The School of Mathematics

THE UNIVERSITY
of EDINBURGH

Predicting basketball games using Integrated Nested Laplace Approximation

by

Muxing Wang

Dissertation Presented for the Degree of
MSc in Statistics and Operational Research

August 2022

Supervised by
Dr Daniel Paulin

Abstract

In this thesis, we fit both Poisson models and logistic models to predict the scores and outcomes of basketball games. The dataset we used includes games from season 2004 to season 2020. In addition, we considered both fixed effects features and random effect features, and we proposed candidate models from simple to complex. The parameter estimation technique we used was the integrated nested Laplace approximation, which is efficient and accurate for Bayesian inference and we used it to obtain posterior marginal distributions of the parameters. Furthermore, we compared the models based on various criteria, including ranked probability score, accuracy, deviance information criterion, conditional predictive ordinate, predictive integral transform, and marginal log-likelihood, and we found that in general, models with random effects have better performance than models without random effects. We also compared the time cost of each candidate model and we found that for models without random effects, Poisson regression is slower than the logistic regression, however, the logistic models took more than twice as long as Poisson class models.

Acknowledgments

I want to express my deepest gratitude to Dr. Daniel Paulin, who supervised my dissertation for over three months. He provided numerous fantastic ideas and he has always been patient with my questions.

University of Edinburgh – Own Work Declaration

This sheet must be filled in, signed and dated - your work will not be marked unless this is done.

Name: Muxing Wang
Matriculation Number: S2201749
Title of work: Predicting basketball games using Integrated Nested Laplace Approximation

I confirm that all this work is my own except where indicated, and that I have:

- Clearly referenced/listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Not sought or used the help of any external professional academic agencies for the work
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Complied with any other plagiarism criteria specified in the Course handbook

I understand that any false claim for this work will be penalised in accordance with the University regulations (`https://teaching.maths.ed.ac.uk/main/msc-students/msc-programmes/statistics/data-science/assessment/academic-misconduct`).

Signature
Muxing Wang

Date 8/24/2022

Contents

List of Tables

List of Figures

1 Introduction

Sports prediction has always been one of the most popular topics. Sports journalists, betting practitioners, or the general public will make their own predictions when important games such as the World Cup and the National Basketball Association (NBA) playoffs come. Sporting event forecasting is not only interesting but also lucrative. In 2021, the value of the global betting market has been estimated to reach USD 76.75 billion [8], and the sports types with the top 3 highest market shares are football, basketball, and horse racing. In the United States, basketball betting has the largest market size. The whole sports betting market is anticipated to have a compound annual growth rate of 10.2% during the period from 2022 to 2030 and the revenue is predicted to reach USD 182.12 billion by 2030. This represents the prosperity of the gambling industry and people's love for predicting sports events.

The way of prediction often leads to different prediction results. Many people make predictions based on their own preferences, while others are based on rational analysis. Before the invention of computers, it was difficult for even rational forecasters to make predictions using data, however, with the improvement of computational power and the development of data science, people are more and more inclined to use models and algorithms to make predictions, and the pursuit of higher accuracy has become the driving force for people to optimize the models. Under such a circumstance, more high-quality datasets are available, and more advanced models and algorithms have been proposed and implemented to tackle this problem.

On the other hand, the prediction and analysis of sports events will also have a huge role in promoting the progress of the sports industry. For example, data analysis can reveal the strengths and weaknesses of the teams or players, so as to help players improve their skills or help coaches develop better tactics. In this way, a large number of sports will have higher-level players and teams, greatly increasing the viewing of the game. In addition, the interpretable models can provide objective evaluations of players, reduce the excessive accusations of fans against players after losing the game, and make the whole industry more harmonious. Therefore, it is meaningful to develop more accurate and interpretable models.

As for NBA, it is a men's professional basketball league composed of 30 professional basketball teams in North America. It is one of the four major professional sports leagues in the United States and the top basketball league in the world. It was founded on June 6, 1946, as the Basketball Association of America. After decades of development, the NBA has become one of the most popular sports leagues in the world, with 70 million followers on Instagram. In addition, the NBA has numerous superstars such as Michael Jordan, Kobe Bryant, LeBron James, and many more. These superstars have the ability to dominate the game and provide the spectators with great watching experiences, and their personal influence is also one of the most important reasons for the popularity of the NBA.

In this thesis, Poisson class models and logistic class models were fit in the way of the Bayesian approach to predict the scores and outcomes of playoffs of NBA and then the models were compared based on various criteria. The rest of this thesis is structured as follows: Section 2 reviews the previous research in this field, Section 3 will introduce the NBA system, the dataset source and schema, the preprocessing procedure of the dataset, the response variables, and the meaning of each feature and how they were extracted. In addition, a simple exploratory data analysis of the win percentage of each team in different seasons will be included as well. Section 4 will present the structure of models, introduce the Integrated Nested Laplace Approximation (INLA) technique[22], discuss the priors, and list the candidate models we proposed. It also includes the formula derivation of the winning probabilities of home teams using the linear predictors. In Section 5, the ranked probability score (RPS) [12] will be introduced first, including the definition, interpretation, properties, and how to compute the expectation and variance of the mean RPS. Then, we will introduce the test framework and compare models' performances in many aspects, including flexibility, stability, time cost, validity, and some other Bayesian model checking criteria. Section 6 consists of the conclusion of the experiments, study limitations, and a discussion of future work.

The codes written for this thesis can be found at `https://github.com/leafstar/OutcomeBasketball`, including data preprocessing steps with `Python` and the model fitting with `R`.

2 Literature review

Many scholars have studied how to predict sports events previously because the topic is very popular, and in this section, we reviewed the existing literature in three main aspects. First, there are two main results of a sports game, one is the outcome of the game, that is, a variable indicating which team wins, and the other is the scores of the two competing teams in a game. Therefore, we reviewed literature that tried to model any of them. Moreover, we were interested in the application of the INLA technique in predicting sports games and reviewed the relevant research.

Scores were normally modelled by some probability distributions. Gill [13] used the normal distribution to model basketball scores and football scores, and Poisson distribution to model hockey scores. He assumed that the number of goals in ice hockey follows independent Poisson processes. Clauset et al. [19] also claimed that scoring events occur independently and can be modelled by a Poisson process. Karlis and Ntzoufras [16]replaced the independent assumption by applying a bivariate Poisson distribution to add some correlation to the scores of the two teams, and they extended it by taking an inflation factor into account to predict the draws in football games more accurately.

The outcome of a game is a more common topic since different sports have different scoring systems, but the result is always a win or a loss (or a draw). Accuracy, which is another metric derived from the outcomes, is also one of the most important criteria used to measure the models. Various models have been developed recently to improve prediction accuracy.

Miljković [20] collected 778 games of season 2009 and used the Naïve Bayes method to predict the outcomes. Based on the result of the 10-fold cross-validation, the performance was evaluated to be 67% in accuracy, which was close to the prediction accuracy of sports journalists from CBS.

Lam [17] proposed a predictive model called the two-layer Gaussian process regression model for winning probability prediction, which was able to correctly predict 85.28% of the matches in the 2014/2015 regular season in the NBA using the 2013/2014 regular season data as the training data.

Pai et al. [21] combined support vector machines and the decision tree to develop a new model called HSVMDT. They also utilized the correlation-based feature selection method to extract the features and they achieved an accuracy of 80% for predicting 80 games from season 2008 to 2010. In addition, HSVMDT can also make up for the lack of interpretability of SVM and can provide coaches with strategic advice.

Cheng et al. [11] applied the maximum entropy principle to predict the NBA playoffs from season 2008 to season 2014, during which there were 10,271 games in total. They achieved a maximum prediction accuracy of 74.4% for the playoff of the season 2007.

As for INLA, it is an efficient technique for estimating the posterior marginal distributions in Bayesian inference and many scholars utilized this method to obtain fast and accurate inferences. Cervone et al. [10] proposed a quantity called expected possession value to estimate the expected number of scores that can be obtained by a single possession based on optical player tracking data. They used four multiresolution transition models to form the framework to estimate the expected possession value, which consisted of the microtransition model, the macrotransition entry model, the macrotransition exit model and the transition probability matrix, and INLA was used for the parameter estimation. Tsokos [26] used five extensions of the Bradley-Terry model and a hierarchical Poisson log-linear model to predict the winning probabilities of soccer games from 52 leagues, and the parameter estimation methods were maximum log-likelihood estimation and INLA respectively. The performance of the best Bradley-Terry model was similar to that of the hierarchical log-linear Poisson model, with 0.5194 and 0.5485 in accuracy, respectively.

3 Data Preprocessing

3.1 Dataset Overview

Before exploring the dataset, we briefly introduce the NBA system.

The NBA currently has 30 teams, divided into the Eastern and Western Conferences, each with three divisions and a total of six divisions. Every season is divided into regular season and playoff. Each team plays 82 games in the regular season, after which the top eight teams in the Eastern and Western Conference will enter the playoff. Each team will obtain a seed based on its regular season ranking, with smaller seeds representing better teams. For example, the top-ranked team will get the 1st seed. The playoffs are a knockout format, with the first round of the Eastern and Western Conference being the 1st seed against the 8th, the 2nd against the 7th, the 3rd against the 6th, and the 4th against the 5th. The knockout stage continues until the Eastern and Western champions are decided. Then the two champions move on to the NBA Finals. Both the playoffs and the finals adopt a best-of-7 system, and the team with a higher regular season winning percentage will gain an additional home-court advantage.

The data is from a public dataset available on Kaggle[18]. It consists of regular games and playoffs of NBA matches from season 2003 to season 2021. The dataset contains part of common box scores [1] of NBA games, for example, the game's date, the home team, the away team, and the total points scored by the home/away team. Apart from the details of each game, the dataset also contains information about each team, such as the stadiums and head coaches of the teams. On top of these data, we collected the salaries of each team in each season and the travel distance between teams. Salary data came from hoopshype[4], and past years' salaries were adjusted for inflation based on the data provided by the U.S. Department of Labor Bureau of Labor Statistics[3].

There were two variables that we considered as response variables, namely, scores of each team in a match, and a binary indicator of if the home team wins. In this study, the logistic model was used to estimate the binary outcome by directly estimating the logit of winning probability, while the Poisson regression was used to estimate the scores.

Exploratory data analysis

We computed the win rate of each team using all the data, and the results are shown in the bar chart 1. It can be seen that the Spurs has the highest win rate of over 0.6, which is significantly higher than all other teams, while the Timberwolves has the lowest win rate of less than 0.4. Moreover, the win percentage of the home team for all games is $\frac{14355}{24411} \approx 58.8\%$. For each season, the win percentage of the home team can be found in Table 1. Clearly, there exists a home court advantage since all the win percentages of the home team are significantly greater than 50%, even greater than 60% for 7 out of 17 seasons.

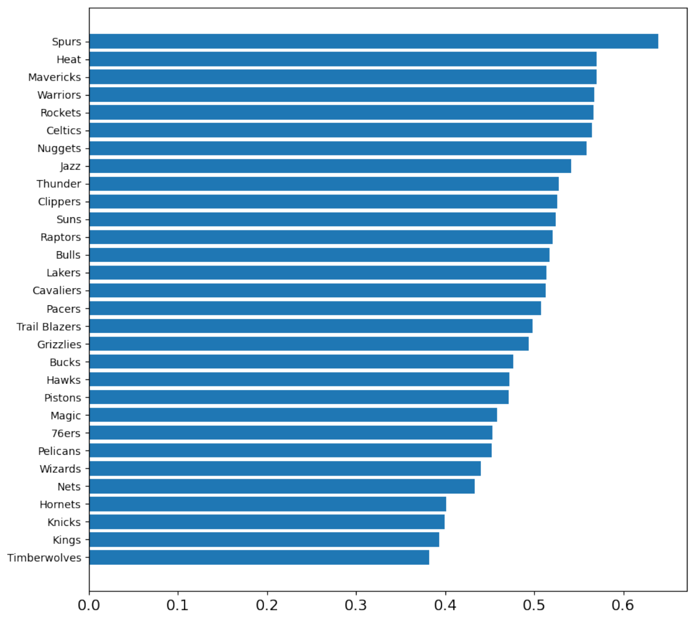
Table 1: Win percentage of the home teams

Season	the number of home wins	the number of total games	win percentage of the home team
2004	824	1362	60.5%
2005	866	1432	60.47%
2006	839	1419	59.13%
2007	862	1411	61.09%
2008	857	1425	60.14%
2009	857	1424	60.18%
2010	871	1422	61.25%
2011	656	1104	59.42%
2012	872	1420	61.41%
2013	819	1427	57.39%
2014	816	1418	57.55%
2015	841	1416	59.39%
2016	816	1405	58.08%
2017	805	1382	58.25%
2018	810	1378	58.78%
2019	666	1241	53.67%
2020	681	1249	54.52%

3.2 Response variable

Let G denote the number of games in our dataset. As described above, the response variable for Poisson models is the scores of both teams in a game and for logistic models, the response variable is the binary outcomes of the games. Moreover, since there are two scores and two binary outcomes in a single game, we split one game into two data instances, one for the home team and the other for the away team, and each data instance has one score and one binary outcome. Therefore, there are $2 \cdot G$ rows in the final dataset.

Figure 1: Win rate of all NBA teams over 17 seasons



3.3 Feature extraction

Although the dataset has already contained many features, few of them can be used directly, and data preprocessing was performed. First, season 2021 was still ongoing during the completion of this thesis, therefore season 2021 was completely removed. Season 2003 data was also incomplete and thus discarded. After deletion, there are 23,335 games in the dataset, that is 46,670 data instances since we split one game into two data instances. Moreover, when predicting the outcome of a game, only the information that is before the game can be used, therefore, instead of post-game statistics, running averages before the game were computed for each box score, and they are feature 4 to feature 9 in Table 2. Based on box scores, we computed two additional features, which are *winning rate* and *average points lost in current season* (features 10 and 11), and they are also the running average in the current season before the game. The abovementioned features can be regarded as the most up-to-date performance of the team.

Attack/Defend

We added the *Attack* and *Defend* covariates as categorical variables. In detail, a data instance has a score or binary outcome as the response variable, which is achieved by one of the home team and the away team. If it is achieved by the home team, then the home team is the attacking team and the away team is the defending team. In other words, for any data instance, the attacking team is the team who achieved the score and the outcome of the data instance.

Salary

It is commonly known that better teams often consist of better players and better players are always paid higher salaries. Therefore, we thought that salaries can measure the overall strength of a team. We used the salaries adjusted for inflation and transformed them by the log function due to the large magnitude of the salaries. For example, the lowest salary of a team is \$34,262,872 and the highest salary is \$180,330,388, and the corresponding log-scale values are 17.35 and 19.01, respectively.

Seasonal attacking and defending strength

It was assumed that the attacking and defending strengths were not fixed for different seasons, therefore we added random effects *seasonal attack* and *seasonal defend* (Feature 16 and 17 in Table 2) to capture this property. Without loss of generality, there are T teams and S seasons, and we created two sets *a_season* and *d_season* to enumerate all the combinations, that is

$$a\_season = \{a\_season_1, ..., a\_season_S, a\_season_{S+1}, ..., a\_season_{T \cdot S}\}$$

$$d\_season = \{d\_season_1, ..., d\_season_S, d\_season_{S+1}, ..., d\_season_{T \cdot S}\}$$

Every S elements were used for one team. For example, $\{a\_season_1, ..., a\_season_S\}$ are the seasonal attacking strength parameters for Team 1, $\{a\_season_{S+1}, ..., a\_season_{2S}\}$ are the seasonal attacking strength parameters for Team 2, and so on.

Travel distance

Travel distance is another feature that we considered relevant to the game outcome. The simple *Home* feature is not able to distinguish the difference between a long trip and a short trip for the away team. Long trips will cause a lack of sleep, having negative effects on the performance of players[15]. Especially since there are many back-to-back games in NBA, players' immediate air travel after the game will cause inadequate hydration. Therefore it seems reasonable to believe that long trips are negative for teams. We collected the longitude and latitude of the stadiums from the website [28] to estimate the travel distance.

Type

With the development of the NBA, some teams have become traditional giants, while some teams have never won a championship, and people usually have a prior belief in teams' strengths. Based on such facts, we classified teams into different levels based on the rankings of teams in the last season. And the *Type* feature will be constructed by concatenating the level of the home team and the away team. For example, if the home team is a strong team and the away team is a weak team for a game, the *Type* of the game will be "SW". This feature interacted with other features to make the model more flexible, and more details will be discussed when introducing M7 in Section 5.4.

The schema of the dataset we fed into the models can be found in Table 2, which lists short names, descriptions, and ranges of all features. Table 3 shows a data instance. In addition, features were categorized into mainly two sets: fixed effects and random effects.

Table 2: Features categorized into fixed and random effects

Features			
Number	Name	Description	Range
Fixed effects			
1	Attack	The team who achieved the score/outcome in this data instance	Categorical
2	Defend	The team playing against the attacking team in this data instance	Categorical
3	Home	1 if the team is the home team; 0 otherwise	{0,1}
4	Points per match	Average points scored in current season before current game	$\Re$
5	Field goal ratio	Average field goal ratio in current season before current game	[0,1]
6	Free throw ratio	Average free throw ratio in current season before current game	[0,1]
7	3-Point goal ratio	Average 3-point goal ratio in current season before current game	[0,1]
8	Assistance	Assistance per game in current season before current game	$\Re$
9	Rebound	Rebounds per game in current season before current game	$\Re$
10	Winning rate	Average winning rate of the home team in current season before current game	[0,1]
11	Points lost per match	Average points lost in current season before the current game	$\Re$
12	log(salary)	Total salaries (in log scale) of the home team in current season	$\Re$
13	Type	Strength category of the game, e.g. "SW" means Strong team versus Weak team	Categorical
14	Type:Attack	Interaction term between Type and Attack	Categorical
15	Type:Defend	Interaction term between Type and Defend	Categorical
Random effects			
16	Seasonal attack	The attacking strength of team i in season j	$\Re$
17	Seasonal defend	The defending strength of team i in season j	$\Re$
18	Travel distance	travel distance group of the away team	{1,...,25}

4 Models

We used both Bayesian Poisson regression and logistic regression models and for each model, we experimented with different combinations of features. As mentioned in Section 3.2, we split one game

Table 3: Data instances

Features			
Number	Name	Match 1	Range
Fixed effects			
1	Attack	TeamX	Categorical
2	Defend	TeamY	Categorical
3	Home	1	{0,1}
4	Points per match	94.82407	$\Re$
5	Field goal ratio	0.4419537	[0,1]
6	Free Throw ratio	0.6968704	[0,1]
7	3-Point goal ratio	0.3386389	[0,1]
8	Assistance	20.14815	$\Re$
9	Rebound	43.50926	$\Re$
10	Winning rate	0.5833	[0,1]
11	Points lost per match	112.21212	$\Re$
12	log(salary)	18.2505	$\Re$
13	Type	"SW"	Categorical
14	Type:Attack	"CavaliersTypeSW"	Categorical
15	Type:Defend	"SpursTypeWS"	Categorical
Random effects			
16	Seasonal attack	(index) 1	$\Re$
17	Seasonal defend	(index) 1	$\Re$
18	Travel distance	(group) 1	{1,...,25}

into two data instances and then the number of rows in the dataset equals to two times the number of games. Then, we modeled the response variables for the home team and the away team for each game by two independent random variables, and we let η_i^H denote the linear predictor for the home team and η_i^A for the away team in game i.

4.1 Poisson model

Let y_i^H denote the score of the home team of game i and y_i^A denote the score of the away team. Then, assume

$$y_i^H \sim \text{Poisson}(\lambda_i^H), y_i^A \sim \text{Poisson}(\lambda_i^A)$$

With the log link, we can write

$$\eta_i^H = \log(y_i^H) = \beta_0^H + \sum_{j=1}^{n_\beta} \beta_i^H x_{ji} + \sum_{k=1}^{n_f} f^{(k)}(z_{ki}),$$

$$\eta_i^A = \log(y_i^A) = \beta_0^A + \sum_{j=1}^{n_\beta} \beta_i^A x_{ji} + \sum_{k=1}^{n_f} f^{(k)}(z_{ki}),$$

where x_{ji} are fixed effect features, $(\beta_0^{A/H}, \beta_1^{A/H}, ..., \beta_{n_\beta}^{A/H})$ are the regression coefficients (fixed effects), and $f^{(k)}$'s are the random functions of some covariates z_{ki} (random effects).

Although the Poisson model was used to estimate the scores of a game, the winning probabilities can also be estimated. Once λ_i^H and λ_i^A are estimated, we can calculate the winning probability of the home team:

$$\begin{aligned} P(\text{Home team wins}) &= P(y_i^H > y_i^A | y_i^H \neq y_i^A) \\ &= \frac{P(y_i^H > y_i^A, y_i^H \neq y_i^A)}{P(y_i^H \neq y_i^A)} \\ &= \frac{P(y_i^H \neq y_i^A | y_i^H > y_i^A) P(y_i^H > y_i^A)}{1 - P(y_i^H = y_i^A)} \\ &= \frac{1 \cdot P(y_i^H > y_i^A)}{1 - P(y_i^H = y_i^A)} \\ &= \frac{P(y_i^H > y_i^A)}{1 - P(y_i^H = y_i^A)} \end{aligned} \tag{4.1}$$

The numerator in equation (4.1) can be written as

$$\begin{aligned} P(y_i^H > y_i^A) &= \sum_{k=0}^{\infty} P(y_i^H > k, y_i^A = k) \\ &= \sum_{k=0}^{\infty} P(y_i^H > k) P(y_i^A = k) \\ &= \sum_{k=0}^{\infty} (1 - P(y_i^H \leq k)) P(y_i^A = k) \\ &= \sum_{k=0}^{\infty} (1 - \sum_{j=0}^{k} \frac{e^{-\lambda_i^H} (\lambda_i^H)^j}{j!}) \frac{e^{-\lambda_i^A} (\lambda_i^A)^k}{k!} \end{aligned} \tag{4.2}$$

and the denominator in equation (4.1) can be written as

$$\begin{aligned} 1 - P(y_i^H = y_i^A) &= 1 - \sum_{k=0}^{\infty} P(y_i^H = k, y_i^A = k) \\ &= 1 - \sum_{k=0}^{\infty} P(y_i^H = k) P(y_i^A = k) \\ &= 1 - \sum_{k=0}^{\infty} \frac{e^{-\lambda_i^H} (\lambda_i^H)^k}{k!} \frac{e^{-\lambda_i^A} (\lambda_i^A)^k}{k!} \\ &= 1 - \sum_{k=0}^{\infty} \frac{e^{-\lambda_i^H - \lambda_i^A} (\lambda_i^H \lambda_i^A)^k}{(k!)^2} \end{aligned} \tag{4.3}$$

Therefore, the winning probability of the home team is

$$P(\text{Home team wins}) = \frac{\sum_{k=0}^{\infty} (1 - \sum_{j=0}^{k} \frac{e^{-\lambda_i^H} (\lambda_i^H)^j}{j!}) \frac{e^{-\lambda_i^A} (\lambda_i^A)^k}{k!}}{1 - \sum_{k=0}^{\infty} \frac{e^{-\lambda_i^H - \lambda_i^A} (\lambda_i^H \lambda_i^A)^k}{(k!)^2}} \tag{4.4}$$

And then, if $P(\text{Home team wins}) > 0.5$, we can make the prediction that the home team wins.

Moreover, Skellam distribution [7] was invented to describe such difference between Poisson random variables and in the implementation, we used the `dskellam` method from the `R` package `extraDistr` [27] to calculate the winning probabilities for simplicity.

4.2 Logistic model

The logistic model is usually used to predict a binary outcome, and it also models the probability of one event occurring. However, in this thesis, we did not use the standard logistic regression for two main reasons. First, since the model would be compared with the Poisson model, we would like to make the structure of the model analogous to the Poisson model, which has two independent response variables for a single game. In addition, we also tried to fit the standard logistic regression but the performance was significantly worse, and most of the RPSs were greater than 0.25. Therefore, the two response variables were treated as independent and the model can be formulated as:

$$\eta_i^H = \text{logit}(\pi_i^H) = \log(\frac{\pi_i^H}{1-\pi_i^H}) = \beta_0^H + \sum_{j=1}^{n_\beta} \beta_i^H x_{ji} + \sum_{k=1}^{n_f} f^{(k)}(z_{ki})$$

$$\eta_i^A = \text{logit}(\pi_i^A) = \log(\frac{\pi_i^A}{1-\pi_i^A}) = \beta_0^A + \sum_{j=1}^{n_\beta} \beta_i^A x_{ji} + \sum_{k=1}^{n_f} f^{(k)}(z_{ki})$$

where π_i^H and π_i^A denote the winning probabilities of the home team and the away team, and the rest of the parameters have the same meaning as in the Poisson model. Once η_i^H and η_i^A are estimated, $\pi_i^H = \text{expit}(\eta_i^H) = \frac{1}{1+e^{-\eta_i^H}}$ and $\pi_i^A = \text{expit}(\eta_i^A) = \frac{1}{1+e^{-\eta_i^A}}$ can be computed subsequently. Since η_i^H and η_i^A are independent variables, they do not necessarily add up to 1, and normalization is needed. That is,

$$\tilde{\pi}_i^H = \frac{\pi_i^H}{\pi_i^H + \pi_i^A}$$

$$\tilde{\pi}_i^A = \frac{\pi_i^A}{\pi_i^H + \pi_i^A}$$

where $\tilde{\pi}_i^H$ and $\tilde{\pi}_i^A$ stand for the normalized probabilities. Then, if $\tilde{\pi}_i^H > 0.5$, the home team is predicted to win.

For simplicity, this paper will refer to this model as the logistic model.

4.3 Priors

As for the priors, we experimented with several prior distributions for the fixed effects but none of them made significant differences in terms of RPS. Therefore, we used the default priors for the fixed effects provided by `R-INLA` [23]. As for the random effects, we found that tuning the prior for the attacking and defending random effect precision does have an effect on how much variability is there between seasons. Moreover, For the travel distance random effect, we used the default prior.

4.4 Estimation

INLA is efficient for approximating the marginal posterior distributions of latent Gaussian models if the parameters are assumed to be Gaussian Markov random fields (GMRFs). The main advantage of INLA is that it avoids MCMC sampling which takes a much longer time to converge.

Moreover, we used INLA to obtain the posterior predictive distribution for each of the above-mentioned linear predictors. Then we drew 1000 samples from each of these posterior predictive distributions and used the sample means as the posterior mean of the linear predictors.

4.5 Candidate models

After specifying the features and the model structures, we can propose the following candidate models and estimate the effects of each term. Note that y stands for the scores of the games when fitting the Poisson models, while it stands for the binary outcome when fitting the logistic models.

- (M1) Baseline model: $y \sim$ Home

- (M2) M1 + Attack + Defend: $y \sim \text{Home} + \text{Attack} + \text{Defend}$
- (M3) M2 + Box scores: $y \sim \text{Home} + \text{Attack} + \text{Defend} + \text{Box Scores}$
- (M4) M3 + log(Salary): $y \sim \text{Home} + \text{Attack} + \text{Defend} + \text{Box Scores} + \log(\text{Salary})$
- (M5) M4 + f(Seasonal Attack)+f(Seasonal Defend): $y \sim$ Home+Attack+Defend+Box Scores+ log(Salary) + f(Seasonal Attack) + f(Seasonal Defend)
- (M6) M5 + f(Travel distance): $y \sim$ Home + Attack + Defend + Box Scores + log(Salary) + f(Seasonal Attack) + f(Seasonal Defend) + f(Travel Distance)
- (M7) M6 + interaction terms: $y \sim$ Home + Attack + Defend + Box Scores + log(Salary) + f(Seasonal Attack) + f(Seasonal Defend) + f(Travel Distance) + Type : Attack + Type : Defend

5 Validation and Results

Accuracy has always been the main criteria for prediction tasks, however, it cannot measure how accurate the predicted probabilities are. For example, if the observation is the home team wins, then both $P(\text{Home team wins}) = 0.51$ and $P(\text{Home team wins}) = 0.99$ will be classified as 1 (a home win), but 0.99 is more accurate than 0.51. However, the RPS is able to measure how good forecasts are in matching the observations. Therefore, we used the RPS as the main metric in this thesis.

5.1 Ranked Probability Score

The RPS is defined as:

$$RPS = \frac{1}{r-1}\sum_{i=1}^{r-1}\{\sum_{j=1}^{i}(p_j - a_j)\}^2$$

where r is the number of possible outcomes, p_j's are the predicted probabilities and a_j's are the observed outcomes. Note that $\sum_{i=1}^{r} p_i = 1$ and $\sum_{i=1}^{r} a_i = 1$. Lower RPS indicates better performance.

Furthermore, there are only two possible outcomes of a basketball game, so $r = 2$, and then the RPS can be simplified as

$$RPS_i = (p_i - a_i)^2 \text{ for game } i. \tag{5.1}$$

Interpretation of RPS

Suppose X_i is a Bernoulli random variable with success probability p_i. That is $P(X_i = 1) = p_i$ and $P(X_i = 0) = 1 - p_i$. Without loss of generality, let X_i be the outcome of game i (i.e. $X_i = 1$ indicates the home team wins and $X_i = 0$ otherwise). Now suppose for game i, we estimated the winning probability of the home team as q_i based on one of our models, then $RPS_i = (q_i - X_i)^2$.

$$\begin{aligned}
E(RPS_i) &= E((q_i - X_i)^2)\\
&= E(q_i^2 - 2q_iX_i + X_i^2)\\
&= E(q_i^2) - 2E(q_iX_i) + E(X_i^2)\\
&= q_i^2 - 2q_iE(X_i) + E(X_i^2)\\
&= q_i^2 - 2q_ip_i + Var(X_i) + E(X_i)^2\\
&\quad \text{since } E(X_i) = p_i \text{ and } Var(X_i) = E(X_i^2) - E(X_i)^2\\
&= q_i^2 - 2q_ip_i + p_i(1-p_i) + p_i^2\\
&= q_i^2 - 2q_ip_i + p_i\\
&= p_i(1-p_i) + (q_i - p_i)^2 \qquad (5.2)
\end{aligned}$$

Then, it is obvious that

$$E(RPS_i)_{min} = p_i - p_i^2 \text{ when } q_i = p_i \tag{5.3}$$

From (5.2), we can see that if $q_i - p_i$ changes by 0.1, the change of $E(RPS_i)$ will be 0.01. In other words, a 0.01 difference in the expected RPS is equivalent to a 10% difference in accuracy. Then, we can set some significance levels as follows:

$$\begin{aligned}
\Delta_{q-p} &= 1\% \implies \Delta_{rps} = 0.0001, \text{ not significant} \\
\Delta_{q-p} &= 5\% \implies \Delta_{rps} = 0.0025, \text{ significant} \\
\Delta_{q-p} &= 10\% \implies \Delta_{rps} = 0.01, \text{ very significant}
\end{aligned} \tag{5.4}$$

Moreover, since we used 0.5 as the threshold to determine the binary outcomes based on the winning probabilities, then p_i is greater than 0.5 if $X_i = 1$, and p_i is less or equal than 0.5 if $X_i = 0$. If we correctly predict the outcome of a game, there are two possibilities, which are $p_i > 0.5, X_i = 1$, and $p_i < 0.5, X_i = 0$. For both cases, $RPS_i = (p_i - X_i)^2 \in (0, 0.25)$ since $p_i - X_i \in (0, 0.5)$. Similarly, two probabilities are $\pi_i \leq 0.5, X_i = 1$ and $\pi_i > 0.5, X_i = 0$ if we wrongly predict the outcome of a game and the corresponding $RPS_i = (p_i - X_i)^2 \in [0.25, 1]$ since $p_i - X_i \in [0.5, 1]$. Then we can say an RPS greater than 0.25 stands for a bad prediction.

Variance of RPS

When comparing the mean RPSs, the variance of RPS is very useful to construct confidence intervals and needs to be computed.

For each game i, we estimated the winning probabilities q_i of home teams, and the RPS can be directly calculated by applying the formula (5.1). However, since we have only one estimated probability for each game, we have to derive the formula for calculating the variance of the RPS.

Recall that $E(RPS_i) = p_i(1 - p_i) + (q_i - p_i)^2$. Since we used q_i as the estimation of p_i, we can assume $p_i = q_i$. Then, $E(RPS_i) = p_i(1 - p_i) = q_i(1 - q_i)$. Then, for each game i,

$$\begin{aligned}
Var(RPS_i) &= E(RPS_i^2) - E(RPS_i)^2 \\
&= p_i \cdot (1 - p_i)^4 + (1 - p_i) \cdot p_i^4 - (p_i(1 - p_i))^2 \\
&= q_i \cdot (1 - q_i)^4 + (1 - q_i) \cdot q_i^4 - (q_i(1 - q_i))^2
\end{aligned} \tag{5.5}$$

Furthermore, if there are n_{test} games in the test dataset, the variance of the mean RPS $\frac{1}{n_{test}} \sum_{i=1}^{n_{test}} RPS_i$ will be

$$Var\left(\frac{1}{n_{test}} \sum_{i=1}^{n_{test}} RPS_i\right) = \frac{1}{n_{test}^2} \sum_{i=1}^{n_{test}} Var(RPS_i) \tag{5.6}$$

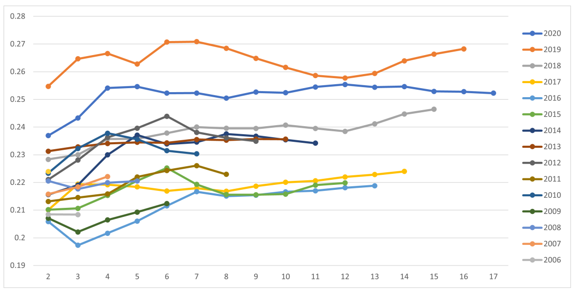
$$\begin{aligned}
\text{Consequently, } SD\left(\frac{1}{n_{test}} \sum_{i=1}^{n_{test}} RPS_i\right) &= \sqrt{\frac{1}{n_{test}^2} \sum_{i=1}^{n_{test}} Var(RPS_i)} \\
&= \frac{1}{n_{test}} \sqrt{\sum_{i=1}^{n_{test}} Var(RPS_i)}
\end{aligned} \tag{5.7}$$

5.2 Test data

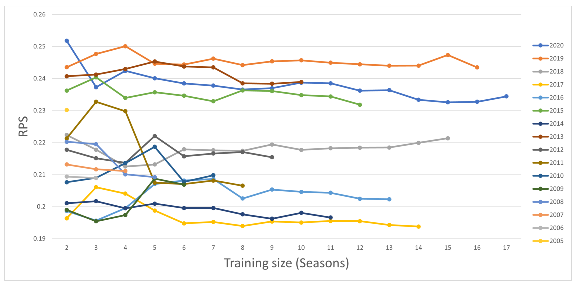
We used the playoffs from season 2005 to 2020 as the test dataset, which is 16 seasons in total. For each model, we computed the overall mean RPS of predictions for all seasons' playoffs together, which gave us a mean RPS and its standard deviation; and for every single season, we also computed the RPS and the standard deviation separately, i.e. 16 RPSs and their standard deviations, which can demonstrate the performance of models for different seasons. In the following sections, the boxplots will illustrate the 16 mean RPSs for each model. Let r_i^P and s_i^P denote the mean RPS and the corresponding standard deviation for model i (Poisson), and r_i^L and s_i^L for model i (Logistic), where $1 \leq i \leq 7$.

Figure 2: M3 and M7 with different training sizes

(a) M3 (Poisson)



(b) M7 (Logistic)



5.3 Determine the training size

Training data size is another hyper-parameter that needs to be determined before the model fitting. As mentioned above, we have data from season 2004 to season 2020, that is 17 seasons in total. For the playoff of each season, the range of training data size can vary from 1 season to 17 seasons. Then it is natural to ask what are the optimal training data sizes. We considered that different models may have different demands for training data as more complex models typically require more training data and simple models may not be able to model the variation of a large dataset. Therefore, we ran sets of simulations to verify this idea.

Specifically, each model experimented with 1 to 17 seasons of training data, and the average RPS of all seasons' playoffs and the standard deviations were recorded to determine the training size. Moreover, we also plotted the mean RPS of each model with different training sizes, which revealed that for the models without random effects, less training data was generally better. In contrast, more training data was beneficial for complex models with random effects. For example, Figure 2 illustrates that as training data increases, the performance of M7 (Logistic) generally increases and stabilizes, while the RPS of M3 (Poisson) has an increasing trend as data size grows. Therefore, in further analysis, we used 2 seasons of training data for models without random effects (M1, M2, M3, M4) and as much data as possible for models with random effects (M5, M6, M7). An additional reason why we used as much data as possible for models with random effects instead of the optimal training data sizes was that there were only a few seasons have large training data sizes and the optimal training sizes did not take this into account. For example, for the performance of the training size of 16 seasons, only season 2019 and season 2020 playoffs were predicted and the result would not be representative. Therefore, using as much data as possible was a reasonable choice since the performance stabilized.

Note that for the test set, we only considered seasons with as least 2 years of training data, i.e. season 2004 playoff was ignored. The optimal training size and actual used training size for each model can be found in Table 4.

Table 4: Optimal and actual training data size

Model	Optimal training size for Poisson regression (seasons)	Optimal training size for Logistic regression	actual used training size
1	2	2	2
2	2	2	2
3	2	2	2
4	2	2	2
5	3	5	2(for 2005) to 17 (for 2020)
6	3	5	2(for 2005) to 17 (for 2020)
7	3	6	2(for 2005) to 17 (for 2020)

5.4 Model comparison and feature analysis

In this section, models will be compared pairwisely and the effect of each feature will be analyzed. For each model, we checked the 95% credible intervals of all the regression coefficients and eliminated the insignificant features using backward selection. Then, we computed the RPSs based on the reduced models and used Welch's t-test to examine which model performs better.

The full results can be found in Table 5. Each cell contains the corresponding mean RPS and the standard deviations are in the parenthesis.

Table 5: RPS based on actual training sizes

Model	RPS for Poisson regression	RPS for Logistic regression
1	0.2343(0.0022)	0.2334(0.0024)
2	0.2209(0.0033)	0.2189(0.0040)
3	0.2196(0.0033)	0.2167(0.0040)
4	0.2210(0.0033)	0.2167(0.0039)
5	0.2185(0.0034)	0.2146(0.0041)
6	0.2188(0.0034)	0.2145(0.0041)
7	0.2183(0.0035)	0.2161(0.0042)

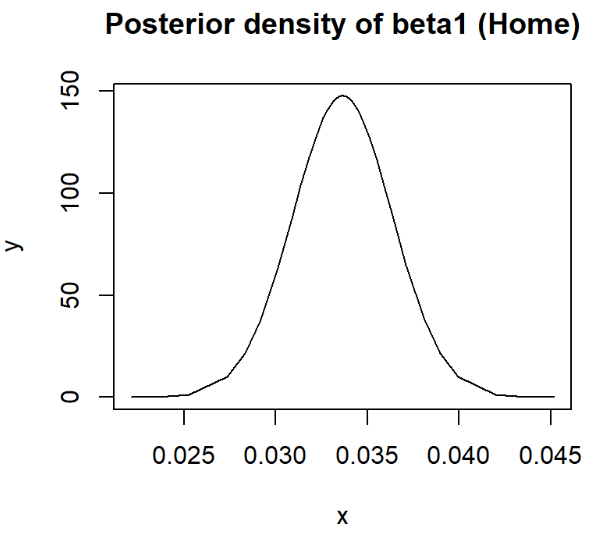
M1 vs M2

M1 only considers if the team plays at home. The model can be written as: $\eta_i = \beta_0 + \beta_1 h_i$, where $h_i = 1$ if the team is the home team, and $h_i = 0$ if the team is the away team. Equivalently, the model can be specified as:

$$\eta_i^H = \beta_0 + \beta_1$$
$$\eta_i^A = \beta_0$$

As can be seen from Section 4.1 and 4.2, the winning probabilities only depend on the linear predictors. Moreover, we can notice that η_i^H and η_i^A do not depend on i in this model, which means the model gives the same probability to all home teams, as well as all the away teams. Therefore, this model is the simplest one and was used as a baseline. In addition, Figure 3 shows the posterior density of *Home* after fitting the model with data from season 2007 to season 2008 excluding the 2008 playoff, and we can observe that *Home* has a significantly positive effect on the winning probability of the home team.

Figure 3: Posterior density of Home from M1 (Poisson), and the model was fit with data from season 2007 to season 2008 (not including the 2008 playoff)

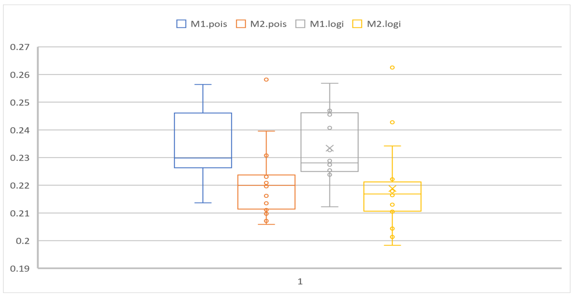


M2 added two features to give each team fixed and unique attacking and defending strengths. That is

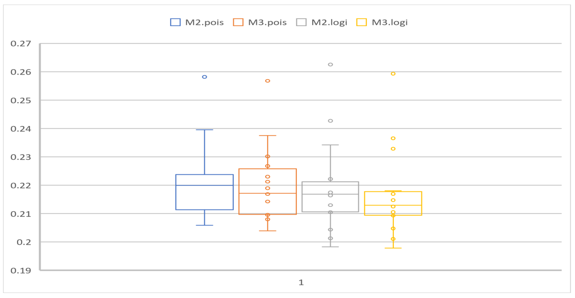
$$\eta_i^H = \beta_0 + \beta_1 + a_{home.team} + d_{away.team}$$
$$\eta_i^A = \beta_0 + a_{away.team} + d_{home.team}$$

Figure 4: Boxplots of mean RPS of all seasons' playoffs for different models

(a) M1 vs M2



(b) M2 vs M3



The attacking and defending strengths are fixed for each team, and they do not depend on the opponent team. For example, Miami Heat will have the same attacking and defending strengths when facing Los Angles Lakers or Golden State Warriors.

There are 60 regression coefficients (2 strengths for each team), and for clarity, the summary of the posterior marginal distributions of each team's attacking and defending strengths of M2 fit with the whole dataset can be found in Appendix A.

As can be seen in Figure 4(a), M2 has a much lower RPS than M1 for both Poisson and logistic regressions, and the interquartile range (IQR) of M2 is smaller than M1, indicating the RPSs are less dispersed. However, there are some outliers for M2. Moreover, the Welch's t-test [9] shows that the 95% confidence interval of $r_1^P - r_2^P$ is (0.0132, 0.0136) and that of $r_1^L - r_2^L$ is (0.0142, 0.0147). According to the analysis in (5.4), this corresponds to approximately an 11.5% difference in accuracy. Thus we can conclude that M2 outperforms M1 significantly.

M2 vs M3

M3 included the box score features and two extended features that can quantify the comprehensive abilities of teams in terms of basketball-specific aspects. The model can be specified as

$$\eta_i^H = \beta_0 + \beta_1 + a_{home.team} + d_{away.team} + \sum_{i=4}^{11} \beta_i x_i$$

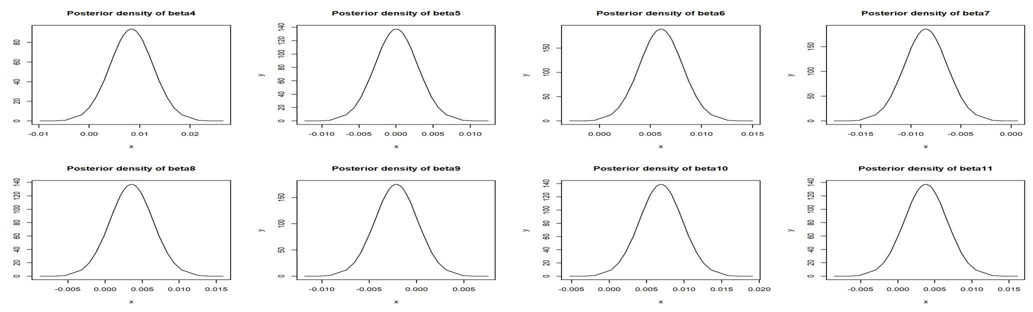
$$\eta_i^A = \beta_0 + a_{away.team} + d_{home.team} + \sum_{i=4}^{11} \beta_i x_i$$

where x_i's are the i-th features in Table 2.

Figure 5 shows the posterior density of each of the box score statistics. For Poisson regression, the 95% credible intervals of β_5, β_8, β_9, β_10, and β_{11} contain 0, which means the effects of feature 5, feature 8, feature 9, feature 10, and feature 11 are not significant for this model. For logistic regression, feature 5, feature 6, feature 7, feature 8, feature 9, and feature 10 do not have significant impacts since the 95% credible intervals of the corresponding regression coefficients cover 0. All of these insignificant features may be removed to improve the model fit. However, by Welch's t-test, we found that it did not improve the RPS.

Figure 5: Posterior densities of box score statistics of M3 fit with data from 2019 to 2020(not including the 2020 playoff)

(a) Poisson



(b) Logistic

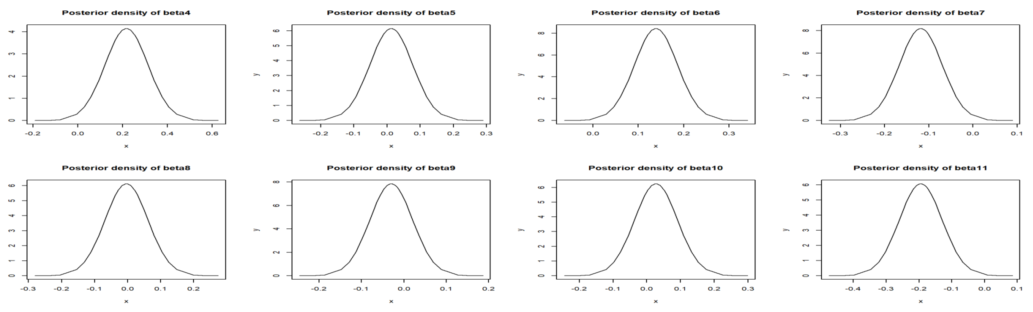


Figure 4(b) shows that M3 has lower median RPSs than M2 in both Poisson and logistic regression cases. However, the differences are not obvious by observation and M3 has more dispersed RPSs for Poisson regression since it has a greater IQR. Both M2 and M3 have outliers for Poisson and logistic regression. Furthermore, we performed the Welch's t-test for the differences $r_2^P - r_3^P$ and $r_2^L - r_3^L$, and the 95% confidence intervals are (0.0011, 0.0016) and (0.00199, 0.00258) respectively. Based on the confidence intervals, we can say that for the Poisson model, the mean RPS of M3 is lower than that of M2, but the improvement is not very significant. For the logistic model, the improvement is significant because (0.00199, 0.00258) corresponds to approximately (4.5%,5.1%) in accuracy according to the analysis in (5.4).

M3 vs M4

M4 took the salary feature into account. The model can be formulated as

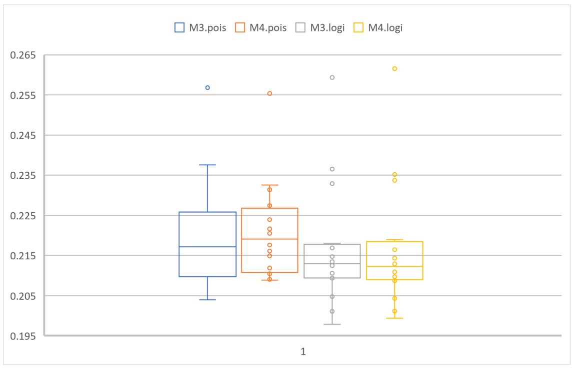
$$\eta_i^H = \beta_0 + \beta_1 + a_{home.team} + d_{away.team} + \sum_{i=4}^{12} \beta_i x_i$$

$$\eta_i^A = \beta_0 + a_{away.team} + d_{home.team} + \sum_{i=4}^{12} \beta_i x_i$$

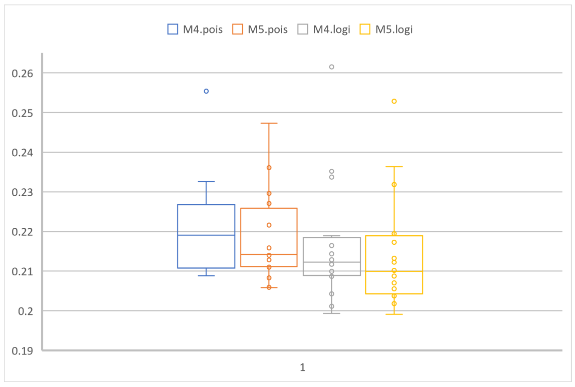
where x_i's are the i-th features in Table 2.

Figure 6: Boxplots of mean RPS of all seasons' playoffs for different models

(a) M3 vs M4



(b) M4 vs M5

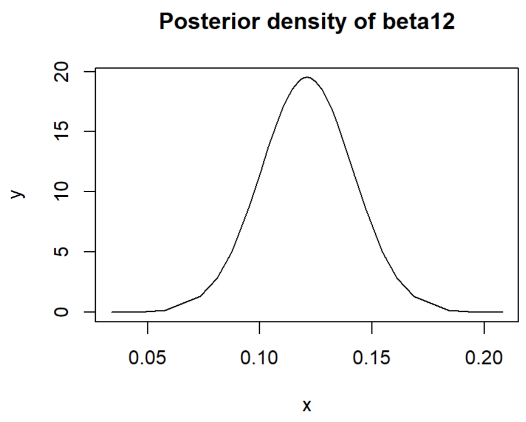


By checking the 95% credible intervals of the posterior densities of the regression coefficients, we found that the effects of feature 6, feature 7, and feature 9 are not significant for the logistic model.

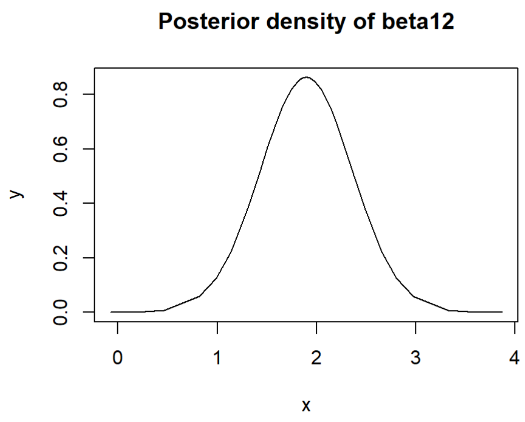
We refit the reduced model and computed the test criteria. It can be seen from Figure 6(a) that M4 does not outperform M3 although it added an additional feature. Based on Welch's t-test, the 95% confidence intervals for $r_3^P - r_4^P$ and $r_3^L - r_4^L$ are (-0.00164, -0.00114) and (-0.00032, 0.000268) respectively, which corresponds to (-4%,-3.4%) and (-1.8%,1.6%) in accuracy. Therefore, we concluded that M3 is better than M4 for Poisson regression and we cannot reject the hypothesis that $r_3^L = r_4^L$. Figure 7 shows the posterior marginal distribution of log(salary), and we can observe that salary has a clear positive effect on the scores but the overall effect may not be significant enough to influence the winning probabilities.

Figure 7: Posterior density of log(salary) from M4 fit with seasons 2019 and 2020 data (not including the 2020 playoff)

(a) Poisson



(b) Logistic



This can be explained by the fact that the NBA has a salary cap[5] which limits the amount of money that each team can pay to their players to prevent big clubs from buying all the top players, and many players are willing to take pay cuts to team up with other top players[6]. Consequently, salary may not be an indicative feature of the strengths of teams.

M4 vs M5

The attacking and defending strengths may vary from season to season and M5 used random effects to model this property.

As mentioned in Section 3.3, the indices $\{\text{a\_season}_{(i-1)\cdot S+1}, \text{a\_season}_{(i-1)\cdot S+2}, ..., \text{a\_season}_{i\cdot S}\}$ were used for the attacking strength for team i, and we assumed that $\forall k \in \{1, 2, ..., S\}$, $f(\text{a\_season}_{(i-1)\cdot S+k})-$

$f(\text{a\_season}_{(i-1)\cdot S+k-1}) \sim N(0, \sigma_f^2)$ for all team i where f is a random function, which is a first-order random walk process. The reason for picking the first-order random walk to model the variation is that we thought the attacking strength of the team does not change much between adjacent seasons and the process is smooth. The same assumption was made for the defending strength. Moreover, we used the `generic0` type model in `R-INLA` to define the first-order random walk for the seasonal attacking and defending strengths. In detail, we constructed the precision matrix $\boldsymbol{Q} = \tau \boldsymbol{C}$, where τ is a hyperparameter and $\boldsymbol{C}$ is the structure matrix. We used the default prior for τ initially and thus we only needed to construct the structure matrix $\boldsymbol{C}$. Without loss of generality, for team i, the structure matrix $\boldsymbol{C}_i$ can be expressed as:

$$\boldsymbol{C}_i = \begin{array}{c} \\ t_is_1 \\ t_is_2 \\ t_is_3 \\ ... \\ t_is_{16} \\ t_is_{17} \end{array} \begin{array}{c} \begin{array}{cccccc} t_is_1 & t_is_2 & t_is_3 & ... & t_is_{16} & t_is_{17} \end{array} \\ \begin{bmatrix} 1 & -1 & 0 & ... & 0 & 0 \\ -1 & 2 & -1 & ... & 0 & 0 \\ 0 & -1 & 2 & ... & 0 & 0 \\ ... & ... & ... & ... & ... & ... \\ 0 & 0 & ... & 0 & 2 & -1 \\ 0 & 0 & ... & 0 & -1 & 1 \end{bmatrix} \end{array}$$

Since the seasonal attacking and defending strengths of team i do not affect that of team j for all $i \neq j$, then the whole structure matrix is a block diagonal matrix with $\boldsymbol{C}_i$ as the element for the entry (t_i, t_i), representing team i. That is,

$$\boldsymbol{C} = \begin{array}{c} \\ t_1 \\ t_2 \\ ... \\ t_{30} \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & ... & t_{30} \end{array} \\ \begin{bmatrix} C_1 & 0 & ... & 0 \\ 0 & C_2 & ... & 0 \\ ... & ... & ... & ... \\ 0 & 0 & ... & C_{30} \end{bmatrix} \end{array}$$

We passed $\boldsymbol{C}$ to `inla()` by setting the parameter `Cmatrix` $= \boldsymbol{C}$. Note that $\boldsymbol{C}$ is a 510×510 matrix.

After the model fitting, we checked the 95% credible intervals of the regression coefficients of the Poisson class models based on their posterior marginal distributions and found that feature 1, feature 2, feature 6, feature 7, feature 10, and feature 11 do not have significant effects. As for the logistic class models, feature 1, feature 2, feature 5, feature 6, feature 7, feature 8, and feature 9 do not have significant effects, and then the reduced models were refitted.

Then, we plotted the posterior mean of attacking and defending strengths random effects for 76ers and Warriors over the 17 seasons. From Figure 8, we can observe that the attacking strength of the 76ers has always been insufficient. In contrast, the Warriors has always been good at attacking over all these seasons. Furthermore, we observed that Poisson models have less variability in the seasonal attacking and defending strengths compared to the logistic models. Therefore, for Poisson models, we tuned the priors for the random effect precision to check if the model can be improved, and we found that with low prior precision, the variability between seasons increased.

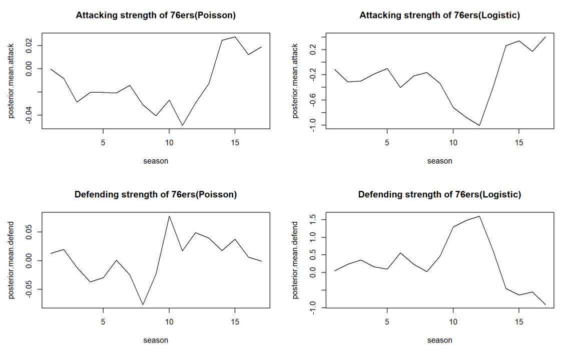
Figure 6(b) shows that M5 has lower median RPSs than M4 for most of the seasons, and M5 has fewer outliers than M4. In addition, for the worst-case scenario, M5 has about 0.01 improvement in RPS, which corresponds to about a 10% difference in accuracy. Furthermore, the 95% confidence intervals for $r_4^P - r_5^P$ and $r_4^L - r_5^L$ are (0.002217, 0.002721) and (0.001807,0.002406) respectively, which is equivalent to approximately (4.7%,45.2%) and (4.3%,4.9%) in accuracy according to the analysis in (5.4). Therefore, we concluded that M5 has significant improvements compared to M4.

M5 vs M6

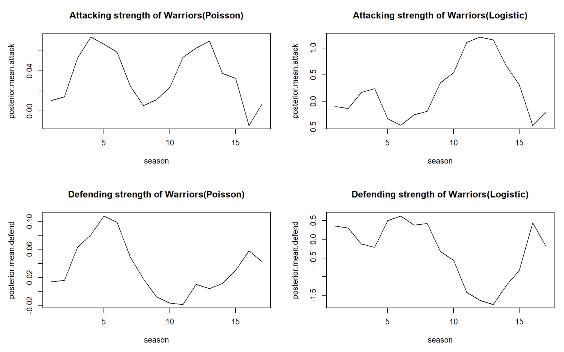
M6 added the random effect on travel distances of away teams. There are 30 teams and 29 arenas in NBA, which implies that there are $\binom{29}{2} = 406$ different distances. Then, all travel distances were categorized into 25 groups, $g_1, g_2, ...g_{25}$, based on the average length of the distances, and for any g_i and g_j, if $i > j$, then the distance represented by group i is longer than the distance represented by group j. We assumed a first-order random walk process for the effects of the groups, that is $f(g_i) - f(g_{i-1}) \sim N(0, \tau^{-1})$ for $i = 2, ..., 26$ where f is a random function.

Figure 8: Seasonal attacking and defending strengths for 76ers and warriors

(a) 76ers

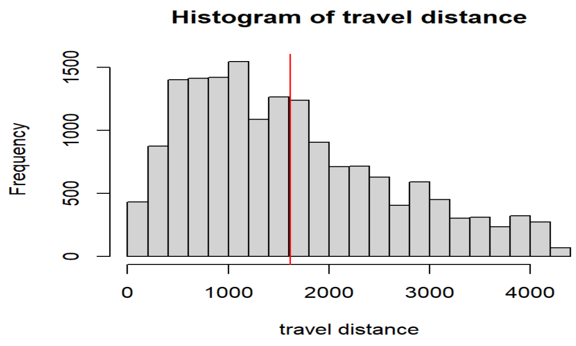


(b) Warriors



Moreover, we plotted the histogram 9 to summarize the distribution of the feature *Travel Distance*, and the value zeroes were removed beforehand because we wanted to focus on the travel distance of the away teams. It can be seen from the histogram that most of the distances are in the range of 500 km to 1500 km, and the mean of the distances was drawn as a red vertical line in the histogram.

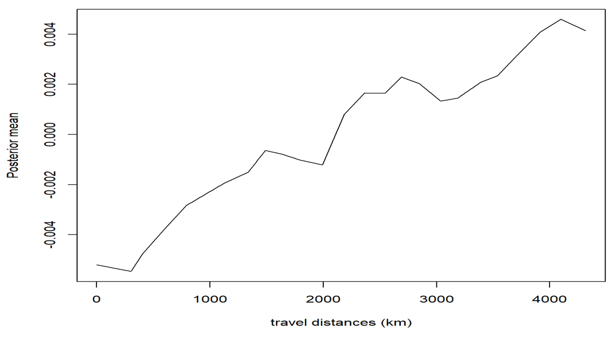
Figure 9: Histogram of the travel distances



For Poisson class models, feature 1, feature 2, feature 4, feature 5, and feature 7 do not have significant impacts while the effects of feature 1, feature 2, feature 5, feature 6, feature 7, feature 8, and feature 9 are not significant for logistics class models.

To investigate the effect of travel distance on game outcomes, we extracted the marginal posterior distribution of this random effect based on the `R-INLA` result of (Poisson) M7. In Figure 10, the horizontal axis represents the median of covariates(distances) in each group and the vertical axis represents the posterior means of the effects of each group. It can be seen that as the travel distance of the away teams increases, the posterior mean also generally increases, which means longer travel distances of the away teams will increase the scores and winning probabilities of the home team. This is reasonable and consistent with the analysis that longer trips have negative effects on the away teams.

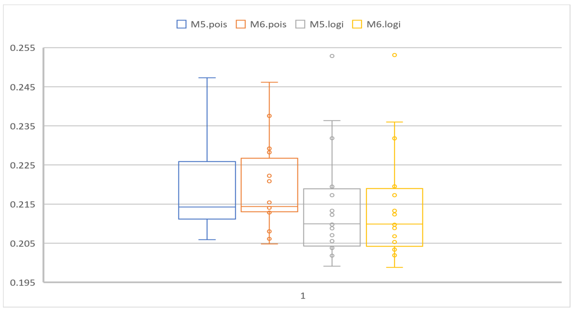
Figure 10: The effect of travel distances



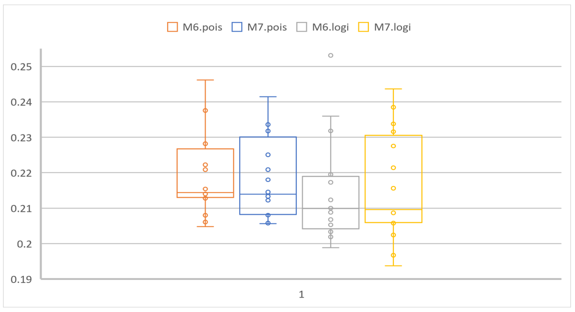
However, we still need to measure the improvements by looking at the boxplots and the confidence intervals. As can be seen in Figure 11(a), M6 has almost the same distribution of RPSs as M5. The 95% confidence intervals for $r_5^P - r_6^P$ and $r_5^L - r_6^L$ are (-0.00051, 0) and (-0.00014, 0.000294) respectively, which means we cannot reject the hypothesis that $r_5^P = r_6^P$ and $r_5^L - r_6^L = 0$. Therefore, M6 has no improvement compared to M5 in terms of the mean RPS.

Figure 11: Boxplots of mean RPS of all seasons' playoffs for different models

(a) M5 vs M6



(b) M6 vs M7



M6 vs M7

Several assumptions of the attacking and defending strengths have already been made in previous candidate models. In M2, the attacking and defending strengths were assumed to be fixed, and M5 added the seasonal variation into them. However, both of these assumptions did not take the opponents into account. In reality, a team has different attacking and defending strengths against different levels of teams. For example, a strong team may have significant attacking advantages over a weak team, while its attacking strength may decrease when facing an equally strong team. Therefore, we added interactions between the type of the game and attacking(defending) strength in M7 (Fearture 14 and 15 in Table 2).

In detail, we first categorized all the teams into l levels based on the rankings of last season, where l is a hyperparameter that needs to be determined, and we added the feature *Type* to each data instance in the manner described in Section 3.3. Then we fit the model with this new feature. To determine an appropriate number of levels, we compared the performance of 3 different values on l, which are 2, 3, and 5. For example, if $l = 3$, the top 10 teams will be categorized as 'Strong', the middle 10 teams will be categorized as 'Medium', and the bottom 10 teams will be classified as 'Weak'.

The boxplot 12 illustrates that the 5-level models have the highest RPSs among all three numbers of levels, and in the worst-case scenario, the RPSs are close to 0.26, which is a very poor performance as explained in Section 5.1. An interesting point we found was that the performance of 5-level models highly depends on the training data size. With a small training dataset, for example, two or three seasons, the model overfit and the performance was terrible, however, with larger datasets, the performance will be close to 2-level models. Using the 2017 playoff as the test dataset, we can observe such behavior in Figure 13. For Poisson regression, the RPS of the 5-level model is about 0.01 greater than the other two models when the training data size is smaller than 8 seasons, and the differences shrink as the training data size increases. For logistic regression, the RPS of the 5-level model is more than 0.27 with a training data size of 3 seasons, and the performance boosts to lower than 0.2 when the training data size grows to 8 seasons. In both cases, with 13 or 14 seasons of training data, the performance of the 5-level model is very close to the 2-level model, slightly underperforming the 3-level model.

Figure 12: The RPSs of different number of levels

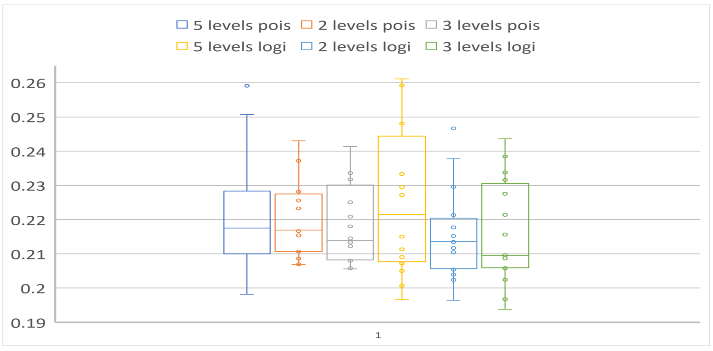
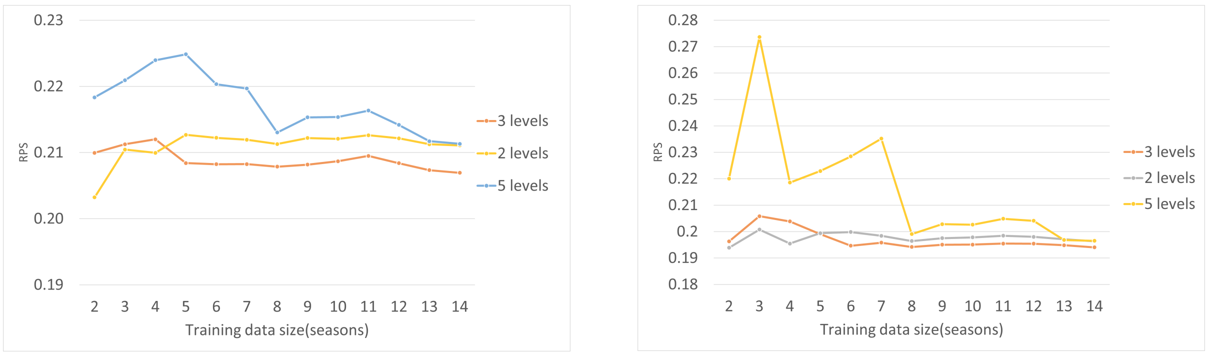


Figure 13: RPS for season 2017 with different levels of teams

(a) Poisson

(b) Logistic



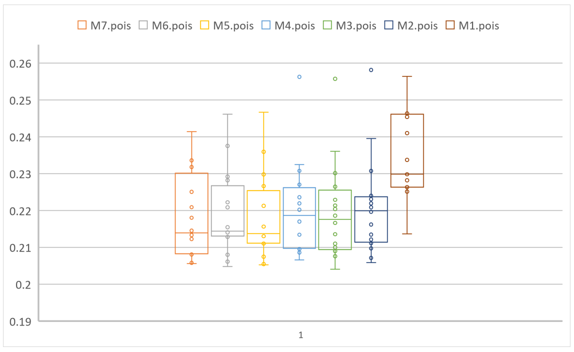
As for the other two models, the 3-level models have lower median RPSs than the 2-level models, as well as the minimum RPSs. However, the RPSs of 2-level models are more concentrated since the whiskers are shorter and the IQRs are also smaller. In addition, there are no outliers for 3-level models. By Welch's t-test, the 95% confidence intervals of the differences in mean RPSs between the 3-level models and the 2-level models are (-0.00134, -0.00097) (for Poisson models) and (0.000416, 0.00086) (for logistic models), respectively. Therefore, the 3-level models have better performances for the Poisson regression while 2-level models are better for the logistic regression. Finally, we thought that 3 was an appropriate choice for the number of levels since its advantage in the Poisson regression is greater than its disadvantage in the logistic regression, and 3-level was used for M7 in this report.

We also found that the 95% credible intervals of $\beta_1, \beta_2, \beta_4, \beta_5$, and β_7 cover 0 for Poisson class models and the 95% credible intervals of $\beta_1, \beta_2.\beta_5, \beta_6, \beta_7, \beta_8$ and β_9 cover 0 for logistic class models. Therefore, the corresponding features should be eliminated from the models.

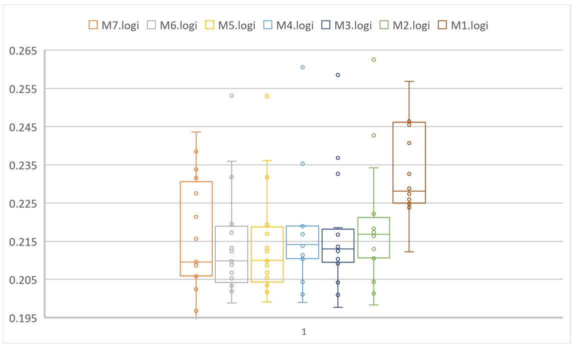
As for the comparison between M6 and M7, Figure 11 (b) depicts that M6 and M7 have similar medians for Poisson regression, and M7 has a smaller maximum and minimum while M6 has a more concentrated IQR. For the logistic regression, the medians are the same and M7 has a much smaller minimum RPS, however, the IQR of M6 is more concentrated. In addition, there is an outlier in M6 (Logistic), which is greater than 0.25. In contrast, there is no RPS greater than 0.25 in M7. By the Welch's test, the 95% confidence intervals for $r_6^P - r_7^P$ and $r_6^L - r_7^L$ are (0.00025, 0.00061) and (-0.00183, -0.0014) respectively, indicating that for Poisson regression, M7 has slightly better performance while M6 has a significant advantage over M7 for the logistic regression.

Figure 14: mean RPSs of all models

(a) Poisson



(b) Logistic



Summary

As mentioned before, the irrelevant features of each model were removed, and Table 6 summarizes what features were used for each reduced model.

Table 6: Features used for each reduced model

Model	Features	
	Poisson class	Logistic class
1	3	3
2	1,2,3	1,2,3
3	1,2,3,4,6,7	1,2,3,4,11
4	1,2,3,4,5,6,7,8,9,10,11,12	1,2,3,4,5,8,10,11,12
5	3,4,5,8,9,12,16,17	3,4,10,11,12,16,17
6	3,6,8,9,10,11,12,16,17,18	3,4,10,11,12,16,17,18
7	3,8,9,11,12,16,17,18,14,15	3,4,10,11,12,16,17,18,14,15

Figure 14 collects all the boxplots shown above to compare all 7 models. As can be seen from the diagram, M1 has the highest RPSs compared to the other 6 models. M2 has the second highest RPSs while it also has the smallest IQR for Poisson regression. Moreover, the interquartile ranges of M3 and M4 are much smaller than models with random effects in logistic regression while in the Poisson regression, M3-M6 have similar IQRs. Besides, M7 has the best performance among all the models for Poisson regression, but it has the largest IQR for both Poisson and logistic regression. M5 and M6 have similar distributions of RPSs in both cases, and they outperform M7 for logistic regression in terms of the median RPS.

Moreover, we computed the 95% confidence intervals for $r_i^P - r_i^L\ \forall i, 1 \leq i \leq 7$ and the results are shown in Table 7. It is obvious that logistic regression has a better performance than Poisson regression for all candidate models. In detail, the smallest improvement is for M1, which is approximately between 0.0008 and 0.0010, and the greatest difference is for M4, for which logistic regression decreases the RPS by about 0.0041 to 0.0047 compared to Poisson regression.

Table 7: Comparison between Poisson and Logistic models

Model	1	2	3	4
C.I. for $r_i^P - r_i^L$	(0.0008,0.0010)	(0.0018, 0.0022)	(0.0028,0.0031)	(0.0041, 0.0047)
Model	5	6	7	
C.I. for $r_i^P - r_i^L$	(0.0037, 0.0041)	(0.0040, 0.0044)	(0.0019, 0.0023)	

5.5 Bayesian model comparison

In this section, we compare the models based on four criteria, namely, conditional predictive ordinate (CPO), deviance information criterion (DIC)[25], predictive integral transform (PIT), and marginal log-likelihood.

- **CPO**

  CPO is a cross-validation type model measurement, which is defined as:

  $$CPO_i = p(y_i|y_{-i}) \text{ for each observation } i,$$

  where y_{-i} stands for the vector of all observations except y_i. Then, we summarized these CPO_i by negative log sum CPO:

  $$NLSCPO = -\sum_{i=1}^{n} \log(CPO_i).$$

  Smaller NLSCPO corresponds to better model fit.

- **DIC**

  DIC estimates how good the model fit is and penalizes the complexity of models. It is defined as:

  $$D(\hat{\boldsymbol{x}}, \hat{\boldsymbol{\theta}}) + 2p_D$$

  where $\hat{\boldsymbol{x}}, \hat{\boldsymbol{\theta}}$ are the posterior means of $\boldsymbol{x}$(latent effects) and $\boldsymbol{\theta}$(hyperparameters), D is the deviance function, and p_D is the effective number of parameters. Smaller DIC corresponds to a better model fit.

- **PIT**

  PIT computes the probability of a new observation from the posterior predictive distribution to be smaller than the observed value for each observation given all other observations, and it is defined as

  $$PIT_i = p(y_i^{new} \leq y_i|y_{-i})$$

  If the model is perfect, then PIT_i follow a Uniform(0,1) distribution.

- **Marginal log-likelihood**

  Marginal log-likelihood estimates the overall fit of the model and it is defined as

  $$\log(m(\mathbf{y})) = \log(\int_{\mathbf{x},\boldsymbol{\theta}} p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta})\pi(\mathbf{x},\boldsymbol{\theta}))$$

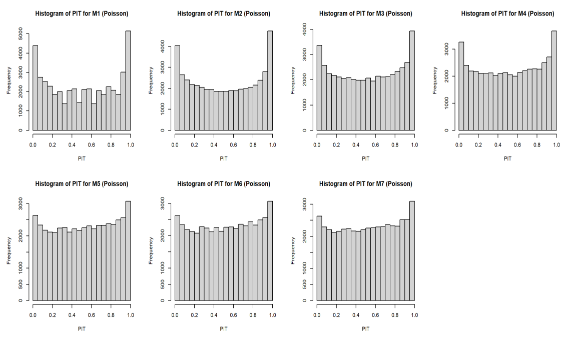
  Greater marginal log-likelihood values mean better model fit.

We fit the abovementioned models on the whole dataset (season 2004 to season 2020) and computed the criteria. Figure 15 illustrates the distribution of PIT_i for both Poisson and logistic class models, and it is obvious that Poisson class models perform well since the PIT_i are approximately uniformly distributed on [0,1]. Moreover, for Poisson class models, the PIT gets closer and closer to a Uniform(0,1) distribution as the model becomes more complex and models with random effects have better PIT distributions than the models without random effects. As for logistic models, there are only two possible observations (0 and 1), then, if $y_i = 1$, $PIT_i = 1$, which is the reason why there are high bars at $PIT_i = 1$. In such a circumstance, PIT is not suitable to evaluate the logistic models.

As for other criteria, the results can be found in Table 8. As can be seen, for each of these criteria, M6 has the best performance. Moreover, as the model gets more complex, the performance improves in terms of all of these criteria except that M7 does not outperform M5 and M6, but it does perform better than the models without random effects. In this regard, our strategy of progressively extending the model did yield better results. In addition, the most significant improvement is from M2 to M3, which means including the box score statistics has a notable positive impact on model fitting.

Figure 15: Histograms of PIT

(a) Poisson



(b) Logistic

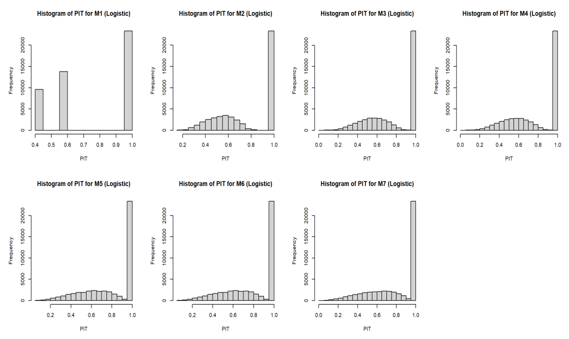


Table 8: Bayesian model comparisons

	Poisson class			Logisstic class		
Model	NLSCPO	DIC	Marginal log-Likelihood	NLSCPO	DIC	Marginal log-Likelihood
1	189905.5	379809.5	-189919.99	31598	63195.99	-31607.17
2	187810.1	375584.5	-188293.52	30872.98	61745.8	-31191.91
3	182901.8	365778.1	-183419.51	30055.69	60111.04	-30386.60
4	181224.4	362426.6	-181801.71	30018.74	60036.85	-30373.68
5	177894.8	355661.73	-178584.33	28284.17	56568.12	-28802.54
6	177873.7	355616	-178636.93	28284.08	56568	-28820.43
7	178006.3	355818.2	-181997.62	28422.69	56821.84	-30853.34

5.6 Model accuracy examination

A way of examining if the model is valid for prediction tasks is to compare the expected win percentage of the home team to the actual win percentage of the home team (See the website [2] for more details). In detail, we collected the estimated winning probabilities of the home team q_i for each game i in the test set, and split them into small bins: [0%,5%), [5%, 10%) ,..., [95%, 100%]. Let B_j denote the bins and also the set that contains all q_i such that $q_i \in B_j$. We picked 5% as the bin size because in such a case there would be enough data points in each bin and the number of bins would not be too small. Then, for each B_j, we computed the average of probabilities $\bar{q_j} = \frac{1}{|B_j|}\sum_{q_i \in B_j} q_i$. Moreover, we computed the actual win percentage of the home team $\tilde{q_j}$ of the games in the same bin as $\frac{1}{|B_j|}\sum_{i=1}^{|B_j|} \mathbf{I}(o_i = 1)$ where o_i's are the observations of outcomes for home teams of these games, and $\mathbf{I}$ is the indicator function defined as:

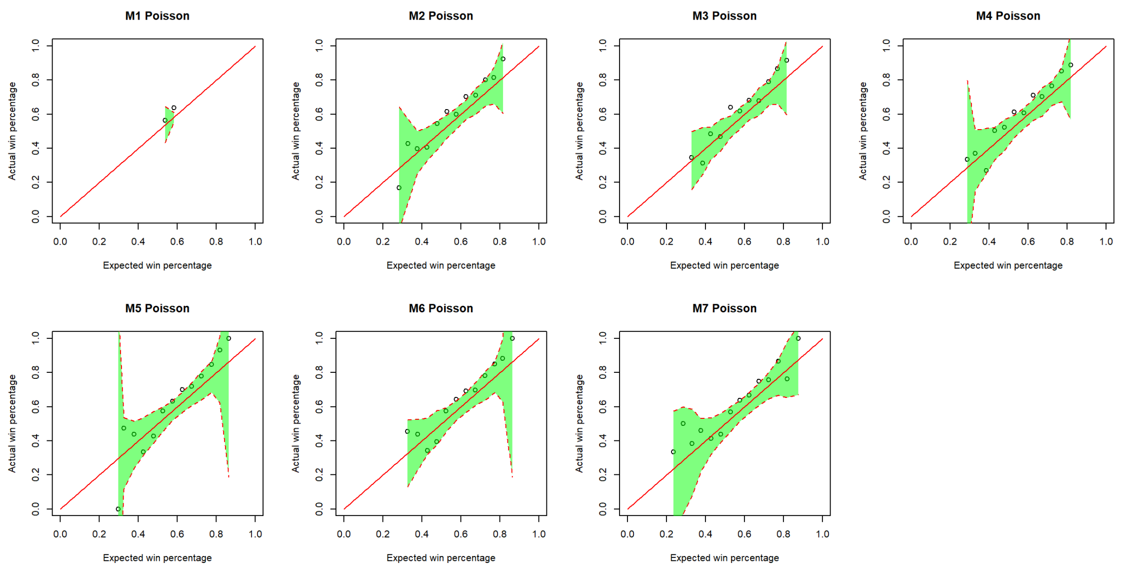
$$\mathbf{I}(o_i = 1) = \begin{cases} 1 \text{ , if } o_i = 1 \text{(Home team wins)} \\ 0 \text{ , otherwise} \end{cases}$$

We also calculated the 95% confidence interval of the expected win percentage for each bin B_j as $\bar{q_j} \mp 1.96 \cdot \frac{\sqrt{\sum_{q_i \in B_j} q_i \cdot (1-q_i)}}{|B_j|}$. Then, we expected that $\bar{q_j} \approx \tilde{q_j}$ and if we create a scatterplot of pairs $(\bar{q_j}, \tilde{q_j})$, the points should be on the straight line $y = x$, or at least within the 95% confidence intervals.

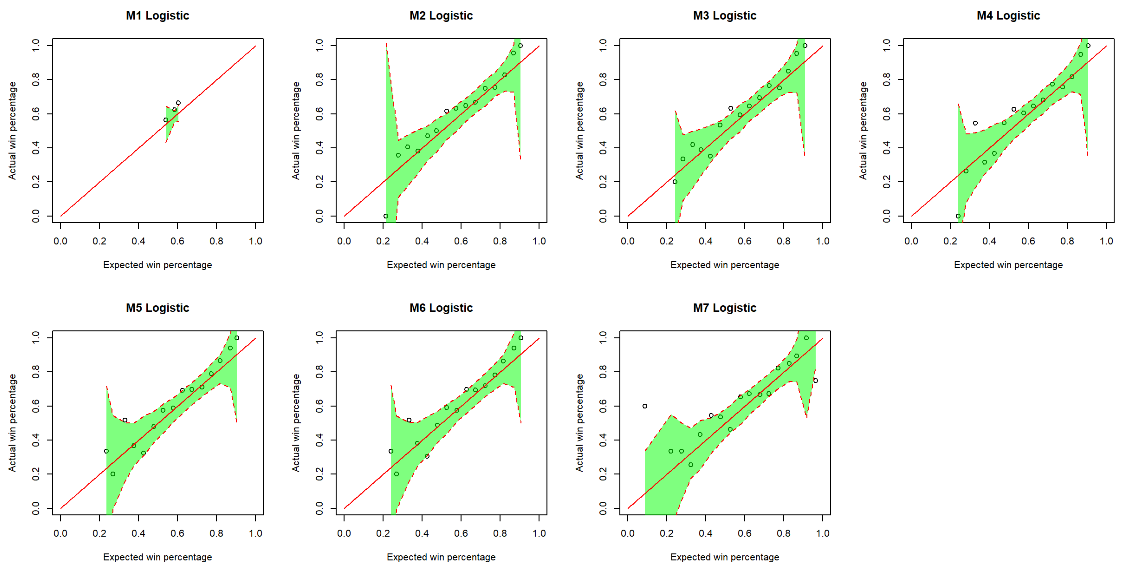
In the plot 16, the red lines represent $y = x$ and the green regions represent the 95% confidence intervals centered on the red line. Then, we can observe that almost all of the points are reasonably scattered near the red lines and within the green region, indicating that the models were valid. For Poisson models, most of the points are above the red lines, which means models underestimated the win percentage of the home team. As for the logistic class models, the points are more dispersed, however, the number of points is larger for each model compared to Poisson models, which means logistic class models are able to make more fine-grained predictions. Furthermore, almost all points have an expected win percentage greater than 20%. Moreover, since we expected there is a linear

Figure 16: Actual win percentage against Expected win percentage, the green regions represent 95% confidence intervals

(a) Poisson



(b) Logistic



relationship between $\tilde{q}_j$'s and $\bar{q}_j$'s, we can fit a linear model $\tilde{\boldsymbol{q}} \sim \bar{\boldsymbol{q}}$ to quantify the relationship. Accordingly, R-squared values and the root mean square error (RMSE) were computed. Note that we repeated the pairs $(\bar{q}_j, \tilde{q}_j)$ as many times as games are in the bin B_j in order not to give too much weight to bins with only a few samples.

Table 9 shows that all the models can capture such a linear relationship between $\tilde{q}_j$'s and $\bar{q}_j$'s since the RMSE and R-squared values are satisfactory. Furthermore, all R-squared values are greater than 0.8, which means at least 80% variance of the actual win percentage can be explained by the expected win percentage.

Since M1 only predicted the home team to win for each season's playoff, there are very few points in the plots and the probabilities are all within bins [50%,55%], [55%, 60%], and [60%, 65%]. It was shown in Table 1 that the win percentages of the home teams are around 0.6, therefore, the predictions made by M1 were very restricted and basic although it has the best RMSE and R-squared values. Therefore, we can conclude that a good model should be able to give fine-grained and wide-ranging probabilities instead of conservative predictions.

Table 9: Linear model checking

Model	RMSE	R-squared	RMSE	R-squared
	Poisson		Logistic	
M1	4.76418e-14	1	0.005992585	0.9453988
M2	0.03147902	0.9329897	0.03054723	0.9399507
M3	0.04119851	0.8918455	0.04318052	0.8984569
M4	0.03968956	0.897432	0.05457122	0.8417943
M5	0.04254582	0.9007239	0.04560058	0.9001276
M6	0.04638428	0.8804267	0.04931385	0.8835004
M7	0.03862449	0.9087755	0.06227491	0.8064707

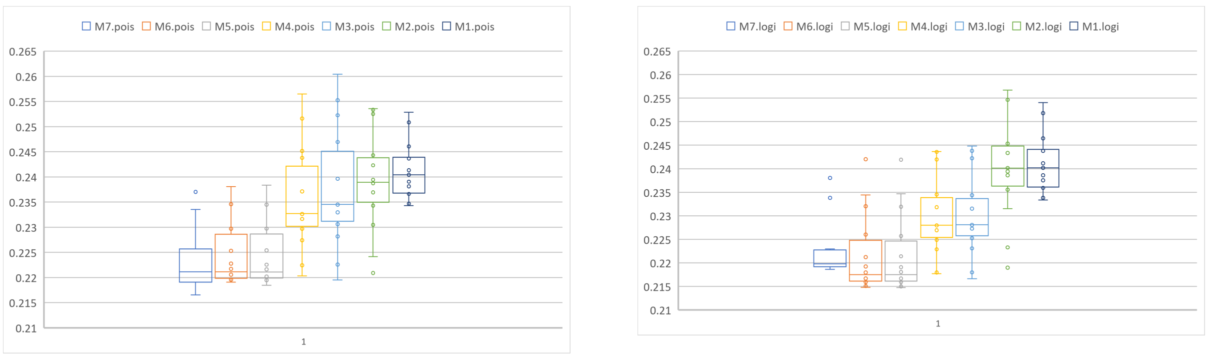
5.7 Performance with arbitrary training sizes

Although we determined the optimal training sizes in the analysis in Section 5.3 previously, obtaining such quantity is normally time-consuming. For example, there were 16 choices of training data size for the playoff in season 2020, and collecting the results for these 16 options required 16 model fits, which took much longer than fitting the model once. Therefore, we were also interested in the performances of models with arbitrary training sizes instead of the optimal ones. We computed the RPSs based on different training sizes for each model and generated boxplots. Boxplot 17(a) illustrates that for Poisson regression, the mean RPSs have a downward trend as the model gets more complicated. In detail, M1 has a median of about 0.24 and an IQR ranging from roughly 0.237 to 0.244. Compared to M1, M2 has a slightly lower median RPS and a higher IQR whose range is about (0.235,0.244). Furthermore, the median of M3 is about 0.235 while its IQR is the largest among all the models, ranging from roughly 0.231 to 0.245, and it also has the longest whiskers, indicating that the model is sensitive to the training size. The distribution of M4 is similar to M3 but with a smaller median RPS. As for the models with random effects (M5, M6, M7), they have the same median RPS of 0.221, which is significantly smaller than the models without random effects. Logistic class models have similar behaviours, however, M3 and M4 have much smaller IQRs compared to Poisson class models. In addition, M7 does not have the lowest RPS, but the IQR of M7 is notably smaller than all other models, indicating a stable performance.

In summary, models with random effects generally have shorter whiskers and smaller IQRs, reflecting that these models are more compatible with arbitrary training data sizes. The improvements are more significant than those in Figure 14, which means the insensitivity to training sizes is the advantage of more complex models.

Figure 17: Performance of all models with different training sizes

(a) Poisson (b) Logistic



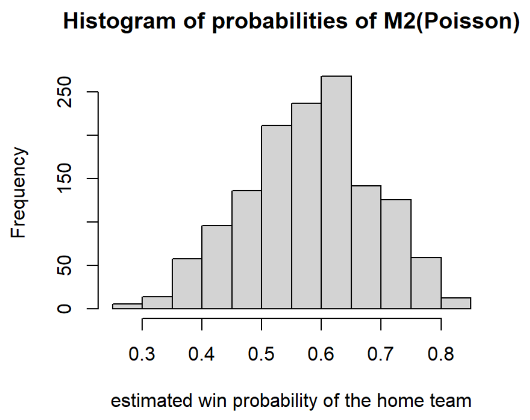
5.8 Space for improvement

Although the RPSs of different models were compared, we were still interested in how small an RPS is satisfactory. As described in (5.3), the minimum of $E(RPS_i) = p_i - p_i^2$ when $q_i = p_i$, where p_i is the true winning probability and q_i is the estimation of p_i. Then, even if we can accurately estimate the true probability p_i for each game, we cannot guarantee the expected RPSs to be very small because they depend on p_i. For example, if the game between Miami Heat and Phoenix Suns is a close match, i.e. $p_i = 0.5$, then $E(RPS_i)_{min} = 0.25$. Therefore, it is also important to investigate the minimum expected RPS when interpreting the RPS. Although we had no access to the true probabilities, we could still measure how good the estimated RPS by resampling with the estimated probabilities. In detail, for the test dataset, we used one of M1 to M7 to estimate q_i for each game i, and then we resampled n_{test} Bernoulli trials o_i with winning probability q_i. Then, we computed the corresponding mean RPS as $\frac{1}{n_{test}}(o_i - q_i)^2$. We repeated this procedure 1000 times and plotted the distribution of these samples in a histogram. Then, since the q_i's were treated as the true probabilities, these RPSs were theoretically the best ones. We can draw a line representing the corresponding estimated RPS (results from Table 5) of the same model to check if there is any space for improvement. From Figures 18, we can see that most of the estimated RPSs (red lines) are already in the left tails of the histograms, which means that there is not much room for improvement. In detail, the estimated mean RPS of each of the Poisson class models is already close to or smaller than the theoretically smallest RPS, and the specific number of samples that are smaller than the estimated RPSs are 0, 0, 0, 0, 0, 0, and 1 out of the 1000 samples. As for logistic class models, the estimated RPS of M1 is smaller than most of the theoretically best RPS calculated from the samples, and for M3, M4, M5, and M6, their estimated RPS is approximately among the smallest ten percent of the distribution. For M2 and M7, they are around the mean. The specific number of samples that are smaller than the estimated RPSs are 0, 361, 113, 92, 79, 92, and 558 out of 1000 samples respectively for each model.

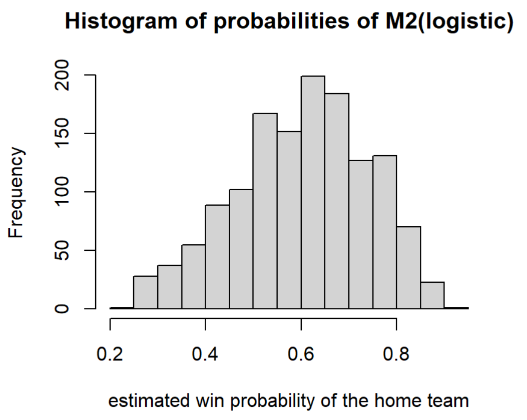
Furthermore, we plotted the histogram of estimated probabilities based on both Poisson and logistic regression. It can be seen from Figure 19 that the variance of probabilities from the logistic regression is larger than that of Poisson regression, and this can explain why logistic regression has smaller RPSs. This is also consistent with the analysis in Section 5.6 that logistic models can make more wide-ranging predictions.

Figure 19: Histograms of the estimated probabilities

(a) Poisson



(b) Logistic



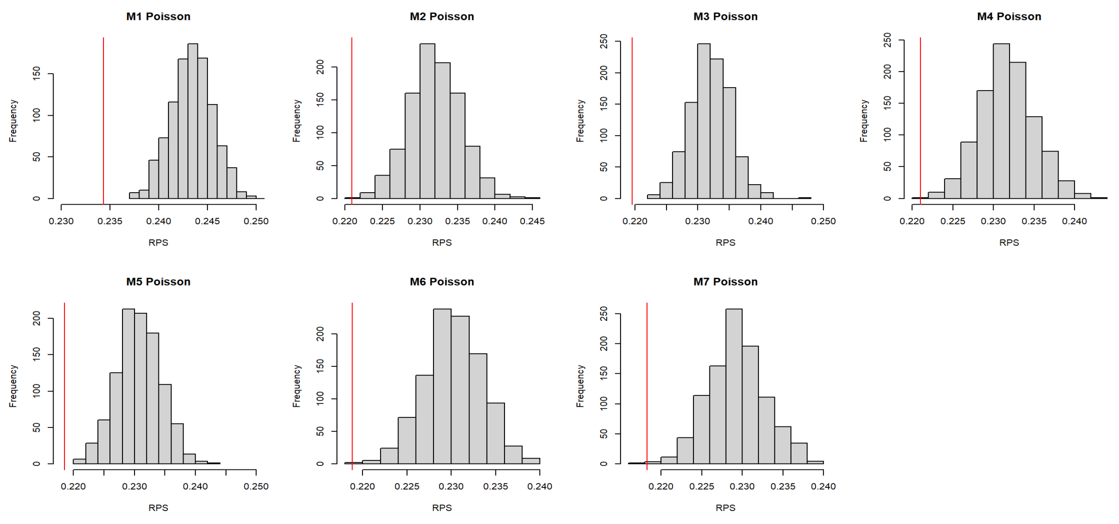
5.9 Time comparison

Another perspective for comparison of these models is the time complexity. Intuitively, one advantage of the simple models is that they normally take less time to run.

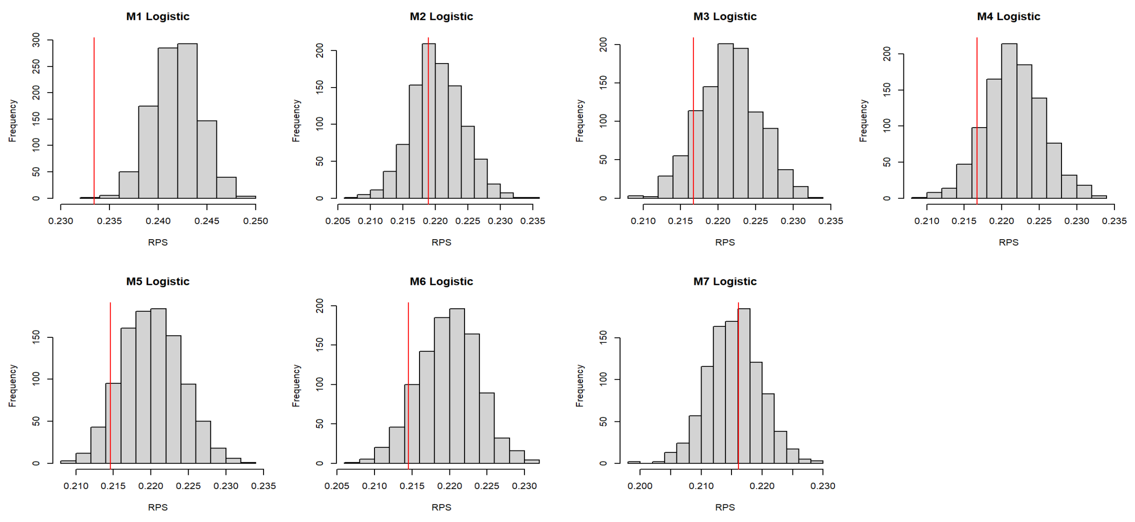
Using the 2020 playoff as the test set, we recorded the training sizes and the corresponding time cost of different models, and the result is displayed in Table 10. Since the available training size is

Figure 18: Posterior predictive check for the RPS

(a) Poisson



(b) Logistic



the largest for predicting the 2020 playoff, the results can be regarded as the upper bound of running time for prediction tasks for a single season's playoff. Rows 5 to 7 were recorded for the purpose of comparison while the last three rows were from the actual models we ran. From the table, we can observe that the running time increases as models become more complex, except that M6 took less time to run than M5. With a training size of 4810, the most complex model took 11.8 seconds and 14.3 seconds to run for Poisson regression and logistic regression respectively, which are approximately 9 and 16 times longer than the running times of the simplest model. Moreover, for models without random effects, logistic regression took less time to run but the differences are very small. In contrast, logistic regression took longer to run for models with random effects. With a training size of 46500, logistic class models with random effects took more than twice as long as Poisson class models. For example, M7 (logistic) took 149 seconds to run while M7 (Poisson) only took 58.5 seconds. Furthermore, the running time is proportional to the training data size for logistic class models since when the training size grows by $\frac{4980}{46670} \approx 9.4$ times, the running times of M5, M6, and M7 also grow by about 8.5, 11.36, and 10.4 times respectively. In contrast, the running time of Poisson class models grows much slower, and the running times increase by 6.2, 6.5, and 4.96 times for M5, M6, and M7 respectively.

In conclusion, simple models took much less time to run compared to complex models, and logistic regression took less time to run for simple models, while for complex models, it took more time than

Poisson class models. Although logistic class models with random effects have better performance, the drawback of longer running time exists.

Table 10: Time cost of different models for the 2020 playoff

Model	Training size (instances)	Time cost (seconds)	
		Poisson	Logistic
1	4810	1.26	0.869
2	4810	1.48	1.45
3	4810	1.48	1.45
4	4810	1.91	1.47
5	4810	4.88	8.87
6	4810	3.91	5
7	4810	11.8	14.3
5	46500	30.4	75.4
6	46500	25.6	56.8
7	46500	58.5	149

6 Conclusions

Poisson class models and logistic class models were fit by the INLA technique and compared for the prediction of NBA games in this thesis. We proposed 7 candidate models for both the Poisson regression and the logistic regression progressively, and there were four models without random effects and 3 models that have random effects. The main criteria we used in this study was the RPS, which can measure how close the predicted probabilities are to the observed outcomes, and we compared the candidate models with different training sizes in terms of the mean RPS on the test set of 1366 games to determine the best training sizes. It turned out that the optimal training sizes for models without random effects are always 2 seasons while for models with random effects, it varied from 3 to 6 seasons. In general, logistic class models performed better than Poisson class models significantly. M6 (Logistic) was the best performing model and it achieved a mean RPS of 0.2145. In addition, M7 was the most flexible model whose performance was very stable no matter how the training size changed, however, it also took the longest time to run, which was 58.5 seconds (for Poisson regression) and 149 seconds (for logistic regression), respectively. Besides, we also checked the models based on DIC, PIT, CPO, and marginal log-likelihood and M6 performed the best. We also did an analysis of the potential improvement of the RPS and found that our models had already generated accurate estimations. If combined RPS and time cost, M6 (Logistic) seems to be the best choice because it has the lowest RPS and took about half the time of M7 to run.

The main purpose of this dissertation is to propose appropriate models for predicting the outcomes and scores of NBA games and to measure the performance of using INLA to do the Bayesian inference. A notable limitation of this research is that for models with random effects, there were not enough data when predicting games that happened in the early years. For example, there were only 2 seasons of training data available for predicting the playoffs of the season 2005, and complex models normally require more data to train. In addition, the extracted features were very basic in this study and future research can explore more advanced features such as player-specific features. On top of adding new features, extending the current features is also worth pursuing. For example, *travel distance* in this study only treated the distances as scalars, however, the direction is also important because trips from west to east have different effects on players' performance than trips from east to west, and whether or not they cross time zones is also an important factor. Future research can make for a more comprehensive analysis of the effect of travel distance and other features. Another limitation of our study is that we did not do a detailed analysis of the impact of Covid-19 because there is not much post-pandemic data. Future research can explore the effect of Covid-19 in order to get better predictions for the seasons 2019 and 2020.

References

[1] Box score. https://en.wikipedia.org/wiki/Box_score. Accessed: 2022-08-06.

[2] Efficiency of pinnacle's opening nba moneyline compared to bet365. https://www.football-data.co.uk/blog/nba_pinnacle_efficiency.php. Accessed: 2022-08-06.

[3] Inflation & prices. https://www.bls.gov/data/#prices. Accessed: 2022-08-06.

[4] Nba salaries. https://hoopshype.com/salaries/. Accessed: 2022-08-06.

[5] Nba salary cap. https://en.wikipedia.org/wiki/NBA_salary_cap. Accessed: 2022-08-06.

[6] Report: James harden taking $15m pay cut in two-year deal with 76ers. https://tinyurl.com/4h6t4ex4. Accessed: 2022-08-06.

[7] Skellam distribution. https://en.wikipedia.org/wiki/Skellam_distribution. Accessed: 2022-08-06.

[8] Sports betting market size, share & trends analysis by platform, by type, by sports type (football, basketball, baseball, horse racing, cricket, hockey, others), by region, and segment forecasts, 2022 - 2030. https://www.grandviewresearch.com/industry-analysis/sports-betting-market-report. Accessed: 2022-08-06.

[9] Welch's t-test. https://en.wikipedia.org/wiki/Welch%27s_t-test. Accessed: 2022-08-06.

[10] D. Cervone, A. D'Amour, L. Bornn, and K. Goldsberry. A multiresolution stochastic process model for predicting basketball possession outcomes. *Journal of the American Statistical Association*, 111(514):585–599, 2016.

[11] G. Cheng, Z. Zhang, M. N. Kyebambe, and N. Kimbugwe. Predicting the outcome of nba playoffs based on the maximum entropy principle. *Entropy*, 18(12), 2016.

[12] E. S. Epstein. A Scoring System for Probability Forecasts of Ranked Categories. *Journal of Applied Meteorology*, 8(6):985–987, Dec. 1969.

[13] P. S. Gill. Late-game reversals in professional basketball, football, and hockey. *The American Statistician*, 54(2):94–99, 2022/08/14/ 2000. Full publication date: May, 2000.

[14] N. Gröwe-Kuska and W. Römisch. *Stochastic unit commitment in hydro-thermal power production planning.* Preprints aus dem Institut für Mathematik. Humboldt-Universität zu Berlin, Institut für Mathematik, 2001.

[15] T. Huyghe, A. T. Scanlan, V. J. Dalbo, and J. Calleja-González. The negative influence of air travel on health and performance in the national basketball association: A narrative review. *Sports (Basel)*, 6(3), Aug. 2018.

[16] D. Karlis and I. Ntzoufras. Analysis of sports data by using bivariate poisson models. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 52(3):381–393, Oct 2003.

[17] M. W. Y. Lam. One-match-ahead forecasting in two-team sports with stacked bayesian regressions. *Journal of Artificial Intelligence and Soft Computing Research*, 8(3):159–171, 2018.

[18] N. Lauga. Nba games data, 2022. Retrieved June 8, 2022 from https://www.kaggle.com/datasets/nathanlauga/nba-games/version/9.

[19] S. Merritt and A. Clauset. Scoring dynamics across professional team sports: tempo, balance and predictability. *EPJ Data Science*, 3(1):4, Feb 2014.

[20] D. Miljković, L. Gajić, A. Kovačević, and Z. Konjović. The use of data mining for basketball matches outcomes prediction. In *IEEE 8th International Symposium on Intelligent Systems and Informatics*, pages 309–312, 2010.

[21] P.-F. Pai, L.-H. ChangLiao, and K.-P. Lin. Analyzing basketball games by a support vector machines with decision tree model. *Neural Computing and Applications*, 28(12):4159–4167, Dec 2017.

[22] H. Rue, S. Martino, and N. Chopin. Approximate bayesian inference for latent gaussian models by using integrated nested laplace approximations. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(2):319–392, 2009.

[23] H. Rue, A. I. Riebler, S. H. Sørbye, J. B. Illian, D. P. Simpson, and F. K. Lindgren. Bayesian computing with INLA: A review. *Annual Reviews of Statistics and Its Applications*, 4(March):395–421, 2017.

[24] T. Shiina and J. R. Birge. Stochastic unit commitment problem. *International Transactions in Operational Research*, 11(1):19–32, 2004.

[25] D. J. Spiegelhalter, N. G. Best, B. P. Carlin, and A. Van Der Linde. Bayesian measures of model complexity and fit. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 64(4):583–639, 2002.

[26] A. Tsokos, S. Narayanan, I. Kosmidis, G. Baio, M. Cucuringu, G. Whitaker, and F. Király. Modeling outcomes of soccer matches. *Machine Learning*, 108(1):77–95, Jan 2019.

[27] T. Wolodzko. *extraDistr: Additional Univariate and Multivariate Distributions*, 2020. R package version 1.9.1.

[28] P. Wong. nba.kv.txt. `https://github.com/electricity345/Full.Text.Classification.Thesis/blob/master/config/kv.pairs/nba.kv.txt`, 2012.

Appendices

A R-INLA outputs of fitted models

- M1

```
Poisson:
formula = y~1+h
Call:
   c("inla.core(formula = formula, family = family, contrasts = contrasts, ", " data = data, quantiles
        =
   quantiles, E = E, offset = offset, ", " scale = scale, weights = weights, Ntrials = Ntrials, strata
        =
   strata, ", " lp.scale = lp.scale, link.covariates = link.covariates, verbose = verbose, ", " lincomb
         =
   lincomb, selection = selection, control.compute = control.compute, ", " control.predictor =
   control.predictor, control.family = control.family, ", " control.inla = control.inla, control.fixed
        =
   control.fixed, ", " control.mode = control.mode, control.expert = control.expert, ", " control.
        hazard
   = control.hazard, control.lincomb = control.lincomb, ", " control.update = control.update,
   control.lp.scale = control.lp.scale, ", " control.pardiso = control.pardiso, only.hyperparam =
   only.hyperparam, ", " inla.call = inla.call, inla.arg = inla.arg, num.threads = num.threads, ", "
   blas.num.threads = blas.num.threads, keep = keep, working.directory = working.directory, ", " silent
         =
   silent, inla.mode = inla.mode, safe = FALSE, debug = debug, ", " .parent.frame = .parent.frame)")
Time used:
    Pre = 0.661, Running = 3.87, Post = 0.262, Total = 4.79
Fixed effects:
             mean    sd 0.025quant 0.5quant 0.975quant mode kld
(Intercept) 4.609 0.001      4.608    4.609      4.611   NA   0
h           0.028 0.001      0.026    0.028      0.029   NA   0

Deviance Information Criterion (DIC) ...............: 379809.55
Deviance Information Criterion (DIC, saturated) ....: 810544.39
Effective number of parameters .....................: 2.00

Marginal log-Likelihood:  -189919.99
CPO, PIT is computed
Posterior summaries for the linear predictor and the fitted values are computed
(Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

```
Logistic:
formula = y.binary~1+h
Call:
   c("inla.core(formula = formula, family = family, contrasts = contrasts, ", " data = data, quantiles
        =
   quantiles, E = E, offset = offset, ", " scale = scale, weights = weights, Ntrials = Ntrials, strata
        =
   strata, ", " lp.scale = lp.scale, link.covariates = link.covariates, verbose = verbose, ", " lincomb
         =
   lincomb, selection = selection, control.compute = control.compute, ", " control.predictor =
   control.predictor, control.family = control.family, ", " control.inla = control.inla, control.fixed
        =
   control.fixed, ", " control.mode = control.mode, control.expert = control.expert, ", " control.
        hazard
   = control.hazard, control.lincomb = control.lincomb, ", " control.update = control.update,
   control.lp.scale = control.lp.scale, ", " control.pardiso = control.pardiso, only.hyperparam =
   only.hyperparam, ", " inla.call = inla.call, inla.arg = inla.arg, num.threads = num.threads, ", "
   blas.num.threads = blas.num.threads, keep = keep, working.directory = working.directory, ", " silent
         =
   silent, inla.mode = inla.mode, safe = FALSE, debug = debug, ", " .parent.frame = .parent.frame)")
Time used:
    Pre = 0.486, Running = 3.65, Post = 0.249, Total = 4.38
Fixed effects:
              mean    sd 0.025quant 0.5quant 0.975quant mode kld
(Intercept) -0.362 0.013     -0.388   -0.362     -0.336   NA   0
h            0.725 0.019      0.688    0.725      0.761   NA   0

Deviance Information Criterion (DIC) ...............: 63195.99
Deviance Information Criterion (DIC, saturated) ....: 63195.99
Effective number of parameters .....................: 2.00

Marginal log-Likelihood:  -31607.17
CPO, PIT is computed
Posterior summaries for the linear predictor and the fitted values are computed
(Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

- M2

```
Poisson:
formula = y~1+h+a+d
Call:
   c("inla.core(formula = formula, family = family, contrasts = contrasts, ", " data = data, quantiles
        =
   quantiles, E = E, offset = offset, ", " scale = scale, weights = weights, Ntrials = Ntrials, strata
        =
   strata, ", " lp.scale = lp.scale, link.covariates = link.covariates, verbose = verbose, ", " lincomb
         =
   lincomb, selection = selection, control.compute = control.compute, ", " control.predictor =
   control.predictor, control.family = control.family, ", " control.inla = control.inla, control.fixed
        =
   control.fixed, ", " control.mode = control.mode, control.expert = control.expert, ", " control.
        hazard
   = control.hazard, control.lincomb = control.lincomb, ", " control.update = control.update,
   control.lp.scale = control.lp.scale, ", " control.pardiso = control.pardiso, only.hyperparam =
    only.hyperparam, ", " inla.call = inla.call, inla.arg = inla.arg, num.threads = num.threads, ", "
    blas.num.threads = blas.num.threads, keep = keep, working.directory = working.directory, ", " silent
        =
    silent, inla.mode = inla.mode, safe = FALSE, debug = debug, ", " .parent.frame = .parent.frame)")
Time used:
    Pre = 1.45, Running = 6.8, Post = 0.531, Total = 8.78
Fixed effects:
                   mean    sd 0.025quant 0.5quant 0.975quant mode kld
(Intercept)       4.603 0.004      4.596    4.603      4.610   NA   0
aBucks            0.015 0.004      0.008    0.015      0.022   NA   0
aBulls           -0.011 0.004     -0.018   -0.011     -0.004   NA   0
aCavaliers        0.000 0.004     -0.007    0.000      0.007   NA   0
aCeltics          0.006 0.004     -0.001    0.006      0.013   NA   0
aClippers         0.020 0.004      0.013    0.020      0.027   NA   0
aGrizzlies       -0.014 0.004     -0.021   -0.014     -0.006   NA   0
aHawks            0.001 0.004     -0.006    0.001      0.008   NA   0
aHeat            -0.003 0.004     -0.010   -0.003      0.004   NA   0
aHornets         -0.021 0.004     -0.028   -0.021     -0.014   NA   0
aJazz             0.007 0.004      0.000    0.007      0.014   NA   0
aKings            0.022 0.004      0.015    0.022      0.029   NA   0
aKnicks           0.000 0.004     -0.007    0.000      0.007   NA   0
aLakers           0.025 0.004      0.018    0.025      0.032   NA   0
aMagic           -0.005 0.004     -0.012   -0.005      0.002   NA   0
aMavericks        0.019 0.004      0.012    0.019      0.026   NA   0
aNets            -0.005 0.004     -0.012   -0.005      0.002   NA   0
aNuggets          0.054 0.004      0.047    0.054      0.061   NA   0
aPacers           0.002 0.004     -0.005    0.002      0.009   NA   0
aPelicans         0.001 0.004     -0.006    0.001      0.008   NA   0
aPistons         -0.023 0.004     -0.030   -0.023     -0.016   NA   0
aRaptors          0.024 0.004      0.017    0.024      0.031   NA   0
aRockets          0.035 0.004      0.028    0.035      0.042   NA   0
aSpurs            0.014 0.004      0.007    0.014      0.021   NA   0
aSuns             0.057 0.004      0.050    0.057      0.064   NA   0
aThunder          0.033 0.004      0.026    0.033      0.039   NA   0
aTimberwolves     0.010 0.004      0.003    0.010      0.017   NA   0
aTrail Blazers    0.009 0.004      0.002    0.009      0.016   NA   0
aWarriors         0.069 0.004      0.062    0.069      0.076   NA   0
aWizards          0.018 0.004      0.011    0.018      0.025   NA   0
dBucks            0.003 0.004     -0.004    0.003      0.010   NA   0
dBulls           -0.026 0.004     -0.033   -0.026     -0.019   NA   0
dCavaliers       -0.013 0.004     -0.020   -0.013     -0.006   NA   0
dCeltics         -0.030 0.004     -0.037   -0.030     -0.023   NA   0
dClippers        -0.009 0.004     -0.016   -0.009     -0.002   NA   0
dGrizzlies       -0.028 0.004     -0.035   -0.028     -0.021   NA   0
dHawks           -0.001 0.004     -0.008   -0.001      0.006   NA   0
dHeat            -0.035 0.004     -0.042   -0.035     -0.028   NA   0
dHornets         -0.007 0.004     -0.014   -0.007      0.001   NA   0
dJazz            -0.030 0.004     -0.037   -0.030     -0.023   NA   0
dKings            0.030 0.004      0.023    0.030      0.037   NA   0
dKnicks           0.013 0.004      0.006    0.013      0.020   NA   0
dLakers           0.004 0.004     -0.003    0.004      0.010   NA   0
dMagic           -0.015 0.004     -0.022   -0.015     -0.008   NA   0
dMavericks       -0.020 0.004     -0.027   -0.020     -0.013   NA   0
dNets             0.004 0.004     -0.003    0.004      0.011   NA   0
dNuggets          0.016 0.004      0.009    0.016      0.023   NA   0
dPacers          -0.019 0.004     -0.026   -0.019     -0.012   NA   0
dPelicans        -0.008 0.004     -0.015   -0.008     -0.001   NA   0
dPistons         -0.035 0.004     -0.042   -0.035     -0.028   NA   0
dRaptors         -0.001 0.004     -0.008   -0.001      0.006   NA   0
dRockets         -0.011 0.004     -0.018   -0.011     -0.004   NA   0
dSpurs           -0.053 0.004     -0.060   -0.053     -0.046   NA   0
dSuns             0.034 0.004      0.027    0.034      0.041   NA   0
dThunder          0.002 0.004     -0.005    0.002      0.009   NA   0
dTimberwolves     0.018 0.004      0.011    0.018      0.025   NA   0
dTrail Blazers   -0.012 0.004     -0.019   -0.012     -0.005   NA   0
dWarriors         0.028 0.004      0.021    0.028      0.035   NA   0
dWizards          0.019 0.004      0.012    0.019      0.026   NA   0
h                 0.027 0.001      0.026    0.027      0.029   NA   0

Deviance Information Criterion (DIC) ...............: 375584.53
Deviance Information Criterion (DIC, saturated) ....: 806319.36
Effective number of parameters .....................: 59.99

Marginal log-Likelihood:  -188293.52
CPO, PIT is computed
Posterior summaries for the linear predictor and the fitted values are computed
(Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')


Logistic:
formula = y.binary~1+h+a+d
Call:
    c("inla.core(formula = formula, family = family, contrasts = contrasts, ", " data = data, quantiles
        =
    quantiles, E = E, offset = offset, ", " scale = scale, weights = weights, Ntrials = Ntrials, strata
        =
    strata, ", " lp.scale = lp.scale, link.covariates = link.covariates, verbose = verbose, ", " lincomb
        =
    lincomb, selection = selection, control.compute = control.compute, ", " control.predictor =
    control.predictor, control.family = control.family, ", " control.inla = control.inla, control.fixed
        =
    control.fixed, ", " control.mode = control.mode, control.expert = control.expert, ", " control.
        hazard
    = control.hazard, control.lincomb = control.lincomb, ", " control.update = control.update,
    control.lp.scale = control.lp.scale, ", " control.pardiso = control.pardiso, only.hyperparam =
    only.hyperparam, ", " inla.call = inla.call, inla.arg = inla.arg, num.threads = num.threads, ", "
    blas.num.threads = blas.num.threads, keep = keep, working.directory = working.directory, ", " silent
        =
    silent, inla.mode = inla.mode, safe = FALSE, debug = debug, ", " .parent.frame = .parent.frame)")
Time used:
    Pre = 1.17, Running = 6.42, Post = 0.559, Total = 8.15
Fixed effects:
                   mean    sd 0.025quant 0.5quant 0.975quant mode kld
```

```
(Intercept)     -0.374 0.076      -0.523    -0.374     -0.225   NA   0
aBucks           0.106 0.074      -0.039     0.106      0.252   NA   0
aBulls           0.260 0.074       0.115     0.260      0.405   NA   0
aCavaliers       0.267 0.074       0.122     0.267      0.411   NA   0
aCeltics         0.460 0.073       0.316     0.460      0.604   NA   0
aClippers        0.420 0.074       0.274     0.420      0.566   NA   0
aGrizzlies       0.253 0.075       0.106     0.253      0.399   NA   0
aHawks           0.107 0.074      -0.038     0.107      0.252   NA   0
aHeat            0.487 0.073       0.343     0.487      0.631   NA   0
aHornets        -0.196 0.076      -0.345    -0.196     -0.048   NA   0
aJazz            0.469 0.075       0.323     0.469      0.615   NA   0
aKings          -0.126 0.076      -0.276    -0.126      0.023   NA   0
aKnicks         -0.200 0.076      -0.349    -0.200     -0.052   NA   0
aLakers          0.370 0.074       0.225     0.370      0.515   NA   0
aMagic           0.096 0.074      -0.049     0.096      0.242   NA   0
aMavericks       0.586 0.074       0.440     0.586      0.732   NA   0
aNets           -0.069 0.075      -0.215    -0.069      0.078   NA   0
aNuggets         0.557 0.075       0.411     0.557      0.703   NA   0
aPacers          0.284 0.074       0.139     0.284      0.429   NA   0
aPelicans        0.130 0.075      -0.018     0.130      0.277   NA   0
aPistons         0.137 0.074      -0.009     0.137      0.282   NA   0
aRaptors         0.292 0.074       0.147     0.292      0.438   NA   0
aRockets         0.643 0.075       0.496     0.643      0.789   NA   0
aSpurs           0.939 0.075       0.792     0.939      1.086   NA   0
aSuns            0.358 0.075       0.211     0.358      0.504   NA   0
aThunder         0.469 0.074       0.323     0.469      0.615   NA   0
aTimberwolves   -0.235 0.077      -0.385    -0.235     -0.084   NA   0
aTrail Blazers   0.329 0.075       0.183     0.329      0.475   NA   0
aWarriors        0.561 0.074       0.415     0.561      0.706   NA   0
aWizards        -0.010 0.075      -0.157    -0.010      0.136   NA   0
dBucks          -0.106 0.074      -0.252    -0.106      0.039   NA   0
dBulls          -0.260 0.074      -0.405    -0.260     -0.115   NA   0
dCavaliers      -0.267 0.074      -0.411    -0.267     -0.122   NA   0
dCeltics        -0.460 0.073      -0.604    -0.460     -0.316   NA   0
dClippers       -0.420 0.074      -0.566    -0.420     -0.274   NA   0
dGrizzlies      -0.253 0.075      -0.399    -0.253     -0.106   NA   0
dHawks          -0.107 0.074      -0.252    -0.107      0.038   NA   0
dHeat           -0.487 0.073      -0.631    -0.487     -0.343   NA   0
dHornets         0.196 0.076       0.048     0.196      0.345   NA   0
dJazz           -0.469 0.075      -0.615    -0.469     -0.323   NA   0
dKings           0.126 0.076      -0.023     0.126      0.276   NA   0
dKnicks          0.200 0.076       0.052     0.200      0.349   NA   0
dLakers         -0.370 0.074      -0.515    -0.370     -0.225   NA   0
dMagic          -0.096 0.074      -0.242    -0.096      0.049   NA   0
dMavericks      -0.586 0.074      -0.732    -0.586     -0.440   NA   0
dNets            0.069 0.075      -0.078     0.069      0.215   NA   0
dNuggets        -0.557 0.075      -0.703    -0.557     -0.411   NA   0
dPacers         -0.284 0.074      -0.429    -0.284     -0.139   NA   0
dPelicans       -0.130 0.075      -0.277    -0.130      0.018   NA   0
dPistons        -0.137 0.074      -0.282    -0.137      0.009   NA   0
dRaptors        -0.292 0.074      -0.438    -0.292     -0.147   NA   0
dRockets        -0.643 0.075      -0.789    -0.643     -0.496   NA   0
dSpurs          -0.939 0.075      -1.086    -0.939     -0.792   NA   0
dSuns           -0.358 0.075      -0.504    -0.358     -0.211   NA   0
dThunder        -0.469 0.074      -0.615    -0.469     -0.323   NA   0
dTimberwolves    0.235 0.077       0.084     0.235      0.385   NA   0
dTrail Blazers  -0.329 0.075      -0.475    -0.329     -0.183   NA   0
dWarriors       -0.561 0.074      -0.706    -0.561     -0.415   NA   0
dWizards         0.010 0.075      -0.136     0.010      0.157   NA   0
h                0.748 0.019       0.710     0.748      0.785   NA   0

Deviance Information Criterion (DIC) ...............: 61745.79
Deviance Information Criterion (DIC, saturated) ....: 61745.79
Effective number of parameters .....................: 59.95

Marginal log-Likelihood:  -31191.91
CPO, PIT is computed
Posterior summaries for the linear predictor and the fitted values are computed
(Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

- M3

```
Poisson:
formula = y~1+h+a+d+scale(PTS_cur_season)+scale(FT_PCT_cur_season)+scale(FG3_PCT_cur_season)
Call:
   c("inla.core(formula = formula, family = family, contrasts = contrasts, ", " data = data, quantiles
       =
   quantiles, E = E, offset = offset, ", " scale = scale, weights = weights, Ntrials = Ntrials, strata
       =
   strata, ", " lp.scale = lp.scale, link.covariates = link.covariates, verbose = verbose, ", " lincomb
       =
   lincomb, selection = selection, control.compute = control.compute, ", " control.predictor =
   control.predictor, control.family = control.family, ", " control.inla = control.inla, control.fixed
       =
   control.fixed, ", " control.mode = control.mode, control.expert = control.expert, ", " control.
       hazard
   = control.hazard, control.lincomb = control.lincomb, ", " control.update = control.update,
   control.lp.scale = control.lp.scale, ", " control.pardiso = control.pardiso, only.hyperparam =
   only.hyperparam, ", " inla.call = inla.call, inla.arg = inla.arg, num.threads = num.threads, ", "
   blas.num.threads = blas.num.threads, keep = keep, working.directory = working.directory, ", " silent
       =
   silent, inla.mode = inla.mode, safe = FALSE, debug = debug, ", " .parent.frame = .parent.frame)")
Time used:
    Pre = 0.719, Running = 4.1, Post = 0.368, Total = 5.19
Fixed effects:
                          mean    sd 0.025quant 0.5quant 0.975quant mode kld
(Intercept)              4.599 0.004      4.592    4.599      4.606   NA   0
aBucks                   0.015 0.004      0.008    0.015      0.022   NA   0
aBulls                   0.009 0.004      0.001    0.009      0.016   NA   0
aCavaliers               0.008 0.004      0.001    0.008      0.015   NA   0
aCeltics                 0.018 0.004      0.012    0.018      0.025   NA   0
aClippers                0.012 0.004      0.005    0.012      0.019   NA   0
```

```
aGrizzlies                 -0.001 0.004      -0.008    -0.001       0.006    NA    0
aHawks                      0.011 0.004       0.004     0.011       0.018    NA    0
aHeat                      -0.002 0.004      -0.009    -0.002       0.005    NA    0
aHornets                   -0.004 0.004      -0.011    -0.004       0.004    NA    0
aJazz                       0.009 0.004       0.002     0.009       0.016    NA    0
aKings                      0.022 0.004       0.015     0.022       0.029    NA    0
aKnicks                     0.011 0.004       0.004     0.011       0.018    NA    0
aLakers                     0.003 0.004      -0.004     0.003       0.010    NA    0
aMagic                     -0.003 0.004      -0.010    -0.003       0.004    NA    0
aMavericks                  0.032 0.004       0.025     0.032       0.039    NA    0
aNets                       0.006 0.004      -0.001     0.006       0.013    NA    0
aNuggets                    0.021 0.004       0.014     0.021       0.028    NA    0
aPacers                     0.016 0.004       0.009     0.016       0.023    NA    0
aPelicans                   0.010 0.004       0.003     0.010       0.018    NA    0
aPistons                   -0.012 0.004      -0.019    -0.012      -0.005    NA    0
aRaptors                    0.031 0.004       0.024     0.031       0.038    NA    0
aRockets                    0.022 0.004       0.015     0.022       0.029    NA    0
aSpurs                      0.031 0.004       0.024     0.031       0.038    NA    0
aSuns                       0.040 0.004       0.033     0.040       0.047    NA    0
aThunder                    0.031 0.004       0.024     0.031       0.038    NA    0
aTimberwolves               0.016 0.004       0.009     0.016       0.024    NA    0
aTrail Blazers              0.026 0.004       0.019     0.026       0.034    NA    0
aWarriors                   0.038 0.004       0.031     0.038       0.045    NA    0
aWizards                    0.014 0.004       0.007     0.014       0.021    NA    0
dBucks                      0.003 0.004      -0.004     0.003       0.010    NA    0
dBulls                     -0.024 0.004      -0.031    -0.024      -0.017    NA    0
dCavaliers                 -0.011 0.004      -0.018    -0.011      -0.004    NA    0
dCeltics                   -0.030 0.004      -0.037    -0.030      -0.023    NA    0
dClippers                  -0.009 0.004      -0.016    -0.009      -0.002    NA    0
dGrizzlies                 -0.027 0.004      -0.034    -0.027      -0.020    NA    0
dHawks                      0.000 0.004      -0.007     0.000       0.007    NA    0
dHeat                      -0.034 0.004      -0.041    -0.034      -0.027    NA    0
dHornets                   -0.006 0.004      -0.013    -0.006       0.002    NA    0
dJazz                      -0.030 0.004      -0.038    -0.030      -0.023    NA    0
dKings                      0.032 0.004       0.025     0.032       0.039    NA    0
dKnicks                     0.014 0.004       0.007     0.014       0.021    NA    0
dLakers                     0.005 0.004      -0.002     0.005       0.012    NA    0
dMagic                     -0.014 0.004      -0.021    -0.014      -0.007    NA    0
dMavericks                 -0.018 0.004      -0.025    -0.018      -0.011    NA    0
dNets                       0.004 0.004      -0.003     0.004       0.011    NA    0
dNuggets                    0.016 0.004       0.009     0.016       0.023    NA    0
dPacers                    -0.019 0.004      -0.026    -0.019      -0.012    NA    0
dPelicans                  -0.007 0.004      -0.014    -0.007       0.000    NA    0
dPistons                   -0.034 0.004      -0.041    -0.034      -0.027    NA    0
dRaptors                   -0.002 0.004      -0.009    -0.002       0.005    NA    0
dRockets                   -0.011 0.004      -0.018    -0.011      -0.004    NA    0
dSpurs                     -0.052 0.004      -0.059    -0.052      -0.045    NA    0
dSuns                       0.035 0.004       0.028     0.035       0.042    NA    0
dThunder                    0.002 0.004      -0.005     0.002       0.009    NA    0
dTimberwolves               0.019 0.004       0.011     0.019       0.026    NA    0
dTrail Blazers             -0.013 0.004      -0.020    -0.013      -0.006    NA    0
dWarriors                   0.027 0.004       0.020     0.027       0.034    NA    0
dWizards                    0.021 0.004       0.014     0.021       0.028    NA    0
h                           0.028 0.001       0.026     0.028       0.030    NA    0
scale(PTS_cur_season)       0.088 0.001       0.086     0.088       0.090    NA    0
scale(FT_PCT_cur_season)   -0.051 0.001      -0.053    -0.051      -0.049    NA    0
scale(FG3_PCT_cur_season)  -0.018 0.001      -0.019    -0.018      -0.016    NA    0

Deviance Information Criterion (DIC) ...............: 365778.05
Deviance Information Criterion (DIC, saturated) ....: 796512.89
Effective number of parameters .....................: 63.00

Marginal log-Likelihood:  -183419.51
CPO, PIT is computed
Posterior summaries for the linear predictor and the fitted values are computed
(Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')


Logistic:
formula = y.binary~1+h+a+d+scale(PTS_cur_season)+scale(PTS_LOST_cur_season)
Call:
   c("inla.core(formula = formula, family = family, contrasts = contrasts, ", " data = data, quantiles
        =
   quantiles, E = E, offset = offset, ", " scale = scale, weights = weights, Ntrials = Ntrials, strata
        =
   strata, ", " lp.scale = lp.scale, link.covariates = link.covariates, verbose = verbose, ", " lincomb
        =
   lincomb, selection = selection, control.compute = control.compute, ", " control.predictor =
   control.predictor, control.family = control.family, ", " control.inla = control.inla, control.fixed
        =
   control.fixed, ", " control.mode = control.mode, control.expert = control.expert, ", " control.
        hazard
   = control.hazard, control.lincomb = control.lincomb, ", " control.update = control.update,
   control.lp.scale = control.lp.scale, ", " control.pardiso = control.pardiso, only.hyperparam =
   only.hyperparam, ", " inla.call = inla.call, inla.arg = inla.arg, num.threads = num.threads, ", "
   blas.num.threads = blas.num.threads, keep = keep, working.directory = working.directory, ", " silent
        =
   silent, inla.mode = inla.mode, safe = FALSE, debug = debug, ", " .parent.frame = .parent.frame)")
Time used:
    Pre = 0.968, Running = 4.06, Post = 0.332, Total = 5.36
Fixed effects:
                             mean    sd  0.025quant 0.5quant 0.975quant mode kld
(Intercept)                -0.260 0.078      -0.412    -0.260      -0.108   NA   0
aBucks                      0.023 0.076      -0.127     0.023       0.172   NA   0
aBulls                      0.146 0.076      -0.003     0.146       0.294   NA   0
aCavaliers                  0.215 0.076       0.067     0.215       0.363   NA   0
aCeltics                    0.090 0.076      -0.058     0.090       0.238   NA   0
aClippers                   0.243 0.076       0.094     0.243       0.393   NA   0
aGrizzlies                  0.181 0.076       0.031     0.181       0.330   NA   0
aHawks                      0.077 0.076      -0.071     0.077       0.226   NA   0
aHeat                       0.280 0.075       0.133     0.280       0.428   NA   0
aHornets                   -0.059 0.077      -0.211    -0.059       0.093   NA   0
aJazz                       0.202 0.076       0.052     0.202       0.352   NA   0
```

```
aKings                       0.034 0.078     -0.118    0.034      0.187   NA   0
aKnicks                     -0.100 0.077     -0.252   -0.100      0.052   NA   0
aLakers                      0.220 0.076      0.071    0.220      0.369   NA   0
aMagic                      -0.065 0.077     -0.215   -0.065      0.085   NA   0
aMavericks                   0.345 0.076      0.196    0.345      0.494   NA   0
aNets                        0.035 0.076     -0.114    0.035      0.185   NA   0
aNuggets                     0.263 0.076      0.113    0.263      0.412   NA   0
aPacers                      0.119 0.076     -0.029    0.119      0.267   NA   0
aPelicans                    0.126 0.077     -0.024    0.126      0.277   NA   0
aPistons                     0.020 0.076     -0.129    0.020      0.169   NA   0
aRaptors                     0.029 0.076     -0.120    0.029      0.178   NA   0
aRockets                     0.303 0.076      0.153    0.303      0.453   NA   0
aSpurs                       0.503 0.077      0.351    0.503      0.654   NA   0
aSuns                        0.219 0.077      0.068    0.219      0.369   NA   0
aThunder                     0.294 0.076      0.145    0.294      0.444   NA   0
aTimberwolves               -0.201 0.079     -0.355   -0.201     -0.047   NA   0
aTrail Blazers               0.196 0.076      0.046    0.196      0.345   NA   0
aWarriors                    0.273 0.077      0.123    0.273      0.424   NA   0
aWizards                     0.044 0.076     -0.105    0.044      0.193   NA   0
dBucks                      -0.112 0.076     -0.260   -0.112      0.036   NA   0
dBulls                      -0.266 0.075     -0.414   -0.266     -0.118   NA   0
dCavaliers                  -0.288 0.075     -0.435   -0.288     -0.141   NA   0
dCeltics                    -0.504 0.075     -0.650   -0.504     -0.357   NA   0
dClippers                   -0.433 0.076     -0.581   -0.433     -0.285   NA   0
dGrizzlies                  -0.259 0.076     -0.408   -0.259     -0.110   NA   0
dHawks                      -0.120 0.075     -0.267   -0.120      0.028   NA   0
dHeat                       -0.522 0.075     -0.668   -0.522     -0.375   NA   0
dHornets                     0.229 0.077      0.078    0.229      0.380   NA   0
dJazz                       -0.471 0.076     -0.620   -0.471     -0.322   NA   0
dKings                       0.156 0.078      0.004    0.156      0.308   NA   0
dKnicks                      0.229 0.077      0.078    0.229      0.381   NA   0
dLakers                     -0.395 0.075     -0.543   -0.395     -0.247   NA   0
dMagic                      -0.104 0.076     -0.252   -0.104      0.045   NA   0
dMavericks                  -0.610 0.076     -0.759   -0.610     -0.462   NA   0
dNets                        0.078 0.076     -0.071    0.078      0.227   NA   0
dNuggets                    -0.574 0.076     -0.722   -0.574     -0.425   NA   0
dPacers                     -0.294 0.075     -0.442   -0.294     -0.146   NA   0
dPelicans                   -0.115 0.077     -0.265   -0.115      0.035   NA   0
dPistons                    -0.127 0.076     -0.275   -0.127      0.021   NA   0
dRaptors                    -0.298 0.075     -0.446   -0.298     -0.150   NA   0
dRockets                    -0.659 0.076     -0.808   -0.659     -0.510   NA   0
dSpurs                      -0.989 0.076     -1.138   -0.989     -0.839   NA   0
dSuns                       -0.374 0.076     -0.523   -0.374     -0.225   NA   0
dThunder                    -0.487 0.076     -0.635   -0.487     -0.339   NA   0
dTimberwolves                0.257 0.078      0.103    0.257      0.410   NA   0
dTrail Blazers              -0.332 0.076     -0.481   -0.332     -0.183   NA   0
dWarriors                   -0.569 0.076     -0.717   -0.569     -0.420   NA   0
dWizards                     0.016 0.076     -0.132    0.016      0.165   NA   0
h                            0.786 0.020      0.748    0.786      0.825   NA   0
scale(PTS_cur_season)        1.059 0.028      1.005    1.059      1.114   NA   0
scale(PTS_LOST_cur_season)  -1.061 0.028     -1.115   -1.061     -1.007   NA   0

Deviance Information Criterion (DIC) ...............: 60111.04
Deviance Information Criterion (DIC, saturated) ....: 60111.04
Effective number of parameters .....................: 61.94

Marginal log-Likelihood:  -30386.60
CPO, PIT is computed
Posterior summaries for the linear predictor and the fitted values are computed
(Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

- M4

```
Poisson:
formula = y~1+a+d+h
          +scale(PTS_cur_season)
          +scale(FG_PCT_cur_season)
          +scale(FT_PCT_cur_season)
          +scale(FG3_PCT_cur_season)
          +scale(AST_cur_season)
          +scale(REB_cur_season)
          +scale(WINRATE_cur_season)
          +scale(PTS_LOST_cur_season)
          +log(salary)
Call:
   c("inla.core(formula = formula, family = family, contrasts = contrasts, ", " data = data, quantiles
        =
   quantiles, E = E, offset = offset, ", " scale = scale, weights = weights, Ntrials = Ntrials, strata
        =
   strata, ", " lp.scale = lp.scale, link.covariates = link.covariates, verbose = verbose, ", " lincomb
        =
   lincomb, selection = selection, control.compute = control.compute, ", " control.predictor =
   control.predictor, control.family = control.family, ", " control.inla = control.inla, control.fixed
        =
   control.fixed, ", " control.mode = control.mode, control.expert = control.expert, ", " control.
        hazard
   = control.hazard, control.lincomb = control.lincomb, ", " control.update = control.update,
   control.lp.scale = control.lp.scale, ", " control.pardiso = control.pardiso, only.hyperparam =
   only.hyperparam, ", " inla.call = inla.call, inla.arg = inla.arg, num.threads = num.threads, ", "
   blas.num.threads = blas.num.threads, keep = keep, working.directory = working.directory, ", " silent
        =
   silent, inla.mode = inla.mode, safe = FALSE, debug = debug, ", " .parent.frame = .parent.frame)")
Time used:
    Pre = 0.859, Running = 6.85, Post = 0.393, Total = 8.1
Fixed effects:
                              mean    sd 0.025quant 0.5quant 0.975quant mode kld
(Intercept)                  3.533 0.046      3.443    3.533      3.623   NA   0
aBucks                       0.008 0.004      0.001    0.008      0.015   NA   0
aBulls                       0.002 0.004     -0.005    0.002      0.009   NA   0
aCavaliers                   0.001 0.004     -0.006    0.001      0.008   NA   0
aCeltics                     0.006 0.004     -0.001    0.006      0.013   NA   0
aClippers                    0.011 0.004      0.004    0.011      0.018   NA   0
```

```
aGrizzlies                  -0.001 0.004     -0.009   -0.001     0.006   NA   0
aHawks                       0.011 0.004      0.004    0.011     0.018   NA   0
aHeat                        0.003 0.004     -0.004    0.003     0.011   NA   0
aHornets                    -0.005 0.004     -0.012   -0.005     0.003   NA   0
aJazz                        0.015 0.004      0.008    0.015     0.022   NA   0
aKings                       0.021 0.004      0.014    0.021     0.028   NA   0
aKnicks                     -0.008 0.004     -0.015   -0.008    -0.001   NA   0
aLakers                      0.001 0.004     -0.006    0.001     0.008   NA   0
aMagic                       0.002 0.004     -0.005    0.002     0.009   NA   0
aMavericks                   0.009 0.004      0.002    0.009     0.016   NA   0
aNets                       -0.002 0.004     -0.009   -0.002     0.005   NA   0
aNuggets                     0.024 0.004      0.017    0.024     0.031   NA   0
aPacers                      0.005 0.004     -0.002    0.005     0.012   NA   0
aPelicans                    0.004 0.004     -0.003    0.004     0.011   NA   0
aPistons                    -0.010 0.004     -0.018   -0.010    -0.003   NA   0
aRaptors                     0.009 0.004      0.002    0.009     0.016   NA   0
aRockets                     0.009 0.004      0.002    0.009     0.016   NA   0
aSpurs                       0.022 0.004      0.015    0.022     0.029   NA   0
aSuns                        0.037 0.004      0.030    0.037     0.044   NA   0
aThunder                     0.029 0.004      0.022    0.029     0.037   NA   0
aTimberwolves                0.006 0.004     -0.001    0.006     0.014   NA   0
aTrail Blazers               0.009 0.004      0.002    0.009     0.017   NA   0
aWarriors                    0.024 0.004      0.017    0.024     0.031   NA   0
aWizards                     0.009 0.004      0.002    0.009     0.017   NA   0
dBucks                       0.002 0.004     -0.005    0.002     0.009   NA   0
dBulls                      -0.023 0.004     -0.030   -0.023    -0.016   NA   0
dCavaliers                  -0.010 0.004     -0.017   -0.010    -0.003   NA   0
dCeltics                    -0.031 0.004     -0.038   -0.031    -0.024   NA   0
dClippers                   -0.009 0.004     -0.016   -0.009    -0.002   NA   0
dGrizzlies                  -0.026 0.004     -0.033   -0.026    -0.019   NA   0
dHawks                       0.001 0.004     -0.006    0.001     0.008   NA   0
dHeat                       -0.033 0.004     -0.040   -0.033    -0.026   NA   0
dHornets                    -0.004 0.004     -0.011   -0.004     0.003   NA   0
dJazz                       -0.030 0.004     -0.037   -0.030    -0.023   NA   0
dKings                       0.032 0.004      0.025    0.032     0.039   NA   0
dKnicks                      0.016 0.004      0.009    0.016     0.023   NA   0
dLakers                      0.005 0.004     -0.002    0.005     0.012   NA   0
dMagic                      -0.013 0.004     -0.020   -0.013    -0.006   NA   0
dMavericks                  -0.017 0.004     -0.024   -0.017    -0.010   NA   0
dNets                        0.005 0.004     -0.002    0.005     0.012   NA   0
dNuggets                     0.017 0.004      0.010    0.017     0.024   NA   0
dPacers                     -0.018 0.004     -0.025   -0.018    -0.011   NA   0
dPelicans                   -0.006 0.004     -0.013   -0.006     0.001   NA   0
dPistons                    -0.031 0.004     -0.038   -0.031    -0.024   NA   0
dRaptors                    -0.002 0.004     -0.009   -0.002     0.005   NA   0
dRockets                    -0.011 0.004     -0.018   -0.011    -0.004   NA   0
dSpurs                      -0.051 0.004     -0.058   -0.051    -0.044   NA   0
dSuns                        0.035 0.004      0.028    0.035     0.042   NA   0
dThunder                     0.004 0.004     -0.003    0.004     0.011   NA   0
dTimberwolves                0.020 0.004      0.013    0.020     0.027   NA   0
dTrail Blazers              -0.012 0.004     -0.019   -0.012    -0.005   NA   0
dWarriors                    0.028 0.004      0.021    0.028     0.035   NA   0
dWizards                     0.021 0.004      0.014    0.021     0.028   NA   0
h                            0.028 0.001      0.026    0.028     0.030   NA   0
scale(PTS_cur_season)        0.114 0.003      0.108    0.114     0.120   NA   0
scale(FG_PCT_cur_season)    -0.052 0.001     -0.055   -0.052    -0.049   NA   0
scale(FT_PCT_cur_season)    -0.018 0.001     -0.020   -0.018    -0.016   NA   0
scale(FG3_PCT_cur_season)   -0.005 0.001     -0.007   -0.005    -0.003   NA   0
scale(AST_cur_season)        0.009 0.001      0.007    0.009     0.011   NA   0
scale(REB_cur_season)       -0.016 0.001     -0.018   -0.016    -0.014   NA   0
scale(WINRATE_cur_season)   -0.003 0.001     -0.005   -0.003     0.000   NA   0
scale(PTS_LOST_cur_season)  -0.016 0.003     -0.021   -0.016    -0.011   NA   0
log(salary)                  0.058 0.002      0.053    0.058     0.063   NA   0

Deviance Information Criterion (DIC) ...............: 362426.60
Deviance Information Criterion (DIC, saturated) ....: 793161.44
Effective number of parameters .....................: 69.00

Marginal log-Likelihood:  -181801.71
CPO, PIT is computed
Posterior summaries for the linear predictor and the fitted values are computed
(Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')


Logistic:
formula = y.binary~1+a+d+h
          +scale(PTS_cur_season)
          +scale(FG_PCT_cur_season)
          +scale(AST_cur_season)
          +scale(WINRATE_cur_season)
          +scale(PTS_LOST_cur_season)
          +log(salary)
Call:
   c("inla.core(formula = formula, family = family, contrasts = contrasts, ", " data = data, quantiles
        =
   quantiles, E = E, offset = offset, ", " scale = scale, weights = weights, Ntrials = Ntrials, strata
        =
   strata, ", " lp.scale = lp.scale, link.covariates = link.covariates, verbose = verbose, ", " lincomb
        =
   lincomb, selection = selection, control.compute = control.compute, ", " control.predictor =
   control.predictor, control.family = control.family, ", " control.inla = control.inla, control.fixed
        =
   control.fixed, ", " control.mode = control.mode, control.expert = control.expert, ", " control.
        hazard
   = control.hazard, control.lincomb = control.lincomb, ", " control.update = control.update,
   control.lp.scale = control.lp.scale, ", " control.pardiso = control.pardiso, only.hyperparam =
   only.hyperparam, ", " inla.call = inla.call, inla.arg = inla.arg, num.threads = num.threads, ", "
   blas.num.threads = blas.num.threads, keep = keep, working.directory = working.directory, ", " silent
        =
   silent, inla.mode = inla.mode, safe = FALSE, debug = debug, ", " .parent.frame = .parent.frame)")
Time used:
    Pre = 1.13, Running = 4.19, Post = 0.37, Total = 5.69
Fixed effects:
```

```
                               mean    sd 0.025quant 0.5quant 0.975quant mode kld
(Intercept)                  -7.341 0.956     -9.214   -7.341     -5.468   NA   0
aBucks                        0.040 0.076     -0.110    0.040      0.190   NA   0
aBulls                        0.153 0.076      0.004    0.153      0.302   NA   0
aCavaliers                    0.193 0.076      0.044    0.193      0.342   NA   0
aCeltics                      0.066 0.076     -0.083    0.066      0.215   NA   0
aClippers                     0.247 0.076      0.097    0.247      0.396   NA   0
aGrizzlies                    0.180 0.077      0.030    0.180      0.330   NA   0
aHawks                        0.104 0.076     -0.046    0.104      0.254   NA   0
aHeat                         0.243 0.076      0.094    0.243      0.392   NA   0
aHornets                      0.016 0.078     -0.138    0.016      0.169   NA   0
aJazz                         0.226 0.077      0.075    0.226      0.377   NA   0
aKings                        0.100 0.078     -0.054    0.100      0.254   NA   0
aKnicks                      -0.137 0.079     -0.291   -0.137      0.017   NA   0
aLakers                       0.190 0.076      0.040    0.190      0.340   NA   0
aMagic                       -0.060 0.077     -0.211   -0.060      0.090   NA   0
aMavericks                    0.315 0.077      0.165    0.315      0.466   NA   0
aNets                         0.024 0.077     -0.126    0.024      0.175   NA   0
aNuggets                      0.308 0.077      0.157    0.308      0.459   NA   0
aPacers                       0.150 0.076      0.002    0.150      0.299   NA   0
aPelicans                     0.146 0.077     -0.005    0.146      0.297   NA   0
aPistons                      0.024 0.076     -0.125    0.024      0.173   NA   0
aRaptors                      0.080 0.077     -0.070    0.080      0.230   NA   0
aRockets                      0.364 0.077      0.213    0.364      0.516   NA   0
aSpurs                        0.471 0.078      0.319    0.471      0.624   NA   0
aSuns                         0.284 0.077      0.133    0.284      0.436   NA   0
aThunder                      0.359 0.077      0.208    0.359      0.511   NA   0
aTimberwolves                -0.152 0.079     -0.307   -0.152      0.003   NA   0
aTrail Blazers                0.218 0.077      0.067    0.218      0.369   NA   0
aWarriors                     0.290 0.077      0.138    0.290      0.441   NA   0
aWizards                      0.079 0.076     -0.071    0.079      0.229   NA   0
dBucks                       -0.113 0.076     -0.261   -0.113      0.035   NA   0
dBulls                       -0.264 0.076     -0.412   -0.264     -0.116   NA   0
dCavaliers                   -0.285 0.075     -0.432   -0.285     -0.138   NA   0
dCeltics                     -0.505 0.075     -0.652   -0.505     -0.358   NA   0
dClippers                    -0.433 0.076     -0.581   -0.433     -0.284   NA   0
dGrizzlies                   -0.258 0.076     -0.407   -0.258     -0.108   NA   0
dHawks                       -0.120 0.075     -0.267   -0.120      0.028   NA   0
dHeat                        -0.522 0.075     -0.668   -0.522     -0.375   NA   0
dHornets                      0.231 0.077      0.080    0.231      0.382   NA   0
dJazz                        -0.474 0.076     -0.623   -0.474     -0.325   NA   0
dKings                        0.157 0.078      0.005    0.157      0.309   NA   0
dKnicks                       0.229 0.077      0.077    0.229      0.380   NA   0
dLakers                      -0.395 0.075     -0.543   -0.395     -0.247   NA   0
dMagic                       -0.102 0.076     -0.251   -0.102      0.046   NA   0
dMavericks                   -0.610 0.076     -0.759   -0.610     -0.462   NA   0
dNets                         0.079 0.076     -0.070    0.079      0.228   NA   0
dNuggets                     -0.576 0.076     -0.725   -0.576     -0.427   NA   0
dPacers                      -0.294 0.075     -0.442   -0.294     -0.146   NA   0
dPelicans                    -0.115 0.077     -0.266   -0.115      0.035   NA   0
dPistons                     -0.126 0.076     -0.275   -0.126      0.022   NA   0
dRaptors                     -0.301 0.076     -0.449   -0.301     -0.153   NA   0
dRockets                     -0.662 0.076     -0.811   -0.662     -0.513   NA   0
dSpurs                       -0.992 0.076     -1.141   -0.992     -0.842   NA   0
dSuns                        -0.374 0.076     -0.523   -0.374     -0.225   NA   0
dThunder                     -0.485 0.076     -0.633   -0.485     -0.336   NA   0
dTimberwolves                 0.259 0.078      0.106    0.259      0.412   NA   0
dTrail Blazers               -0.333 0.076     -0.482   -0.333     -0.184   NA   0
dWarriors                    -0.570 0.076     -0.719   -0.570     -0.422   NA   0
dWizards                      0.017 0.076     -0.132    0.017      0.166   NA   0
h                             0.787 0.020      0.749    0.787      0.826   NA   0
scale(PTS_cur_season)         0.669 0.065      0.542    0.669      0.797   NA   0
scale(FG_PCT_cur_season)      0.079 0.027      0.027    0.079      0.131   NA   0
scale(AST_cur_season)         0.062 0.020      0.023    0.062      0.102   NA   0
scale(WINRATE_cur_season)     0.091 0.024      0.043    0.091      0.138   NA   0
scale(PTS_LOST_cur_season)   -0.846 0.056     -0.956   -0.846     -0.737   NA   0
log(salary)                   0.385 0.052      0.283    0.385      0.486   NA   0

Deviance Information Criterion (DIC) ...............: 60036.85
Deviance Information Criterion (DIC, saturated) ....: 60036.85
Effective number of parameters .....................: 65.94

Marginal log-Likelihood:  -30373.68
CPO, PIT is computed
Posterior summaries for the linear predictor and the fitted values are computed
(Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

- M5

```
Poisson:
formula = y~1+h
          +scale(PTS_cur_season)
          +scale(FG_PCT_cur_season)
          +scale(AST_cur_season)
          +scale(REB_cur_season)
          +log(salary)
          + f(a.season, model = "generic0", Cmatrix = Q.a, rankdef = 1, constr = TRUE)
          + f(d.season, model = "generic0", Cmatrix = Q.d, rankdef = 1, constr = TRUE)
Call:
   c("inla.core(formula = formula, family = family, contrasts = contrasts, ", " data = data, quantiles
      =
   quantiles, E = E, offset = offset, ", " scale = scale, weights = weights, Ntrials = Ntrials, strata
      =
   strata, ", " lp.scale = lp.scale, link.covariates = link.covariates, verbose = verbose, ", " lincomb
      =
   lincomb, selection = selection, control.compute = control.compute, ", " control.predictor =
   control.predictor, control.family = control.family, ", " control.inla = control.inla, control.fixed
      =
   control.fixed, ", " control.mode = control.mode, control.expert = control.expert, ", " control.
      hazard
   = control.hazard, control.lincomb = control.lincomb, ", " control.update = control.update,
   control.lp.scale = control.lp.scale, ", " control.pardiso = control.pardiso, only.hyperparam =
    only.hyperparam, ", " inla.call = inla.call, inla.arg = inla.arg, num.threads = num.threads, ", "
    blas.num.threads = blas.num.threads, keep = keep, working.directory = working.directory, ", " silent
         =
    silent, inla.mode = inla.mode, safe = FALSE, debug = debug, ", " .parent.frame = .parent.frame)")
Time used:
     Pre = 0.568, Running = 24.1, Post = 0.59, Total = 25.3
Fixed effects:
                              mean    sd  0.025quant  0.5quant  0.975quant  mode  kld
(Intercept)                  4.106 0.174      3.764     4.107      4.448    NA    0
h                            0.027 0.001      0.026     0.027      0.029    NA    0
scale(PTS_cur_season)        0.025 0.003      0.019     0.025      0.031    NA    0
scale(FG_PCT_cur_season)    -0.017 0.002     -0.022    -0.017     -0.013    NA    0
scale(AST_cur_season)        0.007 0.002      0.004     0.007      0.010    NA    0
scale(REB_cur_season)       -0.006 0.001     -0.009    -0.006     -0.003    NA    0
log(salary)                  0.027 0.009      0.009     0.027      0.046    NA    0

Random effects:
  Name     Model
    a.season Generic0 model
   d.season Generic0 model

Model hyperparameters:
                           mean      sd  0.025quant  0.5quant  0.975quant  mode
Precision for a.season 1735.37 171.75     1420.93   1727.23     2096.94    NA
Precision for d.season 1056.55  83.87      900.24   1053.53     1230.50    NA

Deviance Information Criterion (DIC) ...............: 355661.73
Deviance Information Criterion (DIC, saturated) ....: 786396.57
Effective number of parameters .....................: 811.47

Marginal log-Likelihood:  -178584.33
CPO, PIT is computed
Posterior summaries for the linear predictor and the fitted values are computed
(Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')


Logistic:
formula = y.binary~1+h
          +scale(PTS_cur_season)
          +scale(WINRATE_cur_season)
          +scale(PTS_LOST_cur_season)
          + log(salary)
          + f(a.season, model = "generic0", Cmatrix = Q.a, rankdef = 1, constr = TRUE)
          + f(d.season, model = "generic0", Cmatrix = Q.d, rankdef = 1, constr = TRUE)
Call:
    c("inla.core(formula = formula, family = family, contrasts = contrasts, ", " data = data, quantiles
         =
    quantiles, E = E, offset = offset, ", " scale = scale, weights = weights, Ntrials = Ntrials, strata
         =
    strata, ", " lp.scale = lp.scale, link.covariates = link.covariates, verbose = verbose, ", " lincomb
         =
    lincomb, selection = selection, control.compute = control.compute, ", " control.predictor =
    control.predictor, control.family = control.family, ", " control.inla = control.inla, control.fixed
         =
    control.fixed, ", " control.mode = control.mode, control.expert = control.expert, ", " control.
         hazard
    = control.hazard, control.lincomb = control.lincomb, ", " control.update = control.update,
    control.lp.scale = control.lp.scale, ", " control.pardiso = control.pardiso, only.hyperparam =
    only.hyperparam, ", " inla.call = inla.call, inla.arg = inla.arg, num.threads = num.threads, ", "
    blas.num.threads = blas.num.threads, keep = keep, working.directory = working.directory, ", " silent
         =
    silent, inla.mode = inla.mode, safe = FALSE, debug = debug, ", " .parent.frame = .parent.frame)")
Time used:
     Pre = 0.439, Running = 68.9, Post = 0.692, Total = 70
Fixed effects:
                                mean    sd  0.025quant  0.5quant  0.975quant  mode  kld
(Intercept)                  -13.593 2.929    -19.332   -13.596     -7.838    NA    0
h                              0.850 0.020      0.810     0.850      0.890    NA    0
scale(PTS_cur_season)          0.610 0.077      0.458     0.610      0.762    NA    0
scale(WINRATE_cur_season)     -0.095 0.030     -0.154    -0.095     -0.036    NA    0
scale(PTS_LOST_cur_season)    -0.584 0.069     -0.719    -0.584     -0.447    NA    0
log(salary)                    0.718 0.160      0.404     0.718      1.030    NA    0

Random effects:
  Name     Model
    a.season Generic0 model
   d.season Generic0 model

Model hyperparameters:
                        mean    sd  0.025quant  0.5quant  0.975quant  mode
Precision for a.season 7.77 1.040        5.89      7.72        9.98    NA
Precision for d.season 4.34 0.381        3.63      4.33        5.13    NA

Deviance Information Criterion (DIC) ...............: 56568.11
Deviance Information Criterion (DIC, saturated) ....: 56568.11
Effective number of parameters .....................: 689.88

Marginal log-Likelihood:  -28802.54
CPO, PIT is computed
Posterior summaries for the linear predictor and the fitted values are computed
(Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

- M6

```
Poisson:
formula = y~1+h
          +scale(FT_PCT_cur_season)
          +scale(AST_cur_season)
          +scale(REB_cur_season)
          +scale(WINRATE_cur_season)
          +scale(PTS_LOST_cur_season)
          +log(salary)
          + f(a.season, model = "generic0", Cmatrix = Q.a, rankdef = 1, constr = TRUE)
          + f(d.season, model = "generic0", Cmatrix = Q.d, rankdef = 1, constr = TRUE)
          + f(inla.group(travel), model = "rw1", scale.model = TRUE)
Call:
   c("inla.core(formula = formula, family = family, contrasts = contrasts, ", " data = data, quantiles
        =
   quantiles, E = E, offset = offset, ", " scale = scale, weights = weights, Ntrials = Ntrials, strata
        =
   strata, ", " lp.scale = lp.scale, link.covariates = link.covariates, verbose = verbose, ", " lincomb
        =
   lincomb, selection = selection, control.compute = control.compute, ", " control.predictor =
   control.predictor, control.family = control.family, ", " control.inla = control.inla, control.fixed
        =
   control.fixed, ", " control.mode = control.mode, control.expert = control.expert, ", " control.
        hazard
   = control.hazard, control.lincomb = control.lincomb, ", " control.update = control.update,
   control.lp.scale = control.lp.scale, ", " control.pardiso = control.pardiso, only.hyperparam =
   only.hyperparam, ", " inla.call = inla.call, inla.arg = inla.arg, num.threads = num.threads, ", "
   blas.num.threads = blas.num.threads, keep = keep, working.directory = working.directory, ", " silent
        =
   silent, inla.mode = inla.mode, safe = FALSE, debug = debug, ", " .parent.frame = .parent.frame)")
Time used:
    Pre = 5.55, Running = 11.7, Post = 0.394, Total = 17.6
Fixed effects:
                              mean    sd 0.025quant 0.5quant 0.975quant mode kld
(Intercept)                  4.070 0.192      3.692    4.070      4.446   NA   0
h                            0.032 0.003      0.026    0.032      0.038   NA   0
scale(FT_PCT_cur_season)    -0.002 0.002     -0.005   -0.002      0.001   NA   0
scale(AST_cur_season)        0.006 0.002      0.003    0.006      0.009   NA   0
scale(REB_cur_season)       -0.004 0.001     -0.006   -0.004     -0.001   NA   0
scale(WINRATE_cur_season)    0.002 0.001      0.000    0.002      0.004   NA   0
scale(PTS_LOST_cur_season)   0.006 0.002      0.002    0.006      0.010   NA   0
log(salary)                  0.029 0.010      0.009    0.029      0.050   NA   0

Random effects:
  Name    Model
    a.season Generic0 model
   d.season Generic0 model
   inla.group(travel) RW1 model

Model hyperparameters:
                                    mean       sd 0.025quant 0.5quant 0.975quant mode
Precision for a.season           1320.60   120.13    1099.70  1315.21    1572.62   NA
Precision for d.season           1043.86    83.68     888.40  1040.65    1217.89   NA
Precision for inla.group(travel) 60698.89 29196.02   21450.29 54964.61  133750.98   NA

Deviance Information Criterion (DIC) ...............: 355615.96
Deviance Information Criterion (DIC, saturated) ....: 786350.79
Effective number of parameters .....................: 839.96

Marginal log-Likelihood:  -178636.93
CPO, PIT is computed
Posterior summaries for the linear predictor and the fitted values are computed
(Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')


Logistic:
formula = y.binary~~1+h
          +scale(PTS_cur_season)
          +scale(WINRATE_cur_season)
          +scale(PTS_LOST_cur_season)
          +log(salary)
          + f(a.season, model = "generic0", Cmatrix = Q.a, rankdef = 1, constr = TRUE)
          + f(d.season, model = "generic0", Cmatrix = Q.d, rankdef = 1, constr = TRUE)
          + f(inla.group(travel), model = "rw1", scale.model = TRUE)
Call:
   c("inla.core(formula = formula, family = family, contrasts = contrasts, ", " data = data, quantiles
        =
   quantiles, E = E, offset = offset, ", " scale = scale, weights = weights, Ntrials = Ntrials, strata
        =
   strata, ", " lp.scale = lp.scale, link.covariates = link.covariates, verbose = verbose, ", " lincomb
        =
   lincomb, selection = selection, control.compute = control.compute, ", " control.predictor =
   control.predictor, control.family = control.family, ", " control.inla = control.inla, control.fixed
        =
   control.fixed, ", " control.mode = control.mode, control.expert = control.expert, ", " control.
        hazard
   = control.hazard, control.lincomb = control.lincomb, ", " control.update = control.update,
   control.lp.scale = control.lp.scale, ", " control.pardiso = control.pardiso, only.hyperparam =
   only.hyperparam, ", " inla.call = inla.call, inla.arg = inla.arg, num.threads = num.threads, ", "
   blas.num.threads = blas.num.threads, keep = keep, working.directory = working.directory, ", " silent
        =
   silent, inla.mode = inla.mode, safe = FALSE, debug = debug, ", " .parent.frame = .parent.frame)")
Time used:
    Pre = 4.86, Running = 40.9, Post = 0.371, Total = 46.1
Fixed effects:
                               mean    sd 0.025quant 0.5quant 0.975quant mode kld
(Intercept)                 -13.581 2.939    -19.338  -13.585     -7.805   NA   0
h                             0.852 0.024      0.806    0.852      0.899   NA   0
scale(PTS_cur_season)         0.609 0.078      0.457    0.610      0.761   NA   0
scale(WINRATE_cur_season)    -0.096 0.030     -0.155   -0.095     -0.036   NA   0
scale(PTS_LOST_cur_season)   -0.583 0.070     -0.719   -0.583     -0.446   NA   0
log(salary)                   0.717 0.160      0.402    0.717      1.031   NA   0

Random effects:
  Name    Model
    a.season Generic0 model
   d.season Generic0 model
   inla.group(travel) RW1 model

Model hyperparameters:
                                    mean       sd 0.025quant 0.5quant 0.975quant mode
Precision for a.season              7.76     1.04       5.93     7.68      10.01   NA
Precision for d.season              4.34     0.38       3.63     4.32       5.13   NA
Precision for inla.group(travel) 18658.51 19935.13     394.69 11716.90   71266.97   NA

Deviance Information Criterion (DIC) ...............: 56568.00
Deviance Information Criterion (DIC, saturated) ....: 56568.00
Effective number of parameters .....................: 691.30

Marginal log-Likelihood:  -28820.43
CPO, PIT is computed
Posterior summaries for the linear predictor and the fitted values are computed
(Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```

- M7

```
Poisson:
formula = y~1+h
          +scale(AST_cur_season)
          +scale(REB_cur_season)
          +scale(PTS_LOST_cur_season)
          +log(salary)
          + f(a.season, model = "generic0", Cmatrix = Q.a, rankdef = 1, constr = TRUE)
          + f(d.season, model = "generic0", Cmatrix = Q.d, rankdef = 1, constr = TRUE)
          + f(inla.group(travel), model = "rw1", scale.model = TRUE)
          + type:a + type:d
Call:
   c("inla.core(formula = formula, family = family, contrasts = contrasts, ", " data = data, quantiles
       =
   quantiles, E = E, offset = offset, ", " scale = scale, weights = weights, Ntrials = Ntrials, strata
       =
   strata, ", " lp.scale = lp.scale, link.covariates = link.covariates, verbose = verbose, ", " lincomb
       =
   lincomb, selection = selection, control.compute = control.compute, ", " control.predictor =
   control.predictor, control.family = control.family, ", " control.inla = control.inla, control.fixed
       =
   control.fixed, ", " control.mode = control.mode, control.expert = control.expert, ", " control.
       hazard
   = control.hazard, control.lincomb = control.lincomb, ", " control.update = control.update,
   control.lp.scale = control.lp.scale, ", " control.pardiso = control.pardiso, only.hyperparam =
   only.hyperparam, ", " inla.call = inla.call, inla.arg = inla.arg, num.threads = num.threads, ", "
   blas.num.threads = blas.num.threads, keep = keep, working.directory = working.directory, ", " silent
       =
   silent, inla.mode = inla.mode, safe = FALSE, debug = debug, ", " .parent.frame = .parent.frame)")
Time used:
    Pre = 9.42, Running = 39.2, Post = 1.43, Total = 50.1
Fixed effects:
                             mean    sd 0.025quant 0.5quant 0.975quant mode kld
(Intercept)                 3.771 2.724     -1.570    3.771      9.113   NA   0
h                           0.032 0.003      0.026    0.032      0.038   NA   0
scale(AST_cur_season)       0.006 0.001      0.003    0.006      0.009   NA   0
scale(REB_cur_season)      -0.003 0.001     -0.006   -0.003     -0.001   NA   0
scale(PTS_LOST_cur_season)  0.003 0.002      0.000    0.003      0.006   NA   0
log(salary)                 0.046 0.011      0.023    0.046      0.068   NA   0
typeMM:a76ers               0.004 9.698    -19.015    0.004     19.023   NA   0
typeMS:a76ers               0.032 9.698    -18.987    0.032     19.051   NA   0
typeMW:a76ers              -0.018 9.698    -19.037   -0.018     19.001   NA   0
typeSM:a76ers               0.037 9.698    -18.982    0.037     19.056   NA   0
typeSS:a76ers              -0.034 9.698    -19.053   -0.034     18.985   NA   0
typeSW:a76ers              -0.016 9.698    -19.035   -0.016     19.003   NA   0
typeWM:a76ers              -0.021 9.698    -19.040   -0.021     18.998   NA   0
typeWS:a76ers               0.013 9.698    -19.006    0.013     19.032   NA   0
typeWW:a76ers              -0.017 9.698    -19.036   -0.017     19.002   NA   0
typeMM:aBucks               0.008 9.698    -19.011    0.008     19.027   NA   0
typeMS:aBucks               0.015 9.698    -19.003    0.015     19.034   NA   0
typeMW:aBucks              -0.018 9.698    -19.037   -0.018     19.001   NA   0
typeSM:aBucks               0.001 9.698    -19.018    0.001     19.020   NA   0
typeSS:aBucks              -0.016 9.698    -19.035   -0.016     19.003   NA   0
typeSW:aBucks               0.010 9.698    -19.008    0.010     19.029   NA   0
typeWM:aBucks              -0.012 9.698    -19.031   -0.012     19.006   NA   0
typeWS:aBucks               0.024 9.698    -18.995    0.024     19.043   NA   0
typeWW:aBucks              -0.013 9.698    -19.032   -0.013     19.005   NA   0
typeMM:aBulls               0.010 9.698    -19.009    0.010     19.029   NA   0
typeMS:aBulls               0.017 9.698    -19.002    0.017     19.036   NA   0
typeMW:aBulls              -0.030 9.698    -19.049   -0.030     18.989   NA   0
typeSM:aBulls              -0.003 9.698    -19.022   -0.003     19.016   NA   0
typeSS:aBulls              -0.021 9.698    -19.039   -0.021     18.998   NA   0
typeSW:aBulls              -0.013 9.698    -19.032   -0.013     19.006   NA   0
typeWM:aBulls               0.001 9.698    -19.018    0.001     19.019   NA   0
typeWS:aBulls               0.027 9.698    -18.992    0.027     19.046   NA   0
typeWW:aBulls              -0.011 9.698    -19.030   -0.011     19.008   NA   0
typeMM:aCavaliers           0.030 9.698    -18.988    0.030     19.049   NA   0
typeMS:aCavaliers           0.040 9.698    -18.979    0.040     19.059   NA   0
typeMW:aCavaliers          -0.007 9.698    -19.026   -0.007     19.012   NA   0
typeSM:aCavaliers          -0.005 9.698    -19.024   -0.005     19.014   NA   0
typeSS:aCavaliers          -0.015 9.698    -19.034   -0.015     19.004   NA   0
typeSW:aCavaliers          -0.001 9.698    -19.020   -0.001     19.018   NA   0
typeWM:aCavaliers          -0.019 9.698    -19.038   -0.019     19.000   NA   0
typeWS:aCavaliers           0.007 9.698    -19.012    0.007     19.025   NA   0
typeWW:aCavaliers          -0.042 9.698    -19.061   -0.042     18.977   NA   0
typeMM:aCeltics            -0.005 9.698    -19.024   -0.005     19.013   NA   0
typeMS:aCeltics             0.020 9.698    -18.999    0.020     19.039   NA   0
typeMW:aCeltics            -0.011 9.698    -19.030   -0.011     19.007   NA   0
typeSM:aCeltics             0.010 9.698    -19.008    0.010     19.029   NA   0
typeSS:aCeltics            -0.006 9.698    -19.025   -0.006     19.013   NA   0
typeSW:aCeltics            -0.008 9.698    -19.026   -0.008     19.011   NA   0
typeWM:aCeltics            -0.009 9.698    -19.028   -0.009     19.009   NA   0
typeWS:aCeltics             0.012 9.698    -19.007    0.012     19.031   NA   0
typeWW:aCeltics            -0.017 9.698    -19.036   -0.017     19.002   NA   0
typeMM:aClippers            0.008 9.698    -19.010    0.008     19.027   NA   0
typeMS:aClippers            0.023 9.698    -18.996    0.023     19.042   NA   0
typeMW:aClippers            0.007 9.698    -19.012    0.007     19.026   NA   0
typeSM:aClippers            0.004 9.698    -19.015    0.004     19.023   NA   0
typeSS:aClippers            0.003 9.698    -19.016    0.003     19.021   NA   0
```

```
typeSW:aClippers          -0.012  9.698   -19.031   -0.012    19.007   NA   0
typeWM:aClippers          -0.028  9.698   -19.047   -0.028    18.991   NA   0
typeWS:aClippers           0.023  9.698   -18.996    0.023    19.042   NA   0
typeWW:aClippers          -0.021  9.698   -19.040   -0.021    18.998   NA   0
typeMM:aGrizzlies         -0.002  9.698   -19.020   -0.002    19.017   NA   0
typeMS:aGrizzlies          0.017  9.698   -19.002    0.017    19.036   NA   0
typeMW:aGrizzlies         -0.033  9.698   -19.052   -0.033    18.986   NA   0
typeSM:aGrizzlies          0.000  9.698   -19.019    0.000    19.019   NA   0
typeSS:aGrizzlies         -0.019  9.698   -19.038   -0.019    19.000   NA   0
typeSW:aGrizzlies          0.007  9.698   -19.012    0.007    19.026   NA   0
typeWM:aGrizzlies         -0.006  9.698   -19.025   -0.006    19.013   NA   0
typeWS:aGrizzlies          0.010  9.698   -19.008    0.010    19.029   NA   0
typeWW:aGrizzlies         -0.004  9.698   -19.023   -0.004    19.014   NA   0
typeMM:aHawks              0.001  9.698   -19.018    0.001    19.020   NA   0
typeMS:aHawks              0.022  9.698   -18.997    0.022    19.041   NA   0
typeMW:aHawks             -0.045  9.698   -19.064   -0.045    18.974   NA   0
typeSM:aHawks             -0.019  9.698   -19.038   -0.019    18.999   NA   0
typeSS:aHawks             -0.033  9.698   -19.052   -0.033    18.986   NA   0
typeSW:aHawks             -0.019  9.698   -19.038   -0.019    19.000   NA   0
typeWM:aHawks              0.026  9.698   -18.993    0.026    19.045   NA   0
typeWS:aHawks              0.045  9.698   -18.974    0.045    19.064   NA   0
typeWW:aHawks              0.013  9.698   -19.005    0.013    19.032   NA   0
typeMM:aHeat              -0.001  9.698   -19.020   -0.001    19.018   NA   0
typeMS:aHeat               0.043  9.698   -18.975    0.043    19.062   NA   0
typeMW:aHeat              -0.008  9.698   -19.027   -0.008    19.011   NA   0
typeSM:aHeat              -0.014  9.698   -19.033   -0.014    19.004   NA   0
typeSS:aHeat              -0.039  9.698   -19.057   -0.039    18.980   NA   0
typeSW:aHeat              -0.032  9.698   -19.051   -0.032    18.986   NA   0
typeWM:aHeat              -0.032  9.698   -19.051   -0.032    18.986   NA   0
typeWS:aHeat               0.058  9.698   -18.961    0.058    19.077   NA   0
typeWW:aHeat               0.020  9.698   -18.999    0.020    19.039   NA   0
typeMM:aHornets           -0.017  9.698   -19.036   -0.017    19.002   NA   0
typeMS:aHornets            0.003  9.698   -19.016    0.003    19.021   NA   0
typeMW:aHornets           -0.028  9.698   -19.047   -0.028    18.991   NA   0
typeSM:aHornets           -0.008  9.698   -19.027   -0.008    19.011   NA   0
typeSS:aHornets           -0.017  9.698   -19.036   -0.017    19.001   NA   0
typeSW:aHornets           -0.006  9.698   -19.025   -0.006    19.012   NA   0
typeWM:aHornets            0.007  9.698   -19.012    0.007    19.026   NA   0
typeWS:aHornets            0.024  9.698   -18.995    0.024    19.043   NA   0
typeWW:aHornets            0.002  9.698   -19.017    0.002    19.021   NA   0
typeMM:aJazz               0.008  9.698   -19.010    0.008    19.027   NA   0
typeMS:aJazz               0.027  9.698   -18.992    0.027    19.046   NA   0
typeMW:aJazz              -0.008  9.698   -19.027   -0.008    19.011   NA   0
typeSM:aJazz               0.003  9.698   -19.016    0.003    19.022   NA   0
typeSS:aJazz              -0.007  9.698   -19.026   -0.007    19.012   NA   0
typeSW:aJazz              -0.012  9.698   -19.031   -0.012    19.007   NA   0
typeWM:aJazz              -0.014  9.698   -19.033   -0.014    19.005   NA   0
typeWS:aJazz               0.013  9.698   -19.006    0.013    19.032   NA   0
typeWW:aJazz              -0.017  9.698   -19.036   -0.017    19.001   NA   0
typeMM:aKings             -0.018  9.698   -19.037   -0.018    19.001   NA   0
typeMS:aKings              0.007  9.698   -19.012    0.007    19.026   NA   0
typeMW:aKings             -0.029  9.698   -19.048   -0.029    18.990   NA   0
typeSM:aKings              0.066  9.698   -18.953    0.066    19.085   NA   0
typeSS:aKings             -0.009  9.698   -19.028   -0.009    19.010   NA   0
typeSW:aKings              0.003  9.698   -19.016    0.003    19.022   NA   0
typeWM:aKings             -0.008  9.698   -19.027   -0.008    19.010   NA   0
typeWS:aKings              0.029  9.698   -18.989    0.029    19.048   NA   0
typeWW:aKings             -0.017  9.698   -19.036   -0.017    19.002   NA   0
typeMM:aKnicks            -0.019  9.698   -19.038   -0.019    18.999   NA   0
typeMS:aKnicks             0.000  9.698   -19.019    0.000    19.019   NA   0
typeMW:aKnicks            -0.040  9.698   -19.059   -0.040    18.978   NA   0
typeSM:aKnicks             0.000  9.698   -19.019    0.000    19.019   NA   0
typeSS:aKnicks            -0.024  9.698   -19.043   -0.024    18.995   NA   0
typeSW:aKnicks            -0.001  9.698   -19.020   -0.001    19.018   NA   0
typeWM:aKnicks             0.004  9.698   -19.015    0.004    19.023   NA   0
typeWS:aKnicks             0.042  9.698   -18.977    0.042    19.061   NA   0
typeWW:aKnicks            -0.002  9.698   -19.021   -0.002    19.017   NA   0
typeMM:aLakers             0.021  9.698   -18.998    0.021    19.040   NA   0
typeMS:aLakers             0.034  9.698   -18.985    0.034    19.053   NA   0
typeMW:aLakers             0.005  9.698   -19.014    0.005    19.024   NA   0
typeSM:aLakers             0.001  9.698   -19.018    0.001    19.020   NA   0
typeSS:aLakers            -0.023  9.698   -19.042   -0.023    18.996   NA   0
typeSW:aLakers            -0.011  9.698   -19.030   -0.011    19.008   NA   0
typeWM:aLakers            -0.012  9.698   -19.031   -0.012    19.007   NA   0
typeWS:aLakers             0.019  9.698   -19.000    0.019    19.038   NA   0
typeWW:aLakers            -0.022  9.698   -19.041   -0.022    18.997   NA   0
typeMM:aMagic              0.001  9.698   -19.018    0.001    19.020   NA   0
typeMS:aMagic              0.013  9.698   -19.006    0.013    19.031   NA   0
typeMW:aMagic             -0.023  9.698   -19.042   -0.023    18.996   NA   0
typeSM:aMagic             -0.008  9.698   -19.027   -0.008    19.011   NA   0
typeSS:aMagic             -0.045  9.698   -19.064   -0.045    18.974   NA   0
typeSW:aMagic             -0.010  9.698   -19.029   -0.010    19.008   NA   0
typeWM:aMagic              0.000  9.698   -19.019    0.000    19.019   NA   0
typeWS:aMagic              0.046  9.698   -18.973    0.046    19.065   NA   0
typeWW:aMagic              0.001  9.698   -19.018    0.001    19.019   NA   0
typeMM:aMavericks         -0.007  9.698   -19.026   -0.007    19.012   NA   0
 [ reached getOption("max.print") -- omitted 395 rows ]

Random effects:
  Name      Model
    a.season Generic0 model
   d.season Generic0 model
   inla.group(travel) RW1 model

Model hyperparameters:
                                    mean       sd 0.025quant 0.5quant 0.975quant mode
Precision for a.season            1305.20   118.33    1087.44  1299.95    1553.29   NA
Precision for d.season            1122.00    96.64     943.73  1117.85    1324.22   NA
Precision for inla.group(travel) 60189.73 29507.40   21350.15 54142.42  134672.11   NA

Deviance Information Criterion (DIC) ...............: 355818.20
Deviance Information Criterion (DIC, saturated) ....: 786553.04
Effective number of parameters .....................: 1207.41
```

```
Marginal log-Likelihood:  -181997.62
CPO, PIT is computed
Posterior summaries for the linear predictor and the fitted values are computed
(Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')


Logistic:
formula = y.binary~~1+h
          +scale(PTS_cur_season)
          +scale(WINRATE_cur_season)
          +scale(PTS_LOST_cur_season)
          +log(salary)
          + f(a.season, model = "generic0", Cmatrix = Q.a, rankdef = 1, constr = TRUE)
          + f(d.season, model = "generic0", Cmatrix = Q.d, rankdef = 1, constr = TRUE)
          + f(inla.group(travel), model = "rw1", scale.model = TRUE)
          + type:a + type:d
Call:
   c("inla.core(formula = formula, family = family, contrasts = contrasts, ", " data = data, quantiles
        =
   quantiles, E = E, offset = offset, ", " scale = scale, weights = weights, Ntrials = Ntrials, strata
        =
   strata, ", " lp.scale = lp.scale, link.covariates = link.covariates, verbose = verbose, ", " lincomb
        =
   lincomb, selection = selection, control.compute = control.compute, ", " control.predictor =
   control.predictor, control.family = control.family, ", " control.inla = control.inla, control.fixed
        =
   control.fixed, ", " control.mode = control.mode, control.expert = control.expert, ", " control.
        hazard
   = control.hazard, control.lincomb = control.lincomb, ", " control.update = control.update,
   control.lp.scale = control.lp.scale, ", " control.pardiso = control.pardiso, only.hyperparam =
   only.hyperparam, ", " inla.call = inla.call, inla.arg = inla.arg, num.threads = num.threads, ", "
   blas.num.threads = blas.num.threads, keep = keep, working.directory = working.directory, ", " silent
        =
   silent, inla.mode = inla.mode, safe = FALSE, debug = debug, ", " .parent.frame = .parent.frame)")
Time used:
    Pre = 10.7, Running = 99.1, Post = 1.85, Total = 112
Fixed effects:
                               mean    sd 0.025quant 0.5quant 0.975quant mode kld
(Intercept)                 -14.379 4.393    -22.999  -14.378     -5.766   NA   0
h                             0.852 0.024      0.806    0.852      0.900   NA   0
scale(PTS_cur_season)         0.615 0.081      0.456    0.615      0.774   NA   0
scale(WINRATE_cur_season)    -0.131 0.031     -0.192   -0.131     -0.070   NA   0
scale(PTS_LOST_cur_season)   -0.576 0.073     -0.719   -0.577     -0.434   NA   0
log(salary)                   0.760 0.188      0.392    0.760      1.130   NA   0
typeMM:a76ers                -0.048 9.703    -19.077   -0.048     18.982   NA   0
typeMS:a76ers                -0.487 9.707    -19.525   -0.487     18.551   NA   0
typeMW:a76ers                -0.017 9.704    -19.047   -0.017     19.013   NA   0
typeSM:a76ers                 0.392 9.708    -18.646    0.392     19.430   NA   0
typeSS:a76ers                -0.064 9.711    -19.108   -0.064     18.980   NA   0
typeSW:a76ers                -0.239 9.708    -19.277   -0.239     18.800   NA   0
typeWM:a76ers                -0.065 9.704    -19.095   -0.065     18.966   NA   0
typeWS:a76ers                 0.163 9.708    -18.875    0.163     19.202   NA   0
typeWW:a76ers                -0.025 9.705    -19.057   -0.025     19.007   NA   0
typeMM:aBucks                -0.253 9.703    -19.281   -0.253     18.776   NA   0
typeMS:aBucks                -0.860 9.707    -19.897   -0.860     18.177   NA   0
typeMW:aBucks                -0.177 9.703    -19.206   -0.177     18.852   NA   0
typeSM:aBucks                -0.367 9.706    -19.401   -0.367     18.668   NA   0
typeSS:aBucks                 0.129 9.710    -18.914    0.129     19.171   NA   0
typeSW:aBucks                 0.292 9.707    -18.745    0.292     19.329   NA   0
typeWM:aBucks                 0.043 9.703    -18.986    0.043     19.071   NA   0
typeWS:aBucks                 0.772 9.707    -18.264    0.772     19.809   NA   0
typeWW:aBucks                 0.239 9.704    -18.792    0.239     19.270   NA   0
typeMM:aBulls                 0.196 9.703    -18.834    0.196     19.225   NA   0
typeMS:aBulls                -0.252 9.707    -19.290   -0.252     18.785   NA   0
typeMW:aBulls                -0.034 9.703    -19.064   -0.034     18.996   NA   0
typeSM:aBulls                -0.403 9.703    -19.431   -0.403     18.625   NA   0
typeSS:aBulls                 0.334 9.707    -18.702    0.334     19.370   NA   0
typeSW:aBulls                -0.066 9.703    -19.095   -0.066     18.964   NA   0
typeWM:aBulls                 0.024 9.704    -19.006    0.024     19.055   NA   0
typeWS:aBulls                 0.412 9.707    -18.626    0.412     19.449   NA   0
typeWW:aBulls                -0.132 9.705    -19.164   -0.132     18.901   NA   0
typeMM:aCavaliers             0.393 9.705    -18.640    0.393     19.426   NA   0
typeMS:aCavaliers            -0.224 9.709    -19.265   -0.224     18.817   NA   0
typeMW:aCavaliers             0.308 9.706    -18.727    0.308     19.343   NA   0
typeSM:aCavaliers            -0.161 9.703    -19.189   -0.161     18.868   NA   0
typeSS:aCavaliers             0.256 9.706    -18.779    0.256     19.292   NA   0
typeSW:aCavaliers             0.414 9.703    -18.616    0.414     19.443   NA   0
typeWM:aCavaliers            -0.668 9.704    -19.699   -0.668     18.364   NA   0
typeWS:aCavaliers             0.184 9.708    -18.855    0.184     19.223   NA   0
typeWW:aCavaliers            -0.497 9.705    -19.530   -0.497     18.536   NA   0
typeMM:aCeltics              -0.455 9.704    -19.487   -0.455     18.577   NA   0
typeMS:aCeltics              -0.904 9.709    -19.944   -0.904     18.136   NA   0
typeMW:aCeltics               0.232 9.705    -18.801    0.232     19.265   NA   0
typeSM:aCeltics               0.026 9.702    -19.002    0.026     19.053   NA   0
typeSS:aCeltics               0.505 9.706    -18.530    0.505     19.540   NA   0
typeSW:aCeltics               0.429 9.703    -18.600    0.429     19.458   NA   0
typeWM:aCeltics              -0.432 9.704    -19.463   -0.432     18.600   NA   0
typeWS:aCeltics               0.441 9.708    -18.598    0.441     19.480   NA   0
typeWW:aCeltics               0.218 9.705    -18.816    0.218     19.252   NA   0
typeMM:aClippers             -0.030 9.704    -19.061   -0.030     19.000   NA   0
typeMS:aClippers             -0.608 9.707    -19.645   -0.608     18.430   NA   0
typeMW:aClippers              0.442 9.704    -18.589    0.442     19.474   NA   0
typeSM:aClippers             -0.093 9.703    -19.122   -0.093     18.935   NA   0
typeSS:aClippers              0.441 9.707    -18.595    0.441     19.477   NA   0
typeSW:aClippers              0.406 9.704    -18.624    0.406     19.436   NA   0
typeWM:aClippers             -0.239 9.704    -19.270   -0.239     18.792   NA   0
typeWS:aClippers              0.094 9.708    -18.945    0.094     19.133   NA   0
typeWW:aClippers             -0.175 9.705    -19.208   -0.175     18.858   NA   0
typeMM:aGrizzlies             0.130 9.702    -18.898    0.130     19.158   NA   0
typeMS:aGrizzlies            -0.429 9.707    -19.465   -0.429     18.607   NA   0
typeMW:aGrizzlies            -0.053 9.703    -19.081   -0.053     18.976   NA   0
typeSM:aGrizzlies            -0.283 9.703    -19.313   -0.283     18.747   NA   0
```

```
typeSS:aGrizzlies        -0.022  9.707   -19.059   -0.022    19.016   NA   0
typeSW:aGrizzlies         0.415  9.704   -18.617    0.415    19.446   NA   0
typeWM:aGrizzlies         0.082  9.704   -18.948    0.082    19.112   NA   0
typeWS:aGrizzlies         0.304  9.707   -18.733    0.304    19.341   NA   0
typeWW:aGrizzlies        -0.121  9.705   -19.153   -0.121    18.911   NA   0
typeMM:aHawks             0.087  9.705   -18.945    0.087    19.120   NA   0
typeMS:aHawks            -0.380  9.709   -19.420   -0.380    18.660   NA   0
typeMW:aHawks             0.000  9.705   -19.033    0.000    19.033   NA   0
typeSM:aHawks            -0.264  9.703   -19.293   -0.264    18.766   NA   0
typeSS:aHawks             0.228  9.707   -18.809    0.228    19.265   NA   0
typeSW:aHawks             0.135  9.704   -18.895    0.135    19.166   NA   0
typeWM:aHawks             0.084  9.704   -18.948    0.084    19.116   NA   0
typeWS:aHawks             0.396  9.708   -18.643    0.396    19.435   NA   0
typeWW:aHawks            -0.288  9.705   -19.322   -0.288    18.746   NA   0
typeMM:aHeat             -0.123  9.706   -19.158   -0.123    18.912   NA   0
typeMS:aHeat             -0.313  9.709   -19.354   -0.313    18.729   NA   0
typeMW:aHeat             -0.405  9.706   -19.440   -0.405    18.630   NA   0
typeSM:aHeat             -0.452  9.703   -19.481   -0.452    18.576   NA   0
typeSS:aHeat              0.086  9.706   -18.950    0.086    19.121   NA   0
typeSW:aHeat              0.153  9.703   -18.877    0.153    19.183   NA   0
typeWM:aHeat             -0.092  9.710   -19.134   -0.092    18.951   NA   0
typeWS:aHeat              1.015  9.714   -18.037    1.015    20.067   NA   0
typeWW:aHeat              0.697  9.714   -18.354    0.697    19.749   NA   0
typeMM:aHornets           0.002  9.704   -19.029    0.002    19.032   NA   0
typeMS:aHornets          -0.805  9.708   -19.844   -0.805    18.233   NA   0
typeMW:aHornets          -0.183  9.704   -19.213   -0.183    18.848   NA   0
typeSM:aHornets          -0.045  9.710   -19.087   -0.045    18.998   NA   0
typeSS:aHornets          -0.078  9.716   -19.132   -0.078    18.976   NA   0
typeSW:aHornets          -0.004  9.711   -19.049   -0.004    19.041   NA   0
typeWM:aHornets           0.080  9.702   -18.948    0.080    19.108   NA   0
typeWS:aHornets           0.351  9.706   -18.685    0.351    19.387   NA   0
typeWW:aHornets           0.134  9.703   -18.895    0.134    19.164   NA   0
typeMM:aJazz              0.043  9.702   -18.985    0.043    19.070   NA   0
typeMS:aJazz             -0.693  9.706   -19.728   -0.693    18.343   NA   0
typeMW:aJazz              0.048  9.703   -18.980    0.048    19.077   NA   0
typeSM:aJazz             -0.038  9.703   -19.068   -0.038    18.991   NA   0
typeSS:aJazz              0.365  9.707   -18.672    0.365    19.402   NA   0
typeSW:aJazz             -0.062  9.704   -19.093   -0.062    18.969   NA   0
typeWM:aJazz             -0.054  9.703   -19.083   -0.054    18.976   NA   0
typeWS:aJazz              0.483  9.707   -18.554    0.483    19.521   NA   0
typeWW:aJazz              0.303  9.705   -18.729    0.303    19.336   NA   0
typeMM:aKings            -0.040  9.705   -19.073   -0.040    18.993   NA   0
typeMS:aKings            -1.074  9.709   -20.115   -1.074    17.967   NA   0
typeMW:aKings            -0.180  9.705   -19.213   -0.180    18.854   NA   0
typeSM:aKings             0.785  9.716   -18.269    0.785    19.840   NA   0
typeSS:aKings             0.314  9.719   -18.746    0.314    19.373   NA   0
typeSW:aKings             0.088  9.714   -18.963    0.088    19.138   NA   0
typeWM:aKings            -0.266  9.704   -19.298   -0.266    18.765   NA   0
typeWS:aKings             0.200  9.708   -18.839    0.200    19.239   NA   0
typeWW:aKings            -0.060  9.705   -19.093   -0.060    18.973   NA   0
typeMM:aKnicks            0.114  9.704   -18.918    0.114    19.145   NA   0
typeMS:aKnicks           -0.778  9.708   -19.817   -0.778    18.261   NA   0
typeMW:aKnicks           -0.121  9.704   -19.152   -0.121    18.910   NA   0
typeSM:aKnicks           -0.275  9.710   -19.317   -0.275    18.767   NA   0
typeSS:aKnicks            0.388  9.716   -18.667    0.388    19.442   NA   0
typeSW:aKnicks           -0.147  9.712   -19.193   -0.147    18.899   NA   0
typeWM:aKnicks           -0.077  9.703   -19.107   -0.077    18.953   NA   0
typeWS:aKnicks            0.305  9.707   -18.732    0.305    19.342   NA   0
typeWW:aKnicks           -0.253  9.704   -19.284   -0.253    18.778   NA   0
typeMM:aLakers            0.121  9.704   -18.909    0.121    19.151   NA   0
typeMS:aLakers           -0.511  9.707   -19.549   -0.511    18.526   NA   0
typeMW:aLakers            0.069  9.704   -18.962    0.069    19.100   NA   0
typeSM:aLakers           -0.163  9.703   -19.192   -0.163    18.866   NA   0
typeSS:aLakers            0.264  9.707   -18.773    0.264    19.301   NA   0
typeSW:aLakers            0.443  9.704   -18.588    0.443    19.474   NA   0
typeWM:aLakers           -0.370  9.703   -19.400   -0.370    18.660   NA   0
typeWS:aLakers            0.377  9.707   -18.660    0.377    19.414   NA   0
typeWW:aLakers           -0.172  9.704   -19.204   -0.172    18.859   NA   0
typeMM:aMagic            -0.235  9.703   -19.264   -0.235    18.795   NA   0
typeMS:aMagic            -1.140  9.707   -20.178   -1.140    17.897   NA   0
typeMW:aMagic            -0.326  9.704   -19.357   -0.326    18.704   NA   0
typeSM:aMagic             0.437  9.706   -18.597    0.437    19.471   NA   0
typeSS:aMagic             0.562  9.709   -18.479    0.562    19.604   NA   0
typeSW:aMagic             0.956  9.707   -18.081    0.956    19.994   NA   0
typeWM:aMagic            -0.324  9.704   -19.355   -0.324    18.707   NA   0
typeWS:aMagic             0.306  9.708   -18.732    0.306    19.344   NA   0
typeWW:aMagic            -0.408  9.705   -19.441   -0.408    18.625   NA   0
typeMM:aMavericks         0.086  9.703   -18.943    0.086    19.115   NA   0
 [ reached getOption("max.print") -- omitted 395 rows ]

Random effects:
  Name     Model
    a.season Generic0 model
   d.season Generic0 model
   inla.group(travel) RW1 model

Model hyperparameters:
                                     mean       sd 0.025quant 0.5quant 0.975quant mode
Precision for a.season               6.21 8.22e-01       4.74     6.16       7.97   NA
Precision for d.season               3.96 3.71e-01       3.29     3.94       4.75   NA
Precision for inla.group(travel) 20553.88 2.02e+04    1324.53 14518.90   73887.93   NA

Deviance Information Criterion (DIC) ...............: 56821.84
Deviance Information Criterion (DIC, saturated) ....: 56821.84
Effective number of parameters .....................: 1096.25

Marginal log-Likelihood:  -30853.34
CPO, PIT is computed
Posterior summaries for the linear predictor and the fitted values are computed
(Posterior marginals needs also 'control.compute=list(return.marginals.predictor=TRUE)')
```