Assessing Team Performance: Using Bayesian Markov Chain Monte Carlo to examine the predictive value of particular baseball statistics

Prosper Atukwatse

December 9, 2016

Abstract

Baseball, one of America's most popular sports, is well known for its extensive use of statistics. Some of the most prominent statistics are Batting Average (BA), On-Base Percentage (OBP), and Slugging Percentage (SLG). Traditionally, predictive analysts have focused on the players' BA, but with the publication of *Moneyball*, Michael Lewis' story of how Billy Beane's Oakland Athletics exploited OBP to assemble a winning team on the cheap, more attention has turned to the latter two statistics. In this paper, we analyze MLB team performance in the 2016 season using Bayesian Markov Chain Monte Carlo to examine the relative importance of BA in contrast to a combination of OBP and SLG in regards to the performance of a given team.

1. Introduction

Offense in baseball can be broadly divided into into skills: hitting and baserunning. While baserunning is important, its value relative to hitting is small, and thus a team's offense mainly hinges on the team's hitting. We wish to examine the relationship between team performance (number of runs scored) and particular baseball statistics (BA, OBP and SLG). We do this by fitting two models to our data. Our first model is a simple normal linear regression model that features the number of runs scored by a given team, and its batting average.The second model is a multiple normal linear regression model that features the number of runs scored, and the team's on-base percentage and slugging percentage. The specific question we wish to answer is: Which statistic is a better predictor of the number of runs that a team is likely to score in a season? This question will be answered by performing a Bayesian MCMC analysis of both models using MLB team performance data in the 2016 season.

2. Markov Chain Monte Carlo

The Markov Chain Monte Carlo can be explained as follows:

Consider a general framework of random quantities x in a p-dimensional space and a target distribution $\Pi(x)$ with a p.d.f $\pi(x)$, discrete or continous. We cannot directly sample from $\pi(x)$ and therefore we focus on exploring $\pi(x)$ by randomly wandering around via some some form of sequential algorithm that aims to identify regions of higher probability and generate a catalogue of x values representing that probability. Imagine that at some time $t-1$ we are at $x = x^{(t-1)} \in \chi$. Looking around in a region near $x^{(t-1)}$ we might see points of higher density, and they would represent interesting directions in χ to move towards. The setup is as follows.

The target distribution has a p.d.f $\pi(x)$ on χ.

A realization of a first-order Markov process is generated, $x^{(1)}, x^{(2)}, ..., x^{(t)}, ...$ starting from some initial state $x^{(0)}$.

A proposal distribution with p.d.f $g(x|x')$ is defined for all x, x' $\in \chi$. g is symmetric such that $g(x|x') = g(x'|x)$.

At step $t-1$, the current state is $x^{(t-1)}$. A candidate state $x*$ is generated from the current proposal distribution,

$$x* \sim g(x * |x^{(t-1)})$$

.

The target density ratio

$$\pi(x*)/\pi(x^{(t-1)})$$

compares the candidate state to the current state.

The Metropolis chain moves to the candidate proposed if it has a higher density than the current state. Otherwise, it moves there with the probability defined by the acceptance function.This acceptance function is defined as:

$$a = min[1, \frac{\pi(x*)}{\pi(x^{(t-1)})}]$$

Hence the chain always moves to the proposed state at time t if the proposed state has a higher target density than the current state, and moves to a lower density in proportion to the density value itself. This leads to a random, Markov process that naturally explores the state space according to the probability defined by $\pi(x)$ and hance generates a sequence that, while dependant, eventually represents draws from $\pi(x)$.

The advantage of an MCMC is that the time needed to obtain acceptable convergence is typically much less than for rejection sampling, since the sampling effort is concentrated in areas of high likelihood or posterior density.

3. MLB Team Performance

The data used in this experiment was obtained from baseballreference.com and exported to Excel in a .csv format. For the year 2016, there were 30 teams in the MLB and their

respective performance metrics represent our dataset. We elect to fit two models to three baseball metrics each designed to measure slightly different skills;

1. Batting Average - This is the rate at which a player reaches base on batted balls only. It is given by;

$$\text{Batting Average} = \frac{\text{Hits}}{\text{At bats}}$$

2. On-Base percentage - This is the rate at which a player reaches base on all types of play. It is given by;

$$\text{On-Base Percentage} = \frac{\text{Hits + Walks + Hit-by-pitches}}{\text{Plate appearances}}$$

3. Slugging Percentage - This metric is a measure of how hard and how far a player hits the ball. It is given by;

$$\text{Slugging Percentage} = \frac{\text{Total bases}}{\text{At bats}}$$

So how are the numbers of runs scored by a team in a given season distributed? Since all the three metrics above reflect elements of the game that are directly related to the scoring of runs, the distribution of the number of runs scored should be well captured by a normal linear distribution.

For the batting average metric, we fitted the following simple normal linear regression model.

The noninformative priors are;

$$\beta_0 \sim Normal(0, \sigma = 10000)$$

$$\beta_{AB} \sim Normal(0, \sigma = 10000)$$

$$sigma \sim Uniform(0, 10000)$$

The likelihood on the data is given by;

$$mu_i = \beta_0 + \beta_{BA} BA_i$$

$$Runs_i \sim Normal(mu_i, sd = sigma)$$

where $Runs_i$ is the number of runs scored by team i, and BA_i is the batting average of team i.

For the On-Base Percentage and Slugging Percentage metrics, we fitted the following normal linear multiple regression model.

The noninformative priors are;

$$\beta_0 \sim Normal(0, \sigma = 10000)$$

$$\beta_{OBP} \sim Normal(0, \sigma = 10000)$$

$$\beta_{SLG} \sim Normal(0, \sigma = 10000)$$

$$sigma \sim Uniform(0, 10000)$$

The likelihood on the data is given by;

$$mu_i = \beta_0 + \beta_{OBP} OBP_i + \beta_{SLG} SLG_i$$

$$Runs_i \sim Normal(mu_i, sd = sigma)$$

where $Runs_i$ is the number of runs scored by team i, OBP_i is the on-base percentage of team i, and SLG_i is the slugging percentage of team i.

For each model, we ran 200,000 iterations of the MCMC with a burn-in of 40000. We implement both models using R interfaced with the Nimble package.

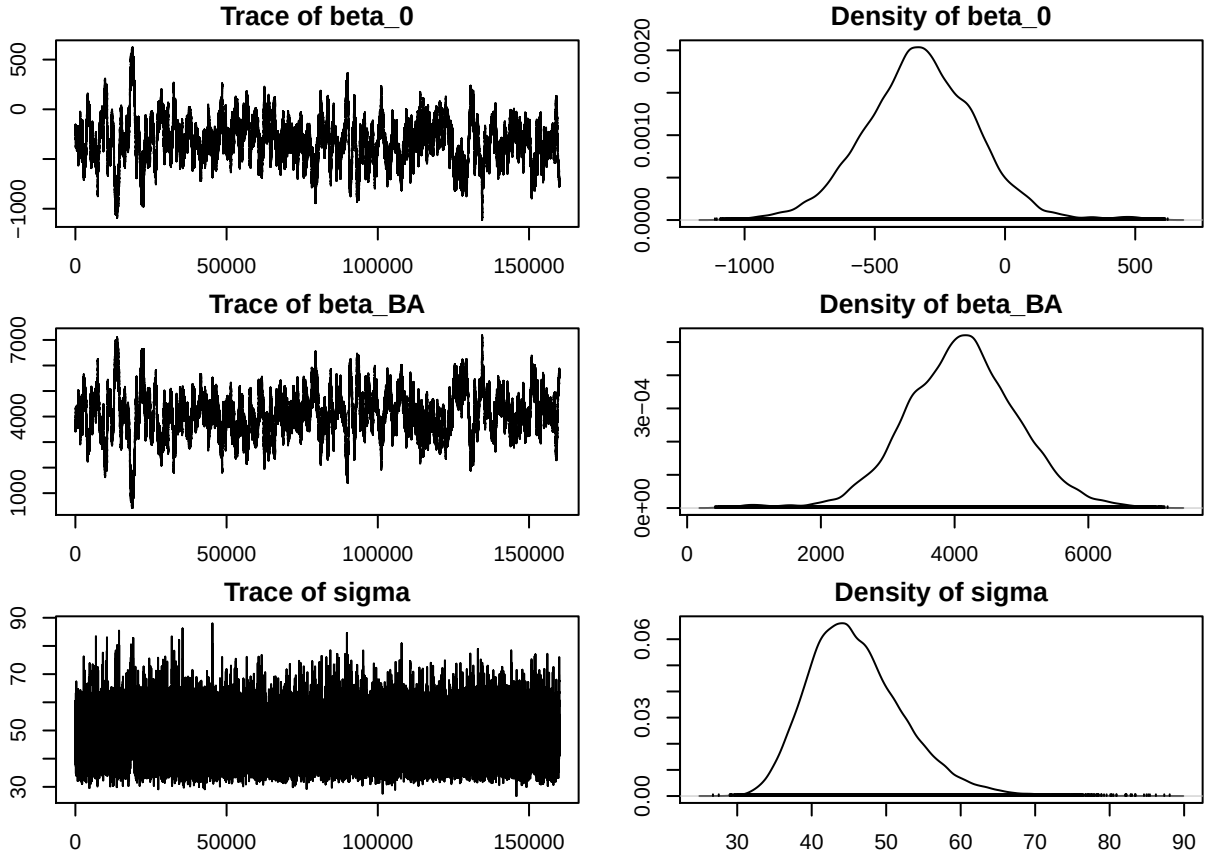
3. Results and Discussion

The R code used to implement the first model is shown below:

```
# first model (features with batting average)
set.seed(0)

code <- nimbleCode({
  # priors on top level parameters
  beta_0 ~ dnorm(0,sd=10000)
  beta_BA ~ dnorm(0,sd=10000)
  sigma ~ dunif(0,10000)

  # likelihood on the data
  for(i in 1:N){
    mu[i] <- beta_0 + beta_BA*x1[i]
```

```
    y[i] ~ dnorm(mu[i],sd=sigma)
  }
})

constants = list(N=30)
data = list(y = as.numeric(as.character(baseball_data$R)),
            x1 = as.numeric(as.character(baseball_data$BA)))
inits = list(beta_0 = 0,beta_BA = 1, sigma = 1)
```

The figures below show the traceplots and posterior density plots from a Markov Chain Monte Carlo using the data from 2016, with the first model fitted.

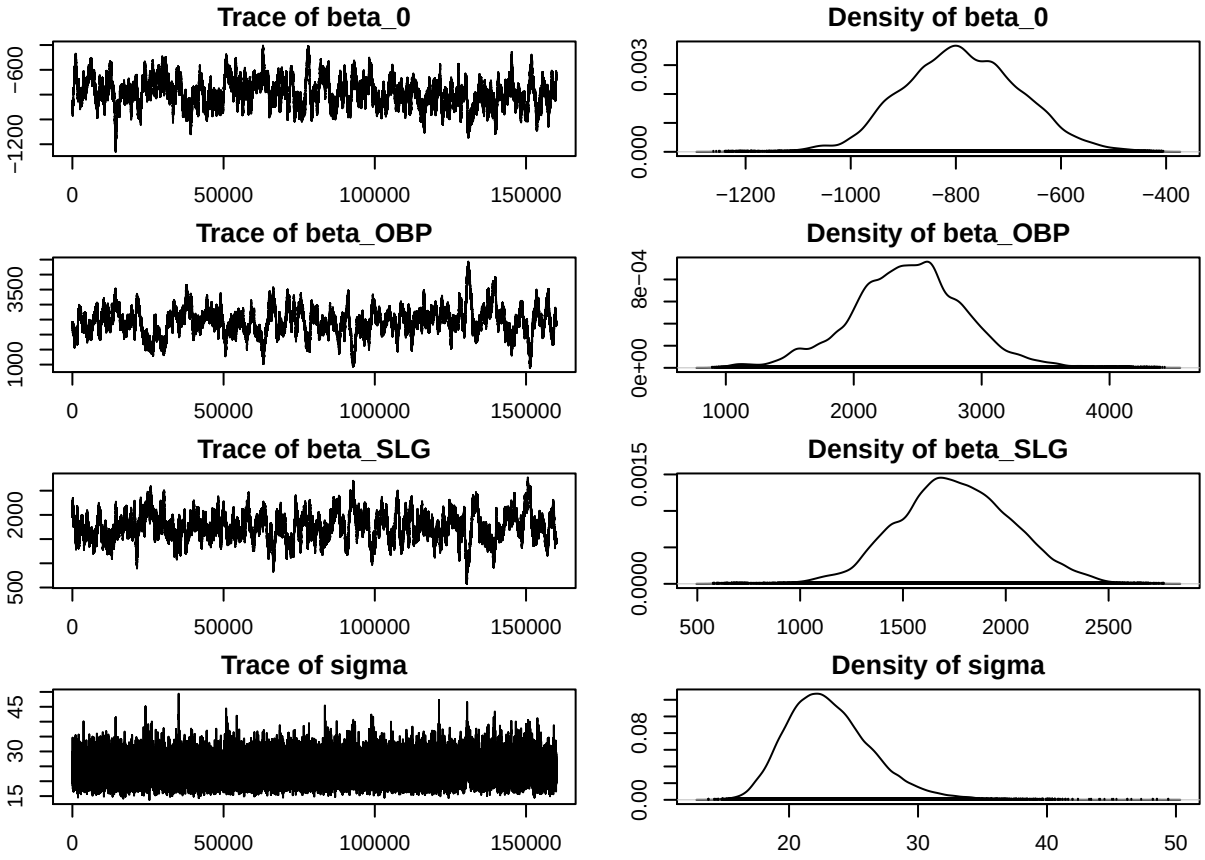


The R code for the second model is shown below:

```
# second model (features OBP and SLG)
set.seed(0)

code <- nimbleCode({
  # priors on top level parameters
  beta_0 ~ dnorm(0,sd=10000)
  beta_OBP ~ dnorm(0,sd=10000)
  beta_SLG ~ dnorm(0,sd=10000)
```

```
  sigma ~ dunif(0,10000)

  # likelihood on the data
  for(i in 1:N){
    mu[i] <- beta_0 + beta_OBP*x1[i] + beta_SLG*x2[i]
    y[i] ~ dnorm(mu[i],sd=sigma)
  }
})

constants = list(N=30)
data = list(y = as.numeric(as.character(baseball_data$R)),
            x1 = as.numeric(as.character(baseball_data$OBP)),
            x2 = as.numeric(as.character(baseball_data$SLG)))
inits = list(beta_0 = 0,beta_OBP = 1, beta_SLG = 1, sigma = 1)
```

The figures below show the traceplots and posterior density plots from a Markov Chain Monte Carlo using the data from 2016, with the second model fitted.



From the above plots, and the Gelman-Brooks-Rubin diagnostic test which returned values of 1, we can see that there is convergence of both models.

Batting Average

The analysis showed that for the first model, the average value of the intercept β_0 was -116. The average value of the fitted parameter β_{BA} was 514. Therefore, for the year 2016, the fitted model using batting average as a predictive metric for the number of runs scored is

$$Runs = -116 + 514 \times Batting\ Average + \epsilon$$

On-Base Percentage and Slugging Percentage

The analysis showed that for the second model, the average value of the intercept β_0 was -176. The average value of the parameter β_{OBP} was 273. The average value of β_{SLG} was 242. Therefore, for the year 2016, the fitted equation for the model that features both OBP and SLG is

$$Runs = -176 + 273 \times On\text{-}Base\ Percentage + 242 \times Slugging\ Percentage + \epsilon$$

The posterior predictive test using the mean values of the top-level parameters shows that the 95% credible interval is (706.3624, 738.8542) for the first model, and (715.3372, 732.5107) for the second model. Given a mean value of 724.8 runs for all the teams in the 2016 MLB season, we see that the model that features both OBP and SLG performs better than the one that features only BA.

4. Conclusions

We have considered the MCMC of two models. From the posterior predictive checks of these models using the mean values of the parameters, we observe that a model incorporating both OBP and SLG is a better predictor than one that only takes into account BA. This is consistent with previous findings that BA is a worse metric than OBP and SLG in predicting the number of runs that a team scores in a season. Based on these results, we would recommend the use of OBP and SLG instead of BA to predict the number of runs that a team is going to score in a season. Further analysis could involve testing these models separately on each of the six leagues that make up the MLB, so as to see if the leagues show different relationships between the number of runs scored and the various baseball statistics.

References

1. Baumer, B., and Zimbalist, A. (2014). *The Sabermetric Revolution.* University of Pennsylvania Press, Philadelphia.