Saber-what? An Analysis of the Use of Sabermetric Statistics in Baseball

Jack McKay
Period 1
January 20, 2005
Draft

Abstract:

For years, baseball theorists have pondered the most basic question of baseball statistics: which statistic most accurately predicts which team will win a baseball game. With this information, baseball teams can rely on technological, statistical-based scouting organizations. The book, *Moneyball* addresses the advent of sabermetric statistics in the 1980s and 1990s and shows how radical baseball thinkers instituted a new era of baseball scouting and player analyzation. This project analyzes which baseball statistic is the single most important. It has been found that new formulas, such as OBP, OPS, and Runs Created correlate better with the number of runs a team scores than traditional statistics such as batting average.

Introduction:

For some time, a baseball debate has been brewing. Newcomers and sabermetricians (the "Statistics Community") feel that baseball can be analyzed as a scientific entity. The *Sabermetric Manifesto* by Bill James serves as the Constitution for these numbers-oriented people. Also, *Moneyball* by Michael Lewis serves as the successful model of practical application of their theories. Traditional scouts (the "Scouting Community") contend that baseball statistics should not over-analyzed and stress the importance of intangibles and the need for scouts. The debate can also be interpreted in terms of statistics. Baseball lifers feel that stats such as batting average are the most important. Meanwhile, the Statistics Community feels that complex, formulaic stats can better predict a player's contributions to a team. The discussion continues in the offices of baseball teams around the country: are computer algorithms better than human senses?

From a statistical sense, baseball is an ideal sport. Plate appearances are discrete events with few, distinct results. In fact, results can be limited to a few distinct outcomes: hit, walk, or out. Outcomes can also be expressed more specifically: single, double, triple, home run, walk, strike-out, fly-out… etc. Most importantly, the outcomes of past plate appearances can accurately predict the outcomes of future plate appearances. Baseball statisticians continue to desire more information in their field in order to become better at analyzing the past and predicting the future.

Definition of Terms

BA – Batting Average
OBP – On Base Percentage
OPS – On Base Percentage Plus Slugging
OPS Adjusted – On Base Percentage * 1.2 Plus Slugging Percentage
Runs Created – On Base Percentage * Slugging Percentage

Theory: Sabermetric Teachings

(1) Best Pitching Statistic

DIPS is Defensive Independent Pitching Statistic - "Looking mainly at a pitcher's strikeouts, walks and home runs allowed per inning does a better job of predicting ERA than even ERA does. It's very counterintuitive to see that singles and doubles allowed don't matter a whole lot moving forward." (Across the Great Divide)

The Defense Independent Pitching Statistic was invented to provide by sabermetricians as a alternate statistic to ERA. Sabermetricians think that ERA does a poor job of future prediction because it is greatly altered by stadium characteristics, the opposing team, luck, and defense. Hence, the invention of DIPS.

(2) Best Hitting Statistic
OBP and OPS as more indicative hitting statistics than the current default: Batting Average. OBP is on-base percentage, which is essential a measure of batting average and plate discipline. OPS is Slugging Percentage plus OBP, which gives a measure of power and plate discipline. In fact, it has been shown by my research that OPS does the best job of any conventional statistics in correlating to wins. Old-school scouts say that Batting Average is a better predictor of a player's potential, because plate discipine can be learned.

The ultimate example is that one team could hit three home runs to get three runs, while another team could have two walks, followed by a home run, to get three runs. In this case, it is shown that walks are important!

(3) The lack of need for Scouts

“The scouts have only a limited idea of what the guy's gonna do. He might do this, he might do that, he might be somewhere in the middle. What you're trying to do is you're trying to take the guys who you think have the best chance. I fully admit that you can't tell the future via stats. My point is that scouting has that equal amount of unpredictability. You can only know so much. You're scouts, you're not fortune tellers.”

It is many sabermetricians view that real scouts are important: nowadays, “scouts” can operate from a laptop looking at a baseball's player's stats.

(4) Draft College Players

“A player who is 21 is simply closer to his peak abilities than a player who's 18.”

Therefore, it is extremely risky to draft younger, usually high-school players. How they play in high school may be far away from how they play in the pros. Meanwhile, the best College Players can sometimes be plugged into a Major League Baseball team's rotation a season or two after being drafted. Simply, college players are more proven.

(5) Use Minor League Statistics to predict Major League numbers

Guys who have higher on-base percentages in Triple-A tend to have higher on-base percentages in the major leagues. However, this area is very underdeveloped, and no studies have been conducted in the field.

Method

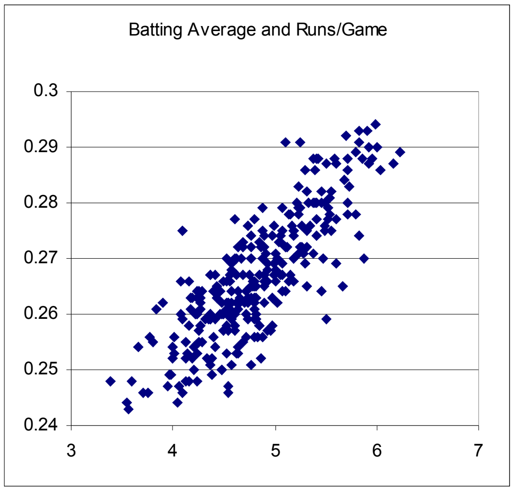
My own analysis had two parts. First, I obtained statistical data about teams in the past ten years and entered it into a Microsoft Excel spreadsheet. I calculated the correlation between certain statistics of teams and the number of runs they scored that season.

The second part of my research consists of a computer program in C++. Right now, I have the "engine" of my program working. The program plays a game between two teams, then outputs a full box score displaying many hitter statistics. With this framework in place, I can tell the program to play the game many times, and store the statistics in variables that I can output to a different file.

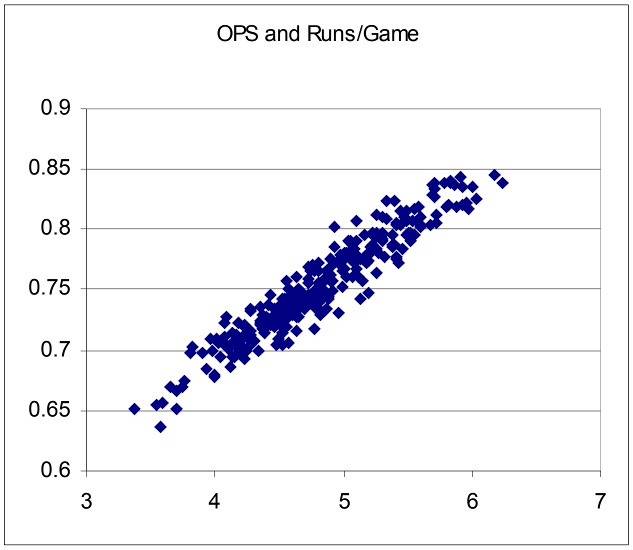
Ways In the future in which I will produce graphs based on my C++ program:

1. Graph of percentage of games won versus number of games played... should even out at the correct percentage when sample size is larger enough.
2. Graph of correlation percentage between OBP and games won, SLUG and games won... and more. Should be a bar graph because the correlation is just a number from -1 to 1.
3. Bar graph with effect of artificially changing OBP and SLUG... what is the effect on runs scored. Which has a bigger effect?

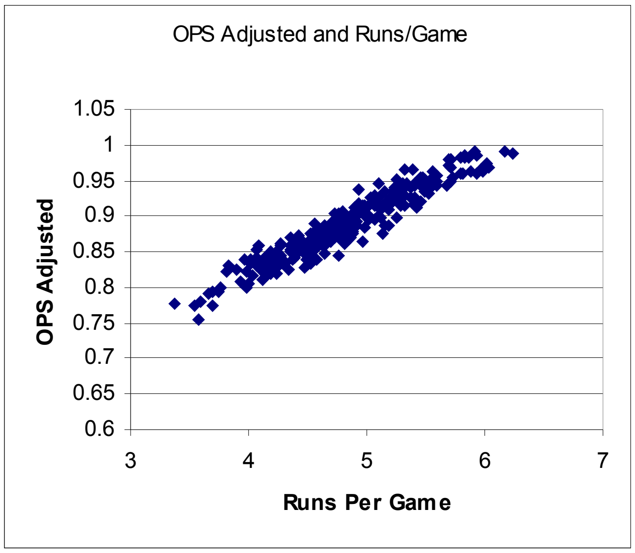
Findings: Correlation Data



Correlation = 0.824



Correlation = 0.953 (better)



Correlation = 0.956 (best)

Code Samples

PRELIMINARY AT-BAT ALGORITHM

```
int Baseball::AtBatResult()   //Still Random
{
          RandomNumber = rand() %1000;

           if (RandomNumber > 975)
                    return (4);

           else if (RandomNumber > 968)
                    return (3);

           else if (RandomNumber > 915)
```

```
                  return (2);

         else if (RandomNumber > 750)
                return (1);

         else if (RandomNumber > 660)
                  return (5);
         else
         {
                 return (0);}
}

void Baseball::PrintBoxScore()
{

//--------------------------PRINT TEAM NAMES AND RUNS (PRINT WINNING TEAM
FIRST)--------------
        if (Status.HomeTeam.HitterStats.Runs >=
Status.AwayTeam.HitterStats.Runs)
        {
                cout<<Status.HomeTeam.Name<<"
"<<Status.HomeTeam.HitterStats.Runs;
                cout<<", "<<Status.AwayTeam.Name<<"
"<<Status.AwayTeam.HitterStats.Runs<<endl;
        }

        else if (Status.AwayTeam.HitterStats.Runs >
Status.HomeTeam.HitterStats.Runs)
        {
                cout<<Status.AwayTeam.Name<<"
"<<Status.AwayTeam.HitterStats.Runs;
                cout<<", "<<Status.HomeTeam.Name<<"
"<<Status.HomeTeam.HitterStats.Runs<<endl;
        }

//----------------PRINT INNING BY INNING SCORE RUNDOWN--------------------
        cout<<endl;
        cout<<Status.HomeTeam.Name<<" ";
        int c;
        for (c = 1;c<10;c++)
        {
                cout<<Status.HomeTeam.HitterStats.RunsByInning[c];
                cout<<" ";
        }
        cout<<endl;
        cout<<Status.AwayTeam.Name<<" ";
        for (c = 1;c<10;c++)
        {
                cout<<Status.AwayTeam.HitterStats.RunsByInning[c];
                cout<<" ";
        }
        cout<<endl;
        cout<<endl;
//--------------------PRINT ACTUAL BOXSCORE, EACH PLAYERS HITS, RBI,
WALKS...---------------
//HOMETEAM
        cout<<Status.HomeTeam.Name<<endl;
        cout<<endl;

        cout<<"          "<<"  "<<"AB"<<"  "<<"H  "<<"  "<<"RBI"<<"
"<<"BB"<<"  "<<"HR"<<"  "<<"TB"<<endl;

        for (c = 1; c<10;c++)
        {
                cout<<setw(10)<<Status.HomeTeam.Hitters[c].Name;
                cout<<"  ";
                cout<<setw(2)<<(Status.HomeTeam.Hitters[c].Stats.PlateApp -
Status.HomeTeam.Hitters[c].Stats.Walks);
                cout<<"  ";
                cout<<setw(2)<<Status.HomeTeam.Hitters[c].Stats.Hits;
                cout<<"  ";
                cout<<setw(2)<<Status.HomeTeam.Hitters[c].Stats.RBI;
                cout<<"  ";
                cout<<setw(2)<<Status.HomeTeam.Hitters[c].Stats.Walks;
                cout<<"  ";
                cout<<setw(2)<<Status.HomeTeam.Hitters[c].Stats.HR;
                cout<<"  ";
                cout<<setw(2)<<Status.HomeTeam.Hitters[c].Stats.Bases<<endl;
        }//end for loop
//AWAYTEAM
cout<<endl;
        cout<<Status.AwayTeam.Name<<endl;
        cout<<endl;

        cout<<"          "<<"  "<<"AB"<<"  "<<"H  "<<"  "<<"RBI"<<"
"<<"BB"<<"  "<<"HR"<<"  "<<"TB"<<endl;


        for (c = 1; c<10;c++)
        {
                cout<<setw(10)<<Status.AwayTeam.Hitters[c].Name;
                cout<<"  ";
                cout<<setw(2)<<(Status.AwayTeam.Hitters[c].Stats.PlateApp -
Status.AwayTeam.Hitters[c].Stats.Walks);
                cout<<"  ";
                cout<<setw(2)<<Status.AwayTeam.Hitters[c].Stats.Hits;
                cout<<"  ";
                cout<<setw(2)<<Status.AwayTeam.Hitters[c].Stats.RBI;
                cout<<"  ";
                cout<<setw(2)<<Status.AwayTeam.Hitters[c].Stats.Walks;
                cout<<"  ";
                cout<<setw(2)<<Status.AwayTeam.Hitters[c].Stats.HR;
                cout<<"  ";
                cout<<setw(2)<<Status.AwayTeam.Hitters[c].Stats.Bases<<endl;
        }//end for loop
```

BOX SCORE FUNCTION

```
void Baseball::PrintBoxScore()
{

//--------------------------PRINT TEAM NAMES AND RUNS (PRINT WINNING TEAM
FIRST)--------------
```

```
        if (Status.HomeTeam.HitterStats.Runs >=
Status.AwayTeam.HitterStats.Runs)
        {
                cout<<Status.HomeTeam.Name<<"
"<<Status.HomeTeam.HitterStats.Runs;
                cout<<", "<<Status.AwayTeam.Name<<"
"<<Status.AwayTeam.HitterStats.Runs<<endl;
        }

        else if (Status.AwayTeam.HitterStats.Runs >
Status.HomeTeam.HitterStats.Runs)
        {
                cout<<Status.AwayTeam.Name<<"
"<<Status.AwayTeam.HitterStats.Runs;
                cout<<", "<<Status.HomeTeam.Name<<"
"<<Status.HomeTeam.HitterStats.Runs<<endl;
        }

//----------------PRINT INNING BY INNING SCORE RUNDOWN--------------------
        cout<<endl;
        cout<<Status.HomeTeam.Name<<" ";
        int c;
        for (c = 1;c<10;c++)
        {
                cout<<Status.HomeTeam.HitterStats.RunsByInning[c];
                cout<<" ";
        }
        cout<<endl;
        cout<<Status.AwayTeam.Name<<" ";
        for (c = 1;c<10;c++)
        {
                cout<<Status.AwayTeam.HitterStats.RunsByInning[c];
                cout<<" ";
        }
        cout<<endl;
        cout<<endl;
//--------------------PRINT ACTUAL BOXSCORE, EACH PLAYERS HITS, RBI,
WALKS...---------------
//HOMETEAM
        cout<<Status.HomeTeam.Name<<endl;
        cout<<endl;

        cout<<"          "<<"  "<<"AB"<<"  "<<"H  "<<"  "<<"RBI"<<"
"<<"BB"<<"  "<<"HR"<<"  "<<"TB"<<endl;

        for (c = 1; c<10;c++)
        {
                cout<<setw(10)<<Status.HomeTeam.Hitters[c].Name;
                cout<<"  ";
                cout<<setw(2)<<(Status.HomeTeam.Hitters[c].Stats.PlateApp -
Status.HomeTeam.Hitters[c].Stats.Walks);
                cout<<"  ";
                cout<<setw(2)<<Status.HomeTeam.Hitters[c].Stats.Hits;
                cout<<"  ";
                cout<<setw(2)<<Status.HomeTeam.Hitters[c].Stats.RBI;
                cout<<"  ";
                cout<<setw(2)<<Status.HomeTeam.Hitters[c].Stats.Walks;
                cout<<"  ";
                cout<<setw(2)<<Status.HomeTeam.Hitters[c].Stats.HR;
                cout<<"  ";
                cout<<setw(2)<<Status.HomeTeam.Hitters[c].Stats.Bases<<endl;
        }//end for loop
//AWAYTEAM
cout<<endl;
        cout<<Status.AwayTeam.Name<<endl;
        cout<<endl;

        cout<<"          "<<"  "<<"AB"<<"  "<<"H  "<<"  "<<"RBI"<<"
"<<"BB"<<"  "<<"HR"<<"  "<<"TB"<<endl;


        for (c = 1; c<10;c++)
        {
                cout<<setw(10)<<Status.AwayTeam.Hitters[c].Name;
                cout<<"  ";
                cout<<setw(2)<<(Status.AwayTeam.Hitters[c].Stats.PlateApp -
Status.AwayTeam.Hitters[c].Stats.Walks);
                cout<<"  ";
                cout<<setw(2)<<Status.AwayTeam.Hitters[c].Stats.Hits;
                cout<<"  ";
                cout<<setw(2)<<Status.AwayTeam.Hitters[c].Stats.RBI;
                cout<<"  ";
                cout<<setw(2)<<Status.AwayTeam.Hitters[c].Stats.Walks;
                cout<<"  ";
                cout<<setw(2)<<Status.AwayTeam.Hitters[c].Stats.HR;
                cout<<"  ";
                cout<<setw(2)<<Status.AwayTeam.Hitters[c].Stats.Bases<<endl;
        }//end for loop
```

HITTER STATISTICS CLASS

```
class HitStat
{
        public:
        HitStat::HitStat()
        {
                Runs = 0; Bases = 0; Hits = 0;
                Walks = 0; Singles = 0; Doubles = 0;
                Triples = 0; HR = 0; PlateApp = 0;
                OPS = 0; AVG = 0; OBP = 0; SLUG = 0; RBI = 0;

                int LoopInt;
                for (LoopInt = 1; LoopInt < 10; LoopInt++)
                        RunsByInning[LoopInt] = 0;
        }

        int Runs;
        int RunsByInning[10];  //use 1- 9.. used for boxscore
        int Bases;
        int Hits;
        int Walks;
        int Singles;
        int Doubles;
        int Triples;
        int RBI;
        int HR;
        int PlateApp;
        float OPS;
        float AVG;
        float OBP;
        float SLUG;
};
```

References

"Across the Great Divide."
http://sports.espn.go.com/mlb/columns/story?columnist=schwarz_alan&id=1963830

Lewis, Michael. Moneyball. W.W. Norton, New York. 2004."Sabermetrics."

"Sabermetrics." http://en.wikipedia.org/wiki/Sabermetrics

"Sabermetric Revolution Sweeping the Game."
http://proxy.espn.go.com/mlb/columns/story?columnist=neyer_rob&id=1966043