A Method of Analyzing a Baseball Pitcher's Performance Based on Statistical Data Mining

D. Ezra Sidran
Dept. of Computer Science
University of Iowa
Iowa City, IA
dsidran@cs.uiowa.edu

ABSTRACT
In this paper, we describe a method for extracting pitchers, pitches, pitch results (e.g. hits, foul balls, balls, called strikes, etc.) from the EVN data format and outputting the results in a format that allow them to be imported and graphed in a Microsoft Excel ™ program.

Categories and Subject Descriptors
I.2.1 Applications and Expert Systems

Keywords
Data mining, statistical analysis, baseball, pitcher

INTRODUCTION
American baseball is unique among the world's sports in that for over a century every play of every professional baseball game has been meticulously recorded. Furthermore, since, about 1965, the detail of the recorded games has included every pitch thrown and the results of the pitch. This has resulted in a staggering amount of data which has not, as yet, been fully data-mined.

The sport of baseball attracts many statisticians (both amateur and professional). [1] [2] [3] [4] Also, numerous books, [5] [6] which are compendiums of baseball statistics, have been published. A major goal of this work is the hope of predicting the future performance of baseball players on their past statistics.

One of the most intriguing problems that a baseball manager faces is the decision as to when a tiring, or faltering, pitcher should be removed from the game and replaced by a substitute pitcher. This decision making process is made more difficult by the fact that the substitute pitcher needs about 10 minutes to "warm up" before he can enter the game. However, the substitute pitcher should not warm up for an excessive length of time unless he becomes too fatigued. Consequently, there is a need for a function that can determine in advance when a pitcher is about to falter and when the substitute pitcher should begin warming up.

Our first step in the development of this function is the creation of a new metric to measure a pitcher's performance from data extracted from recorded baseball games. This new metric is then output in a format which allows us to create a visual representation of the metric for analysis. We are hopeful that analysis of these metrics will assist in the creation of a function that will predict when a baseball pitcher should be removed from a game before he falters and tires.

The metric that we use to analyze a pitcher's performance is the accuracy of every pitch thrown and the results of that pitch.

We are hopeful that the development of this function will be a significant contribution to baseball management strategy.

This introduction is followed by sections describing the data files used, problems related to extracting the data from the files, the algorithm for extracting the data, calculating the metrics and outputting the metrics in the desired format, examples, future work and a conclusion.

THE BASEBALL DATA FILES
The files employed in this experiment were provided by the Retrosheet Organization, a volunteer organization that has collected and posted on the Internet Play-by-Play files of every Major League professional baseball game from 1965 to 1992. [7]. While these files are extremely detailed, they are not organized in a traditional relational database format. Indeed, these files (called EVN from their extension) are simply ASCII text files in which each line starts with an identifier keyword and the variable length and variable number of data fields are separated by commas. Each EVN file contains records of all the games played by a particular team for an entire year; consequently each EVN file contains the records for over 160 games each. Some examples below:

```
id,CHN199204100
```

This is an example of an identification data field; in this case a game played by the Chicago Cubs of the National Baseball League on April 10, 1992.

```
info,hometeam,CHN
```

This is an example of an information field showing that the home team was the Chicago Cubs.

```
start,lankr001,"Ray Lankford",0,1,8
```

This is an example of a starting lineup information field; in this case showing that a player named "Ray Lankford", who has a corresponding ID of lankr001, started for the visiting team playing the position of center field and batted first.

```
play,1,0,lankr001,22,BFCFFBFS,K
```

This is an example of a 'play' information field; in this case the information can be read as: first inning, visiting team, player at bat = lankr001, the 'count' was two balls and two strikes, the sequence of pitches was ball, foul ball, called strike, foul ball, foul ball, ball, foul ball, swinging strike, strike-out.

```
sub,vizcj001,"Jose Vizcaino",1,1,6
```

This is an example of a 'substitution' information field; in this case showing that "Jose Vizcaino" with id vinzcj001, came into the game as a pitcher for the home team in the 6th inning.

There are also other information fields, such as 'com' for comments, and 'data' for final box scores which we ignored for this experiment.

PROBLEMS RELATED TO EXTRACTING THE DATA FROM THE EVN FILES

The EVN data files are large (> 780 kb) ASCII files. An analysis of one such file, for the 1992 Chicago Cubs season, showed that it contained a large number of 'events' that had to be extracted for analysis:

Number of games in file = 162
Number of runs in file = 1,214
Number of plays in file = 14,723
Number of walks in file = 1,103
Number of singles in file = 4,176
Number of doubles in file = 3,369
Number of triples in file = 229
Number of homers in file = 211
Number of hit batsmen in file = 75
Number of stolen bases in file = 193
Number of strike-outs in file = 1,731
Number of sacrifice-hits in file = 0
Average runs per game = 7.493827 (both teams)
Average plays per run = 12.127677

Because the data fields were of variable length, could appear anywhere throughout the file and could only be identified by parsing the keyword identifier the creation of an algorithm for extracting the necessary data from the EVN files was not a trivial matter.

THE ALGORITHM FOR EXTRACTING THE DATA FROM THE EVN FILES

Following is the algorithm for extracting the data from the EVN files. The algorithm was designed with a number of 'defined' values which appear in all capital letters such as WALK, SINGLE, DOULBE, etc. This allows for easily changing the value of these metrics (as we shall see in the next section).

Algorithm for extracting data from EVN files:

Note: This symbol ⇦ is used to represent appending a string to another string.

```
NumGames, NumPlays , NumStolen, NumSingles, NumDoubles, Num-
Triples, NumHomers, NumHitBatsmen, NumKs← 0
HomePitcherStats, VisitorPitcherStats ← Ø
PitchFlag ← FALSE
while a line of text can be read from the EVN file
   do read line of text
   if the first 2 chars of line of text = "id"
      NumGames ← NumGames + 1
      HomePitcherStats ⇦ GameID
      VisitorPitcherStats ⇦ GameID

   if the first 5 chars of line of text = "start"
      if the second to the last char of line of text = "1"
      // This is the notation for the pitcher
         if the sixth from the last char of line of text = "1"
         // This is the notation for the Home Team
         HomePitcherStats ⇦ HomePitcher.ID
         HomeBalls, HomeStrikes, HomePitcherScore ← 0;
         HomePitcherStats ⇦ ","
         Write HomePitcherStats to output file

         if the sixth from the last char of line of text = "0"
         // This is the notation for the Visitor Team
         VisitorPitcherStats ⇦ VisitorPitcher.ID
         VisitorBalls, VisitorStrikes, VisitorPitcherScore ← 0;
         VisitorPitcherStats ⇦ ","
         Write VisitorPitcherStats to output file

   if the first 3 chars of line of text = "sub"
   CommaCount ← 0
   while CommaCount < 3
      do read each char in line of text
         if char[i] = ","
            CommaCount ← CommaCount + 1
      if the second to the last char of line of text = "1"
      // This is the notation for the pitcher
         if the sixth from the last char of line of text = "1"
         // This is the notation for the Home Team
         HomePitcherStats ⇦ HomePitcher.ID
         HomeBalls, HomeStrikes, HomePitcherScore ← 0
         HomePitcherStats ⇦ ","
         Write HomePitcherStats to output file

         if the sixth from the last char of line of text = "0"
         // This is the notation for the Visitor Team
         VisitorPitcherStats ⇦ VisitorPitcher.ID
         VisitorBalls, VisitorStrikes, VisitorPitcherScore ← 0
         VisitorPitcherStats ⇦ ","
         Write VisitorPitcherStats to output file

   if the first the first four letters of the line of text = "play"
   NumPlays ← NumPlays +1
      if the seventh char of the line of text = '0'
         AtBat ← VISITOR;
       else
         AtBat ←HOME;
CommaCount, i ← 0
   while CommaCount < 5 and PitchFlag = FALSE
      do read each char in line of text
      i ← i + 1
```

```
    if char[i] = ","
        CommaCount ← CommaCount + 1
PitchFieldStart = i+1;
PitchFlag ← TRUE;
if (CommaCount = 6) // Reading the play field
    PitchFieldEnd ← i - 1;
    PitchFlag ← FALSE;
        copy the string of the playfield into the string PitchFieldLine
        for j ← 0, j <= length of  PitchLineLength, j ← j + 1
        switch on  PitchField[j]
        case 'C':    // Pitch was a called strike
        TotalPitches ← TotalPitches +1
        if AtBat = HOME
            VisitorPitcherScore ← VisitorPitcherScore +1
            VisitorPitcherStats ⇦  VisitorPitcherScor
        else
            HomePitcherScore ← HomePitcherScore +1
            HomePitcherStats ⇦  HomePitcherScore

        case 'S':    // Pitch was a swinging strike
        TotalPitches ← TotalPitches +1
        if AtBat = HOME
            VisitorPitcherScore ← VisitorPitcherScore +1
            VisitorPitcherStats ⇦  VisitorPitcherScor
        else
            HomePitcherScore ← HomePitcherScore +1
            HomePitcherStats ⇦  HomePitcherScore

        case 'X':    // Pitch was a swinging strike
        TotalPitches ← TotalPitches +1
        if AtBat = HOME
            VisitorPitcherScore ← VisitorPitcherScore +1
            VisitorPitcherStats ⇦  VisitorPitcherScor
        else
            HomePitcherScore ← HomePitcherScore +1
            HomePitcherStats ⇦  HomePitcherScore

        case 'F':    // Pitch was a foul ball
        TotalPitches ← TotalPitches +1
        if AtBat = HOME
            VisitorPitcherScore ← VisitorPitcherScore +1
            VisitorPitcherStats ⇦  VisitorPitcherScor
        else
            HomePitcherScore ← HomePitcherScore +1
            HomePitcherStats ⇦  HomePitcherScore

        case 'B':    // Pitch was a called a  ball
        TotalPitches ← TotalPitches +1
        if AtBat = HOME
            VisitorPitcherScore ← VisitorPitcherScore -1
            VisitorPitcherStats ⇦  VisitorPitcherScor
        else
            HomePitcherScore ← HomePitcherScore -1
            HomePitcherStats ⇦  HomePitcherScore
        for j ← 0, j <= length of  PlayLineLength, j ← j + 1
        switch on  PlayField[j]
        case 'W': // Play was a 'walk'
        NumWalks ← NumWalks + 1;
        if AtBat = HOME)
            VisitorPitcherScore ← VisitorPitcherScore + WALK
        VisitorPitcherStats ⇦  VisitorPitcherScore
        else
            HomePitcherScore ← HomePitcherScore + WALK;
            HomePitcherStats ⇦  HomePitcherScore

        case 'S':
        if PlayField[j+1] = 'B' // It is a 'stolen base'
            NumStolen ← NumStolen + 1
            if AtBat = HOME
                VisitorPitcherScore ← VisitorPitcherScore  +
                        STOLENBASE
                VisitorPitcherStats ⇦  VisitorPitcherScore
            else
                HomePitcherScore ← HomePitcherScore  +
                        STOLENBASE;
            HomePitcherStats ⇦  HomePitcherScore

        else  // Must be a hit single
            NumSingles ← NumSingles +1
        if AtBat = HOME
                VisitorPitcherScore ← VisitorPitcherScore  + SINGLE;
                VisitorPitcherStats ⇦  VisitorPitcherScore
            else
                HomePitcherScore ← HomePitcherScore  + SINGLE;
                HomePitcherStats ⇦  HomePitcherScore

        case 'D':  // It's a double
        NumDoubles ← NumDoubles +1
        if AtBat = HOME
            VisitorPitcherScore ← VisitorPitcherScore  + DOUBLE;
            VisitorPitcherStats ⇦  VisitorPitcherScore
        else
            HomePitcherScore ← HomePitcherScore  + DOUBLE;
            HomePitcherStats ⇦  HomePitcherScore

        case 'T':    // Play was a 'triples'
        NumTriples ← NumTriples +1
        if AtBat = HOME
            VisitorPitcherScore ← VisitorPitcherScore  + TRIPLE;
            VisitorPitcherStats ⇦  VisitorPitcherScore
        else
            HomePitcherScore ← HomePitcherScore  + TRIPLE;
            HomePitcherStats ⇦  HomePitcherScore

        case '-':    // Play was a 'strike out'
        if PlayField[j+1] = 'H' // Somebody went home;
            NumRuns++;

        case 'K':    // Play was a 'strike out'
        NumKs ← NumKs + 1;
        if AtBat = HOME
            VisitorPitcherScore ← VisitorPitcherScore + STRIKEOUT
            VisitorPitcherStats ⇦  VisitorPitcherScore
        else
            HomePitcherScore ← HomePitcherScore + STRIKEOUT;
            HomePitcherStats ⇦  HomePitcherScore

        case 'H':
            if PlayField[j+1] = 'R' // It is a Home Run
            NumHomers ← NumHomers + 1
            if AtBat = HOME
                VisitorPitcherScore ← VisitorPitcherScore  + HOMER;
                VisitorPitcherStats ⇦  VisitorPitcherScore
            else
                HomePitcherScore ← HomePitcherScore  + HOMER;
                HomePitcherStats ⇦  HomePitcherScore
        if  PlayField[j+1] = 'P'                // It's a hit batsmen
            NumHitBatsmen ← NumHitBatsmen +1
        if AtBat = HOME
                VisitorPitcherScore ← VisitorPitcherScore
                + HITBATSMEN;
                VisitorPitcherStats ⇦  VisitorPitcherScore
            else
                HomePitcherScore ← HomePitcherScore
                + HITBATSMEN;
                HomePitcherStats ⇦  HomePitcherScore
Write HomePitcherStats  to output file
Write VisitorPitcherStats  to output file
```

CALCULATING THE METRICS

Every pitcher starts with a score of '0'. The results of each pitch are then added to the pitcher's running score. After every pitch the pitcher's score is then written to a string and a comma is appended to the string. Either at the end of a game or when a substitute pitcher is brought into a game the string of the pitcher's running score is written to an output ASCII file.

Table 1. Default values of metrics

Event:	Value added to pitcher's score
Ball	-1
Strike	+1
Walk	-1
Single	-1
Double	-2
Triple	-3
Home Run	-4
Stolen Base	-1
Ball Put In Play	+1
Foul Ball	+1

The above table shows the default values for the metrics used to evaluate a pitcher's performance. These values can easily be adjusted and modified if further experiments suggest different optimal values for the metrics.

A sample pitcher's running score follows:

PHI199204070,mulht001,1,2,3,2,3,1,2,3,2,3,2,1,0,-1,-2,-1,-2,-3,-4,-5,-6,-7,-6,-8,-9,-10,-11,-10,-11,-12,-13,-14,-13,-12,-13,-12,-13,-14,-13,-12,-11,-10,-12,-13,-12,-11,-12,-11,-10,-9,-10,-9,-8,-7,-6,-7,-6,-10,-12,-13,-12,-11,-12,-13,-12,-13,-15,-14,-13,-15,-16,-15,-16,-18,-17,-18,-20,-21,-22,-21,-22,-21,-20,-22,-23,-22,-21,-22,-24,-23,-22,-21,-20,-19,-20,-19,-18,-17,-18

The above example represents the pitching performance of the pitcher with the ID "mulht001" (who is a player named Terry Mulholland, for the Philadelphia Phillies) in a game played on April 7, 1992. Starting with an initial value of 0 the results of every pitch are added to the running score and appended with a comma to the string. In this example Mr. Mulholland was removed in the seventh inning and replaced by a substitute pitcher. At the time Mr. Mulholland left the game his score was -18.

EXAMPLES OF ANALYZING A PITCHER'S PERFORMANCE

After the strings of the pitchers' running scores have been output to an ASCII file they can easily be imported into Microsoft Excel™ for graphing. Examples follow:

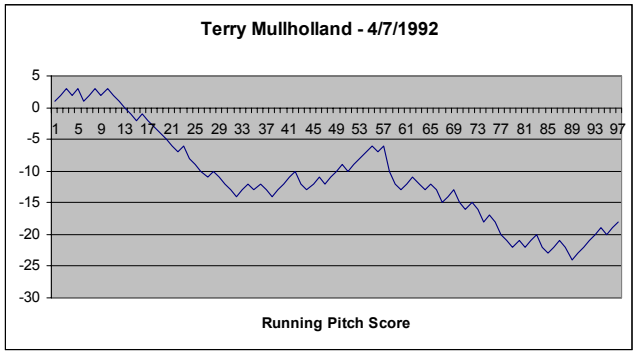

Figure 1. The running pitch score for Terry Mulholland on April 7, 1992. Mr. Mulholland was removed from the game in the seventh inning after having given up four earned runs.

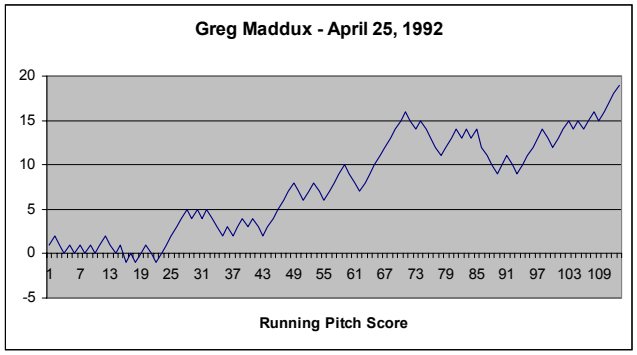

Figure 2. The running pitch score for Greg Maddux on April 25, 1992. Mr. Maddux was the 1992 winner of the National League Cy Young Award which is given to the best pitcher in the league.

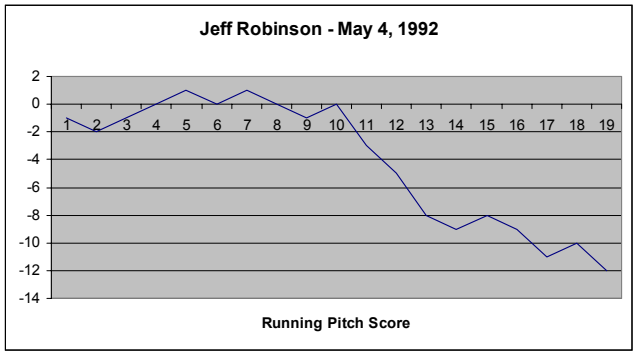

Figure 3. The running pitch score for substitute pitcher Jeff Robinson on May 4, 1992. Mr. Robinson pitched one inning, gave up 2 hits and one earned run (a very poor performance). The official scorer's comments embedded in the EVN file are: com,"32 year old mop up man finds new home. What a team. And the pitching! My god, its Jeff Robinson!"

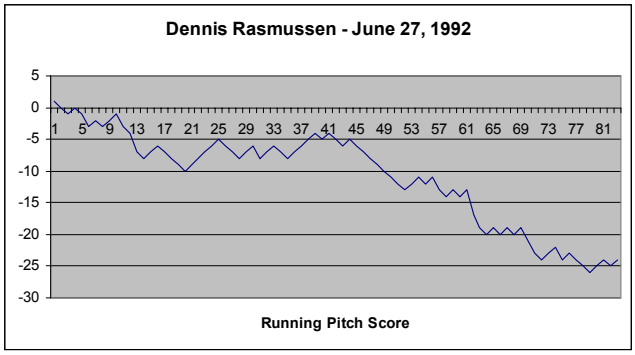

Figure 4. . The running pitch score for pitcher Dennis Rasmussen on June 27, 1992. In four innings Mr. Rasmussen gave up 4 hits, 3 earned runs, walked 2 and struck out none.

From the above examples we are optimistic that we can begin work on developing the desired function that will predict when a pitcher should be removed from a baseball game. For example, in Figure 4, above, Mr. Rasmussen, while not having a great start, held his own until approximately pitch number 41 when he had a disastrous 4th inning that resulted in his removal from the game. If Mr. Rasmussen had been pulled from the game before the fourth inning the results of the game would probably have been quite different.

FUTURE WORK

We have just begun work on data-mining the wealth of baseball statistics publicly available. We hope to develop a function that will predict when a pitcher should be removed from a baseball game before he falters (as in the case of the unfortunate Mr. Robinson on June 27, 1992). We also intend to convert the EVN files into a relational database which will facilitate numerous queries including some posed to us by the eminent baseball statistician Bill James.

Mr. James is interested in asking such questions as, "What percentage of all players who score runs reach base on walks? What percentage on singles? On doubles? On triples? On hit batsmen? How has that changed over time? What was the (walk) percentage in 1950? What was it in 1970? What is it now? How is that percentage different on good teams and bad teams? How does it differ between pitchers? Of the 2,178 runners who scored off of Nolan Ryan, how many reached base on a walk? How many stole a base before they scored? How many scored on singles, on home runs, on bases-loaded walks, on Wild Pitches? There are a billion questions to ask, and it is likely that the answers could improve our understanding of the game somewhat, possibly enough to impact on-field decisions. . ."[From a private email from Bill James].

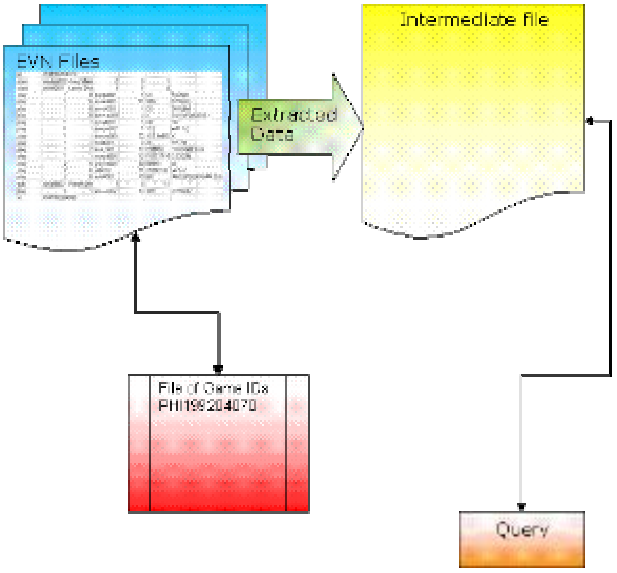

Figure 5. Schematic diagram of a proposed relational database extracted from multiple EVN files. Note: because each EVN file contains all games played by "Team A" the complete yearly set of EVN files will also include in the EVN file of "Team B" the games that it played with "Team A". Consequently, each game will appear twice in a complete yearly set of EVN files. Therefore, a separate index of Game IDs must be kept to ensure that duplicate games are not entered into the new Intermediate File of the Relational Database.

ACKNOWLEDGMENTS

We would like to acknowledge and thank the Retrosheet Organization who maintains the EVN file database and makes it publicly available on the Internet. We would also like to thank Mr. Bill James for his encouragement ad advice. We would also like to thank Dr. Hwanjo Yu of the University of Iowa his encouragement and advice.

REFERENCES

[1] Baseball Reference <http://www.baseball-reference.com/>

[2] The Baseball Archive <http://sports.espn.go.com/mlb/statistics>

[3] Baseball Stats <http://www.baseball-almanac.com/bstatmen.shtml>

[4] The Baseball Archives <http://www.baseball1.com/>

[5] James, B., "1982 Bill James Baseball Abstract." Ballantine. Note: Bill James has written a similar book of baseball statistics every year since 1982.

[6] James, B., "The Bill James Player Rating Book 1995."Fireside.

[7] Play-by-Play Data Files (Event Files) from 1965-1992. Retrosheet Organization <http://www.retrosheet.org/game.htm >