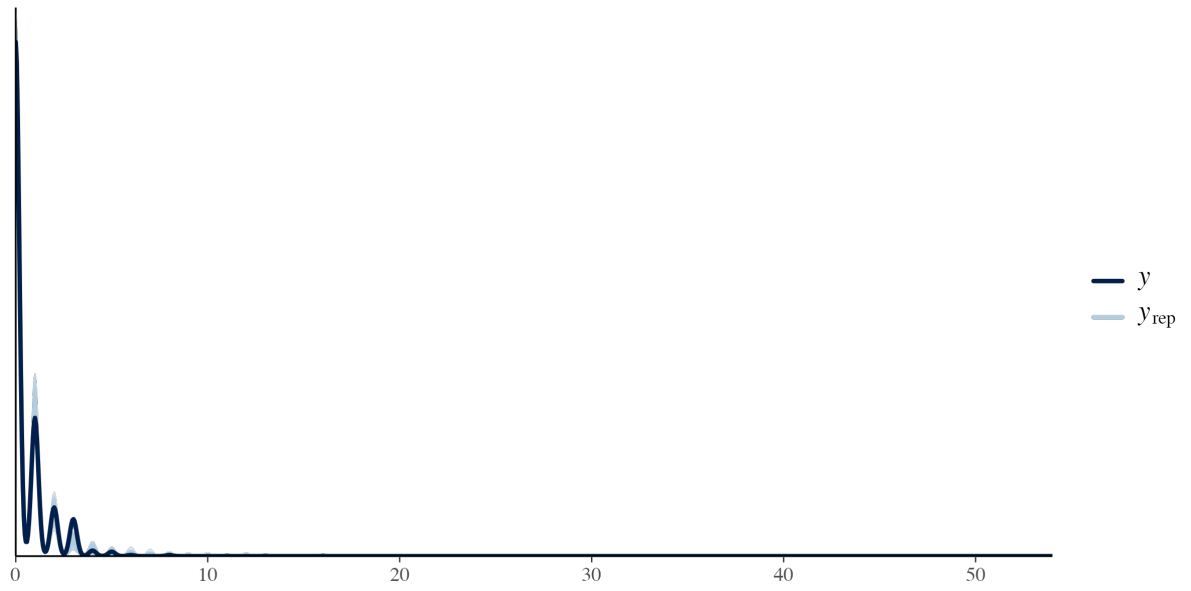

Figure 14: Team Linear model PPC plot

D Prior Sensitivity Analysis plots

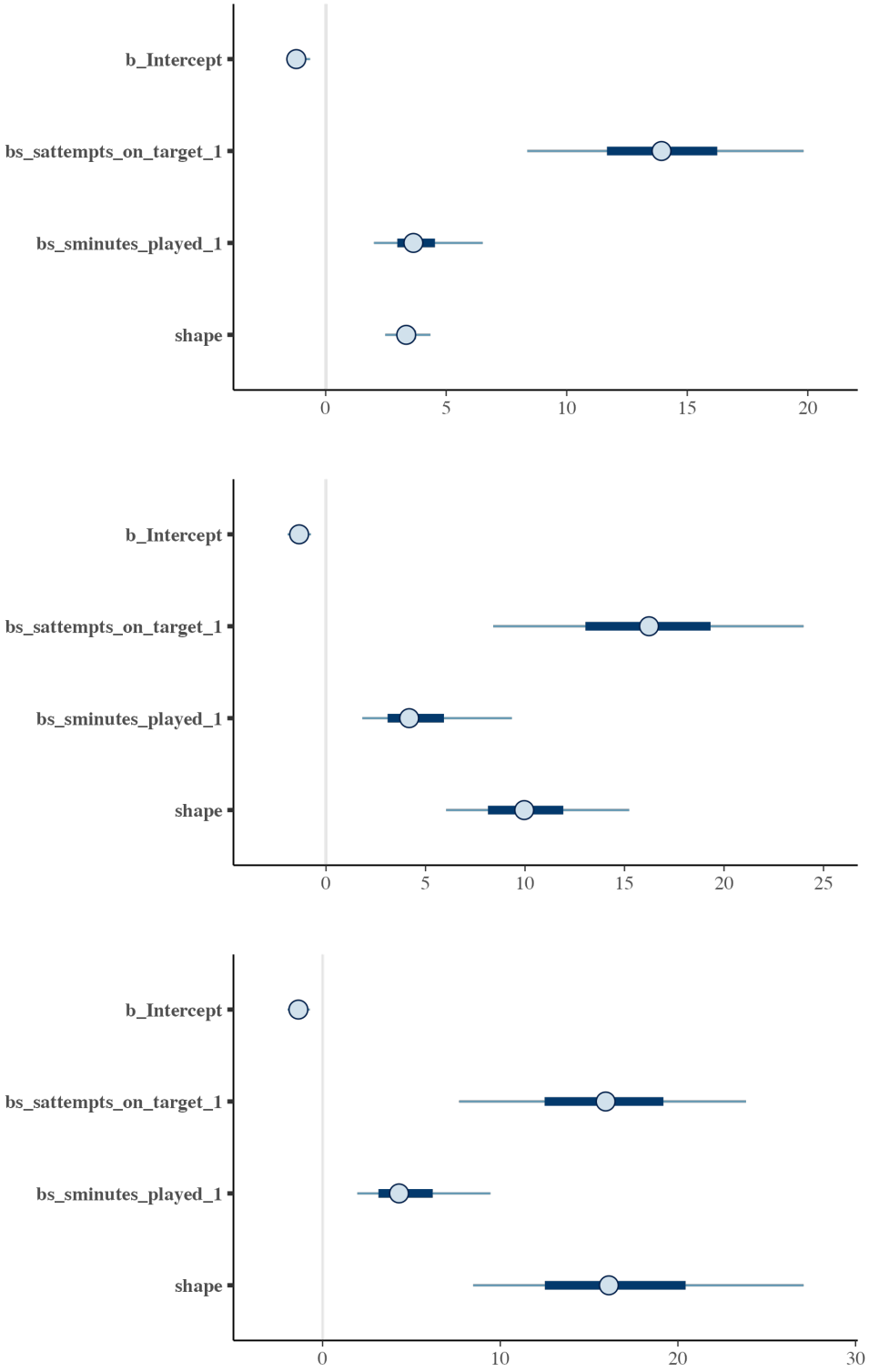

Figure 15: MCMC intervals plot of team non-linear model with tight (top), original (middle), and wide (bottom) prior

References

Bürkner, P.-C. (2017). Brms: An R package for bayesian multilevel models using stan. *J. Stat. Softw.*, *80*(1). https://doi.org/10.18637/jss.v080.i01

Wilkins, P. R. (2024). UEFA Champions League 2025. https://www.kaggle.com/datasets/pabloramoswilkins/ucl-2025-players-data/data%7D