Predicting the use of the sacrifice bunt in Major League Baseball

BUDT 714
May 10, 2007

Group 6
Charles Gallagher
Brian Gilbert
Neelay Mehta
Chao Rao

Executive Summary

Background

When a runner is on-base during a baseball game, the batter at the plate may be ordered by the manager to bunt, rather than swing away. A properly placed bunt can advance the runner a base, and although the bunter himself will get thrown out, it is a productive out as the runner is now in a position to score. This is called a sacrifice bunt, a technique often used in the course of a game. If the opposing team has insight into the specific situations when a sacrifice bunt will be executed, they will put themselves in a better position to defend it.

Goal

The goal of this project is to create a model for the St. Louis Cardinals that predicts the tendency of an opposing team to execute a sacrifice bunt in certain situations. While the occurrences of the sacrifice bunt in a given season are small when compared to the total number of plate occurrences that provide an opportunity to bunt, a successful defense of a sacrifice bunt or a sacrifice bunt executed in the right situation can mean a win for the manager ordering the bunt. It is often the case that teams advancing to the playoffs do so by a margin of one win over another team.

Data

To accomplish our goal, we were sponsored by Sig Mejdal, (the Senior Quantitative Analyst for the Cardinals) who gave us a dataset of 65,535 different events that occurred during the 2004 baseball season (random teams and games) with men on base. The dataset included the two teams, which of the teams was at home, which of the teams was on offense, the event that occurred, the inning, the number of outs before the event, the base-runners (and which bases) before the event, the score before the event, the fielding position of the batter, the batter's spot in the hitting order, and the batter's name.

Analysis

After an extensive analysis of the data which included adding variables from outside sources as well as creating new variables from the given dataset, we attempted to build classification trees which would provide us a scenario based tool which could be utilized in game situations in order to predict when a sacrifice bunt would be successful. Our end goal was to produce team specific trees that the manager could use as a predictive edge. Due to the extremely low occurrences of bunts in the data, creating a tree was not realistic as initially thought. Even when over sampling the data, the high predictive error rates compared to the Naïve Rule led us to investigate additional models/option.

Conclusion

We relied heavily on domain knowledge to help guide us through the selection process of the numerous variables given in the data-set. As our next goal, we attempted to use logistic regression to build an accurate model for the customer. In building a global model and several team specific models, many of our assumptions were validated. In looking at the team specific models, we were able to explain many of the significant variables such as why a given position on a certain team would be more likely to bunt. Ultimately, this helped us explain how and why much of the data acted as observed and in the end, led us to believe that a Manager's intuition has the potential to be even more powerful.

Technical Summary

The Question

The question we are trying to answer is predicting the tendency of an opposing baseball team to execute a sacrifice bunt in certain situations.

The Data

Our data set comes from Sig Mejdal, the Senior Quantitative Analyst for the St. Louis Cardinals. The original data set consisted of over 250,000 observations from the 2004 major league baseball season; in order to fit into Excel, the data was purged, giving us a random sampling of about 65,000 observations. Each observation had 20 variables describing the event. The original variables are described below:

- tyrgam – a unique numerical code identifying the game
- bat.tm – a unique numerical code identifying the team at bat
- bat.tm.nm – the popular name of the team at bat
- fd.tm – a unique numerical code identifying the fielding team
- fd.tm.nm – the popular name of the fielding team
- home.tm – a unique numerical code identifying the home team
- event – a numerical code describing the outcome of the at bat, such as single, double, etc.
- innin – the number of the inning in which the event took place
- out.before – the number of outs before the at bat
- out.after – the number of outs after the at bat
- basecode.before – a numerical code indicating the position of base runners before the at bat
- basecode.after – a numerical code indicating the position of base runners after the at bat
- score.bat.before – the number of runs scored by the batting team before the at bat
- score.field – the number of runs scored by the fielding team before the at bat
- score.bat.after – the number of runs scored by the batting team after the at bat
- bat.type – is a code binning the results from the event column into larger groups such as pop-up, grounder, line out, bunt, etc.
- fieldpos – a numerical code indicating the fielding position of the player at bat
- bpos – a number indicating the batter's order in the lineup
- last.name – the last name of the batter
- first.name – the first name of the batter

Our domain knowledge experts are Neelay Mehta and Brian Gilbert, two avid baseball fans. The granularity of the data is at the individual at bat level. The total number of observations in the data set is 65,535. Each of the 30 teams in Major League Baseball has between 1,962 and 2,395 observations. Bunting occurs 3.10% of the time in the overall data set (aka. The Naïve Rule). On average bunting occurs between 4.19% (with a runner on first) and 0.15% (with bases loaded) of the time, when looking at different scenarios with runners on base. On average batters bunt between 12.23%

(batter #9, a pitcher or the team's worst hitter) and 0.12% (batter #4, generally a power hitter) of the time.

Data cleaning

No information was missing in the data set. It was necessary to remove some variables such as out.after, basecode.after, and score.bat.after, because these variables are unknown at the time when we are trying to make our prediction. More variables were eliminated (tyrgam, fd.tm.nm and fieldpos) based on domain knowledge. Additional variables were eliminated based on redundancy, such as bat.tm, fd.tm, and event.

Data preprocessing

Next we derived some new variables by binning bat.type, innin, and home.tm. The variables were transformed into dummy variables as follows: bat.type becomes "Bunt:1, No Bunt: 0", innin becomes "Inning 1-9:1, >9:0", and home.tm becomes "Home:1, Away: 0". We also created dummy variables for each of the nine categories in bpos, and each of the seven scenarios in basecode.before. Next we created two new variables "AL: 0 NL: 1", a dummy variable indicating whether American League or National League rules were in effect, because different rules can encourage or discourage bunting. The other variable created was "ScoreDiff" (the score of the batting team minus the score of the fielding team), because in close games the batting team is more likely to bunt to get the winning or tying run. Also, we isolated ten "top bunters" from information on the 2004 season obtained from baseball-reference.com, so those players frequently called on to bunt are accounted for.

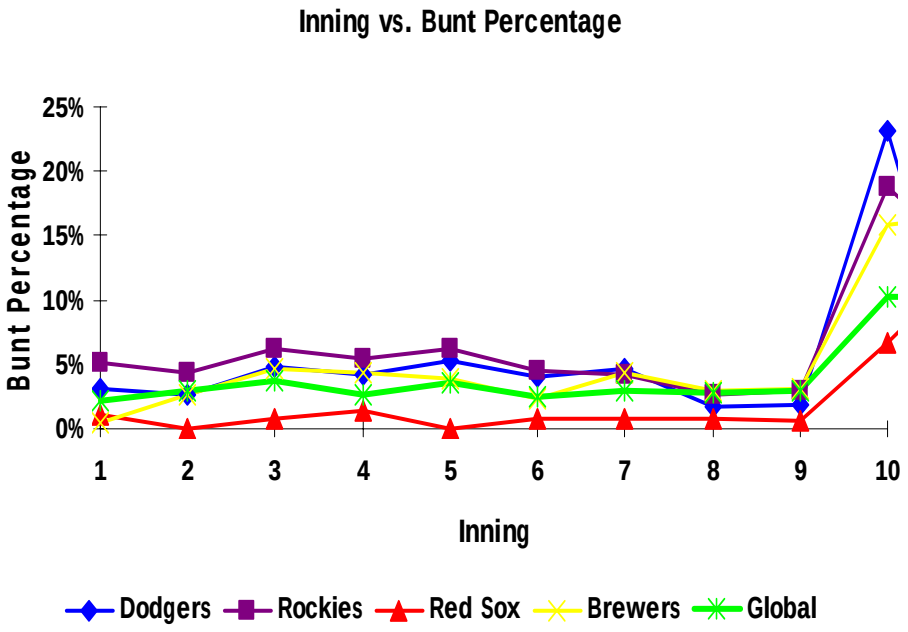
To visualize the data, we compared the global numbers to those of four specific teams: The Red Sox, Rockies, Brewers, and Dodgers. The Red Sox and Rockies were included because they were the most extreme in terms of bunt usage (Red Sox most rarely, Rockies most frequently). The Brewers and Dodgers had managers (Ned Yost and Jim Tracy) who currently manage intra-division rivals of the Cardinals (Yost still manages the Brewers, Tracy now manages the Rockies). The Cardinals play intra-division teams most frequently.

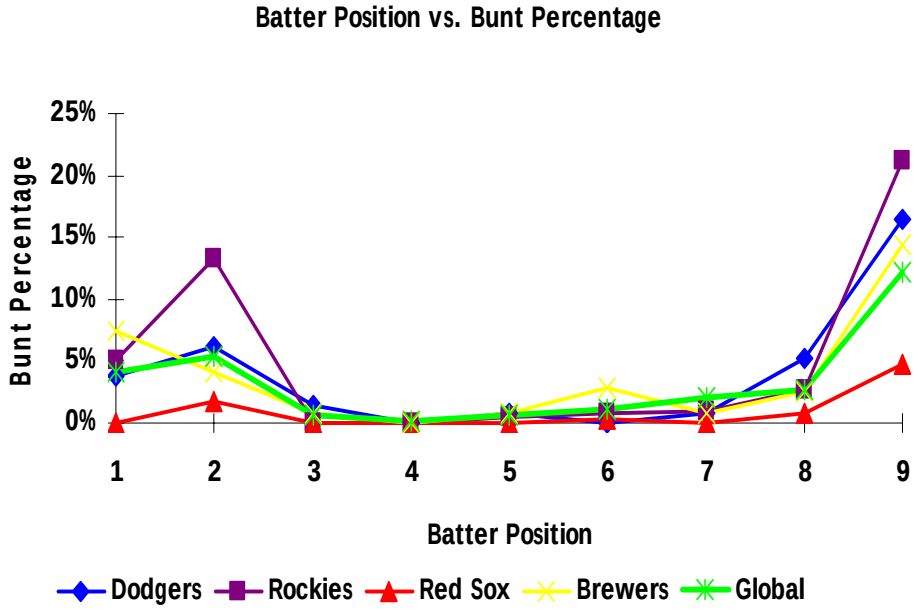
Analysis

Based on conversations with Sig Mejdal, we knew he was looking for a predictive model with a high level of transparency, specifically mentioning a decision tree. This made sense to us as well, given that the end user would be someone who is non-technical, such as a team manager. However, regardless of how many attempts we manipulated the data, classification trees were unsuccessful. The best classification tree was generated using over-sampling, and generated an error rate of 21.68%, which is much higher than the Naïve Rule of 3.10%.

Next we experimented with logistic regression, which would require a black box approach when actually implemented. We experimented with two versions of logistic regression models: (1) a global model, using all the data to construct one model and (2) separate models specific to each team who the Cardinals are playing against. While we preferred the Team-specific models as they were more parsimonious, their problems with predictive accuracy forced us to use the global models. Our global model used a best subset, with a cutoff of 0.5. It scored a validation rate of 3.00%, marginally better than the Naïve Rule of 3.10%.

Appendix 1: Visualizing the data





Appendix 2: The final regression model

The Regression Model

Input variables	Coefficient	Std. Error	p-value	Odds
Constant term	-2.91484118	0.19527394	0	*
Inning 1-9: 0 10+: 1	1.43896341	0.31070757	0.00000363	4.2163229
out.before	-2.06532502	0.12200621	0	0.12677707
basecode.before_3	-1.18709564	0.52609891	0.02404486	0.3051061
basecode.before_4	0.47677508	0.16713402	0.00433562	1.61087108
basecode.before_6	-2.63196373	1.02573335	0.01028984	0.07193705
basecode.before_7	-2.14123392	0.72531897	0.00315593	0.11750975
binned score diff_2	0.66862828	0.20002525	0.00082962	1.95155859
binned score diff_5+	-0.85927516	0.3422673	0.01205473	0.42346892
binned bpos_3-7	-1.65075326	0.1863011	0	0.1919053
binned bpos_9	1.85490274	0.15460882	0	6.39107656
Top Bunter	1.51956904	0.26843038	0.00000002	4.57025528
Team binned_2	0.90669906	0.20409924	0.00000889	2.47613549
Team binned_3	1.00761068	0.20899259	0.00000143	2.73904896
Team binned_4	1.47704506	0.21063162	0	4.3799839

Residual df	9985
Residual Dev.	1757.736206
% Success in training data	3.17
# Iterations used	9
Multiple R-squared	0.3749283

Validation Data scoring - Summary Report

Cut off Prob.Val. for Success (Updatable)	0.5

Classification Confusion Matrix		
	Predicted Class	
Actual Class	1	0
1	226	1317
0	184	48273

Error Report			
Class	# Cases	# Errors	% Error
1	1543	1317	85.35
0	48457	184	0.38
Overall	50000	1501	3.00

Elapsed Time

Overall (secs)	593.00