NBA Season and Playoff Series Results Predictions

Alexander Stroud and Tianchi Liu

March 20, 2018

Motivation: NBA Team Analytics

As the 2018 NBA postseason draws closer and top NBA teams start to clinch playoff spots, interests in forecasting which team will find the most success in the postseason have been elevating. Once the regular season ends and playoff matchups are finalized, direct team-to-team comparison and projection of playoff series outcomes will be interesting to various groups of people, such as sports analysts, bettors, and basketball fans at large. While fans will always enjoy subjective arguments regarding the merits of one team compared to another, there is also a place for statistical analysis in this discussion. It is in this place where nbamodelR is most useful.

The nbamodelR package gives users the ability to perform their own analyses of NBA teams. It does so by providing easy access to various types of NBA data, including game logs containing records of each individual game played, and season summary statistics for each team, averaged per game, per 100 possessions, or totaled up. Often, easy access to data is the largest barrier to performing a statistical analysis, so by providing an easy way to gather data, we are hopefully opening the door for future analysts. The data that can be scraped using our package comes from three sources: Basketball-Reference.com, stats.nba.com, and FiveThirtyEight.com, so even if one of those sites changes their data format or their data access policy, nbamodelR will still have data-collecting functionality.

Users can take data extracted with nbamodelR and use it to perform their own analyses. In case they are unsure where to start, however, nbamodelR provides functions to model NBA team skill, using FiveThirtyEight's Elo rating system, or a Bradley-Terry model. Once these models are built, nbamodelR makes team comparisons easy, as it includes a function to forecast playoff series outcomes given the results from its modeling functions. Additionally, it provides a function to estimate team win totals over a full 82-game season, to give an idea of the "true talent" of each team.

The package's only minor deviation from its proposal is that users will not have the flexibility of choosing which features to emphasize while modeling matchup results. This is mainly due to the fact that both the Elo rating system and Bradley-Terry model have been successful models for different kinds of sports analytics, and these models do not depend on a wide array of features. Additionally, R by default allows for easy creation of basic linear models with user-specified features, and we thought it would be less useful to write a function that was simply a glorified wrapper for stats::lm.

Evaluating Teams: Bradley-Terry

Scraping data

For users who are interested in inspecting individual game records, **getGameLogs** can be used to extract team statistics in each game from stats.NBA.com. The range of seasons covered can be specified with "from" and "to" arguments, and the argument "type" specifies whether playoff or regular season records are scraped from the website. Note that default value of the argument "to" is the same as "from", so that single season statistics are extracted when the argument "to" is not specified. The argument "simple" is FALSE by

default so that all types of game statistics for both home and away teams are collected. If the argument "simple" is set to TRUE, then the extracted dataset will only contain the following variables: Game_ID, GAME_DATE, H.TEAM, A.TEAM, H.PTS, A.PTS. H stands for the home team, and A for the away team. Two of the options for collecting game logs are presented below:

```
gamelogs2017_full = getGameLogs(from = 2017, type = "Regular Season")
gamelogs2016to2017_simple = getGameLogs(from = 2016, to = 2017, type = "Regular Season",
simple=TRUE)
```

Building a Model

Once these game logs are collected, we turn to predicting game results. The Bradley-Terry model is implemented here for estimating point differential and winning probabilities in a matchup. It predicts the point differential S_i in a matchup between home team H_i and away team A_i as:

$$S_i = \alpha + \beta_{H_i} - \beta_{A_i} + \epsilon_i$$

where α is a constant accounting for home court advantage, β_{H_i} and β_{A_i} are estimated "strengths"" of home and away teams, and ϵ_i is random noise in the model that follows:

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

Team strength and coefficient of home court advantage can be estimated using matrix method in which $\beta = \begin{bmatrix} \alpha \\ \beta_1 \\ \beta_2 \\ \dots \\ \beta_p \end{bmatrix}$, $Y = \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ \dots \\ S_n \end{bmatrix}$ and $X = \begin{bmatrix} 1 & X_{12} & X_{13} & \dots & X_{1p} \\ 1 & X_{22} & X_{23} & \dots & X_{2p} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & X_{n2} & X_{n3} & \dots & X_{np} \end{bmatrix}$, where $X_{ij} = 1$ if the i-th home team is Team j-1, $X_{ij} = -1$ if the i-th away team is Team j-1, and 0 otherwise. With a regularization term added to the model, the vector β can be determined using $\widehat{\beta_\lambda} = (X^T X + \lambda I)^{-1} X^T Y$ for some regularization coefficient $\lambda > 0$. Finally, estimated point differential in a matchup between certain home and away teams is $\widehat{S} = \widehat{\alpha} + \widehat{\beta_H} - \widehat{\beta_A}$

The above computation for point differential can be easily implemented using the **buildPtDiffMatrixBT** function in the package. It estimates team strength based on the inputted data frame of game logs, and then returns an estimated point differential matrix containing all possible matchups between teams included in the game logs. The regularization term is included by default, but can also be omitted. For example, the following command uses the regularized Bradley-Terry model and returns a point differential matrix with all possible matchups between NBA teams in 2017.

```
btPointDiffs = buildPtDiffMatrixBT(gamelogs2017_full)
```

The Bradley-Terry model can also be used to estimate winning probabilities in a matchup according to:

$$p_i = \frac{exp(\alpha + \beta_H - \beta_A)}{1 + exp(\alpha + \beta_H - \beta_A)}$$

It estimates the probability of the home team defeating the away team based on their "strengths" that are estimated using the method described above. In this package, **buildWinProbMatrixBT** takes a game log

dataset and returns a winning probability matrix in which each entry represents the probability of the team in that row defeating the team in that column.

```
btWinProb = buildWinProbMatrixBT(gamelogs2017_full)
```

For easier access to point differential matrix and winning probability matrix described above, one can use the **buildModelsBT** function, which takes a game log dataset and returns a list with both the winning probability (winProb) matrix and the point differential (pointDiff) matrix as the first and second elements of the list, respectively. The example below again uses **gamelogs2017_full**, but returns a list containing the results from previous two examples:

```
btList = buildModelsBT(gamelogs2017_full)
```

Evaluating Teams: Elo

Another commonly used technique for modeling sports game outcomes is the Elo rating system. It is developed by Hungarian-American physicist Arpad Elo and it was originally implemented as a chess rating system. The difference in Elo ratings between two competitors in a matchup can be used to predict game outcomes, and teams' Elo rating will either increase or decrease depending on the game result. A winning team will always take points from the losing team, and the amount of points transferred is determined by the difference in ratings between the two teams. If a high-rated team beats a low-rated team, only a few points will be transferred, whereas many points will be transferred if a low-rated team beats a high-rated team. Therefore, an Elo model is largely self-correcting, as teams incorrectly ranked too low will gain points rapidly, and teams incorrectly ranked too high will lose points just as quickly.

Due to its ability to quickly respond to changes in team skill, and its excellence in estimating relative skill levels of players in zero-sum games such as chess, it has been widely applied in different kind of sports, including basketball.

Gathering Data

FiveThirtyEight.com uses either pure Elo or Elo-based ratings systems to evaluate team skill in a wide array of sports, including the NBA. Fortunately for us, they maintain a publically available, continuously updated dataset of every game in NBA and ABA history, along with the pre- and post-game Elo ratings of the teams in each game. The function **getEloGameLogs** extracts this history of Elo ratings, potentially for the entire history of the NBA and ABA, if desired. These Elo-based game logs for a specific range of seasons or for one particular year can also be extracted by specifying the "from" and "to" aguments in the function. Such game logs can then be used for further modeling using the Elo rating system.

```
eloLogsAll = getEloGameLogs()
eloLogs2017 = getEloGameLogs(2017)
eloLogsFrom2000 = getEloGameLogs(2000, 2017)
```

Building a Model

Once these Elo-based game logs are extracted using **getEloGameLogs**, we turn to evaluating the individual teams. In Elo models, all that matters for predicting the outcome of a matchup is the difference in Elo rating of the two participants. However, to calculate this difference, we must first know each team's Elo rating.

Fortunately, nbamodelR supplies the function **extractTeamElos**, which takes a data frame with Elo game logs and returns Elo ratings of all teams at the end of a specified season. The following example extracts Elo ratings of all teams in 2017:

```
elos2017 = extractTeamElos(eloLogsAll, 2017)
head(elos2017)
```

Once these ratings are obtained, users will have an estimate of teams' strengths relative to one another. However, they can also use nbamodelR to build Elo-based winning probability and point differential matrices that are structured in the same way as those estimated by the Bradley-Terry model. For building the winning probability matrix, each home team's chance of winning visiting team can be modeled using logistic regression, in which the binary variable game result (+1 for home team wins, 0 for home team losses) is regressed onto the difference in elo ratings of the two teams (always home elo - away elo). The point differential can be modeled by regressing the score difference (home score - away score) onto the difference in elo ratings of the two teams.

Winning probability and point differential matrices of a particular season can be built based on Elo-based game logs using **buildWinProbMatrixElo** and **buildPtDiffMatrixElo**, respectively. As with the Bradley-Terry matrices, each entry represents the probability of the team in that row at home defeating the team in that column, or the point differential in a matchup between the team in that row at home against the team in that column. An example for computing these matrices is shown below:

```
eloWinProb = buildWinProbMatrixElo(eloLogsAll, 2017)
eloPointDiffs = buildPtDiffMatrixElo(eloLogsAll, 2017)
```

For easier access to both matrices, one can use the function **buildModelsElo** which also takes an eloLogs dataset and returns both matrices for a specified season as a list. The list's first element is winning probability matrix and the second element is point differential matrix, as with the Bradley-Terry model.

```
eloList = buildModelsElo(eloLogsAll, 2017)
```

Predicting Results with Bradley-Terry and Elo

Once we have our model outputs, we turn to predicting matchups between teams.

Predicting Full Season Win Totals

Based on the winning probability matrix of one particular season, the total number of wins for each team can be projected using the **projectSeasonWins** function. This function takes a winning probability matrix that is generated either by the Bradley-Terry or the Elo model, and returns projected wins for each team in that season. The input winning probability matrix can be validated by checking if total wins for all teams fall into a reasonable range, roughly 20-65 wins per season, barring exceptional cases. Examples are the following:

```
projectSeasonWins(btList$WinProb)
projectSeasonWins(eloList$WinProb)
```

Predicting playoff series outcomes

With the availability of winning probability matrix, one can estimate the probability of each outcome of a matchup between two teams in a 7-game playoff series. This can be done using the **playoffSeries** function, which takes certain home and away teams and the winning probability matrix described above. Note that the "home team" specified here refers to the team having 4 home games in the 7-game series. The function returns a vector giving the probability of each playoff series outcome. An optional bar plot of playoff series outcomes ranging from home team wins in 4 games to away team wins in 4 games can be generated if the "plot" argument is set to TRUE. An example is the following:

```
playoffSeries(prob = btList$WinProb, home = "CLE", away = "GSW", plot = T)
playoffSeries(prob = eloList$WinProb, home = "CLE", away = "GSW")
```

Contributions

All data-collecting code was written by astroud. The Bradley-Terry model code was written almost entirely by kitliu5. Elo model code was written by astroud. Playoff series prediction and plotting code was written mainly by kitliu5, with minor edits and tweaks by astroud. Season prediction code was written by astroud, with debugging assistance from kitliu5. Documentation for each function was written by the author of the function, with both astroud and kitliu5 checking all the documentation for consistency and correctness. This report was drafted by kitliu5, and revised by astroud.

Extensions and Future Additions

Season Summary Statistics

While all of the models referenced above utilize game-by-game data to make their predictions, data with statistics aggregated across an entire season are also potentially of use.

The function **getSeasonStats** allows users to extract NBA season data from Basketball-Reference.com. The argument "type" is used to specify the type of season data to be scraped: either data totaled for the entire season, averaged by game, or averaged by every 100 possessions. It also leaves users the freedom to cover any range of seasons by specifying the "from" and "to" arguments. As with **getGameLogs**, the default value of the argument "to" is the same as "from" so that single season statistics are extracted when the argument "to" is not specified. Examples are below:

```
seasonStats2017_per100p = getSeasonStats(from = 2017, type = "per 100 poss")
seasonStats2015to2017_totals = getSeasonStats(from = 2015, to = 2017, type = "totals")
```

With this data, users are free to create their own models. While nbamodelR does not provide any functions for building models from season summary data, we encourage the exploration of which attributes are most likely to lead to high win totals, as well as which type of aggregated statistics are most informative.

Creating an Elo Difference Matrix

The 2017 Elo difference matrix can be built based on Elo ratings of all teams in 2017 extracted above. Each entry in this matrix represents the difference in Elo ratings between the team in that row and the team in that column. This can be done using the function **buildEloDiffsMatrix**. However, because the winning probability and point differential matrices are more directly related to predicting matchups, and the elo difference matrix

is more of an intermediate result, we chose to have the function **buildEloDiffsMatrix** be internal. Still, users might find it potentially of use, and we would consider exporting it in future versions of the package.

```
# eloDiffsMatrix = buildEloDiffsMatrix(elos2017)
```

Providing Naming Consistency

This section regards potential functionality that does not yet exist within this package, but could be useful, especially when dealing with multiple data sources. In certain cases, different websites use the same abbreviation to refer to multiple different teams, or multiple abbreviations for the same team. For instance, consider the NBA's history in Charlotte: the Charlotte Hornets existed from 1988 to 2002, before moving to New Orleans. Then, the city of Charlotte was given a new NBA team, which competed as the Bobcats from 2004 to 2014, before renaming to the Hornets once again in 2014. FiveThirtyEight's game logs refer to the 1988-2002 version of the team with the abbreviation "CHH", and the 2004-2018 iteration as "CHO". Note that depending on the year, "CHO" could refer to the Bobcats or the Hornets. Annoyingly, stats.nba.com uses "CHA" instead to refer to the Charlotte franchise from 2004 to 2018. The potential for user confusion here is large, but to be able to map abbreviations to names, we would need a comprehensive table with seasons, names, and abbreviations, which we did not have time to implement. Adding a function to convert a combination of season and abbreviation to the name of the team is one of our primary future goals.

Conclusion

That's all we have to share about our package! We hope that reading this was informative, and that you enjoy using nbamodelR.