MARKOV LEAGUE BASEBALL

Baseball Analysis Using Markov Chains

VJ Asaro
Statistics Department
Cal Poly, San Luis Obispo

2015-2016

Table of Contents

I. Introduction

My love for the game of baseball began when I received a Ken Caminiti Padres jersey when I was three years old. My dreams and goals evolved around baseball throughout my entire life and I aspired to be a professional baseball player. However, as perfectly stated by a scout in the movie *Moneyball,* “We're all told at some point in time that we can no longer play the children's game, we just don't know when that's gonna be. Some of us are told at eighteen, some of us are told at forty, but we're all told.” Unfortunately, I was told at 18.

The passion I held for baseball didn’t end, it just transitioned to a new area, which is where baseball analytics stepped in. I became a statistics major to explore the behind-the-scenes action of baseball, and it led me to this project. Baseball analytics have been in the game ever since the ability to record data, but the start of the 21st century catapulted its importance and usefulness. It all began with Moneyball, as the 2002 Oakland A’s shocked the baseball world with 103 wins and the third lowest payroll in the MLB. Since then, teams across the league have used analytics to create optimal batting lineups and pitching rotations. As I saw this process unfold before my eyes, I had a new dream involving baseball, to create a new statistic that would be displayed on *Baseball Tonight* every night.

Throughout my first couple years in college, I realized creating a statistic was easier said than done. However, my classes about probabilities and literature readings led me to Markov chains. The purpose of this analysis is to use Markov chains to predict winning percentages of teams in a single season. Along the way, I dove into run expectancies, and player analysis before ultimately reaching my goal of predicting wins using Markov chains.

II. Background

The investigation process of advanced baseball analytics began when I purchased *The Book: Playing Percentages In Baseball*. This book is written by three esteemed baseball statisticians Tom Tango, Mitchel Lichtman, and Andrew Dolphin. In the first chapter, Toolshed, *The Book* explains the mathematical tools that were used throughout. One of these tools, is Markov chains. Markov chains, which are described in the next section, are very powerful systems that have been involved with sabermetrics since as early as 1960.

The data that I will be using can be found at Baseball Reference. This is a very popular baseball data site that consists of statistics as far back as 1871. Also, each of their data sets easily converts to CSV, and copied into an excel file. Once the data is in an excel file, I use R for all of my analysis and data transformations. Within R, I created two functions, prob and trans, which you can see in Section X. The prob function will read in the excel file that has numerical values, and will convert them into probabilities. The trans function will take the created probability data set and formally construct the Markov Chain process, which will be explained in further detail. Creating these functions made it easier to input any data set from Baseball Reference, as long as it has the desired variables, and I can run the analysis smoothly.

III. Markov Chains

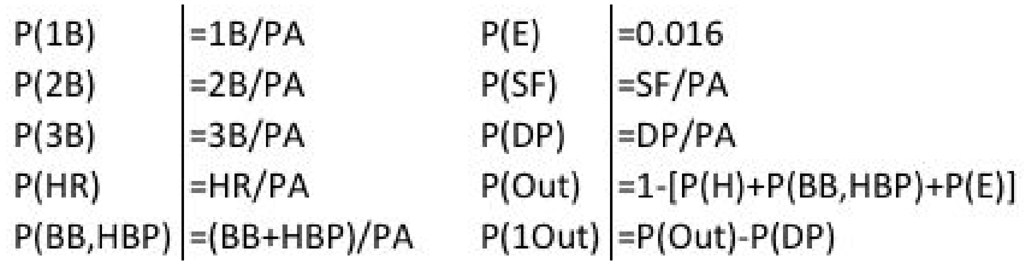
Markov chains are stochastic processes that follow a "one step" dependence system. This process starts in one state, and transitions to a following state, which is called a step. If the chain is currently in state i, then it moves to state j in one step with a probability denoted P_{ij}. One very important property of Markov chains is that P_{ij} does not depend upon which states the chain was in before the current state. In other words, the probability of the process to be in state j at time n+1, is only dependent on the process being in state i at time n. This is known as the conditional independence property that is mathematically constructed as the following:

Property: $P(X_{n+1}=j \mid X_n=i , X_{n-1}=i_{n-1} , \dots , X_0=i_0) \rightarrow P(X_{n+1}=j \mid X_n=i) = P_{ij}$

A Markov chain contains a number of states and creates a transition matrix (P) that include the probabilities of going from one state to the next. In baseball, these states represent the different base runners and number of outs combinations. For example, runner on first and 1 out, and bases loaded with 2 outs represent two different states. These states will be denoted as (1,1) and (123,2) respectively. There are 24 total combinations baseball states, not including the three-out state. The three-out state will be our absorbing state. A state is considered an absorbing state if the probability of going from that state to itself is 1. In baseball, the absorbing state will be third out of an inning. This makes sense because once you reach the third out, you cannot transition to any other state since the inning is over.

The table below shows you how the data from Baseball Reference is converted to probabilities using the prob function in R. Also, a key for what the variables represent.

Table 1: Formulas for computing probabilities from Baseball Reference data



P(1B)	=1B/PA	P(E)	=0.016
P(2B)	=2B/PA	P(SF)	=SF/PA
P(3B)	=3B/PA	P(DP)	=DP/PA
P(HR)	=HR/PA	P(Out)	=1-[P(H)+P(BB,HBP)+P(E)]
P(BB,HBP)	=(BB+HBP)/PA	P(1Out)	=P(Out)-P(DP)

1B= Single	E= Error	PA= Plate Appearance
2B= Double	SF= Sacrifice Fly	
3B= Triple	DP= Double Play	
HR= Homerun	BB,HBP= Walk, Hit by Pitch	

Using these probabilities, we create the transition probabilities from going from one state to another. For example, $(0,0) \rightarrow (1,0)$= P(1B)+P(BB,HBP)+P(E). Another example, $(1,0) \rightarrow (x,3)$= P(DP).

There are some assumptions the need to be made when constructing the transition probabilities. First, an error advances each base runner only one base. Multi-base errors are not included in the transitions. Next, there are no stolen bases in this model. Although they can be included with more in-depth investigating, they are left out for the sake of simplicity. Random baseball events such as the triple play, pass balls, and wild pitches are not included. Most importantly, are baserunning tendencies, or the way the base runners behave on the base path. I assumed that a runner on base advances the same number of bases as the batter. For example, if there is a runner on first, and the batter hits a double, then the base runner goes to third 100% of the time. If there is a runner on first and a triple is hit, then the runner will score. However, I made an adjustment to the situation where there is a runner on 2nd and a single is hit. According to data from 2011-2014, I inputted the probabilities of runners attempting to score from second on a single along with the probability of them being safe. These probabilities differ by the number of outs, which makes sense because base coaches are more willing to risk sending the runner home when there are 2 outs versus 0 outs. This allowed a more realistic approach to the base running tendencies in MLB games. Below are the probabilities of such events. Also, if the runner attempts to score, they are safe 96% of the time, which is consistent for all outs.

(x2x,0): 0.4673

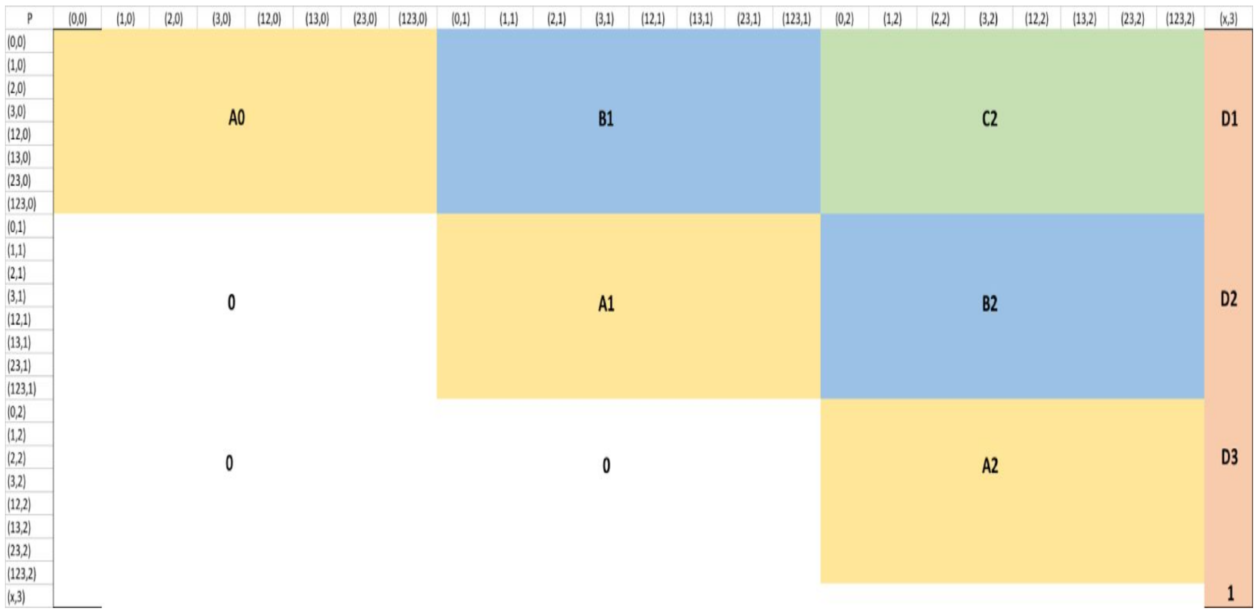
(x2x,1): 0.5927

(x2x,2): 0.9079

Situational hitting is also ignored in the system of Markov chains. I assumed that no matter the situation or inning, the batter has the same probabilities for each batting event.

Finally, we are ready to construct the transition matrix, which will be referred to as the P matrix. The P matrix contains all of the P_{ij} transition probabilities. Since there are 25 states including the absorbing three-out states, the matrix will be a 25x25. Constructing the 25x25 transition matrix is easier when split into multiple submatrices as shown in Figure 1. The A (yellow) matrices represent the probabilities of transitioning to a state with the same number of outs. For example, transitioning from (12,0) to (123,0) or (12,1) to (123,1). The B (blue) matrices contain the probabilities of transitioning to a state adding one out (12,1) to (12,2) or (0,0) to (0,1). The C2 matrix contains the probabilities of a double play with 0 outs, or adding two outs (1,0) to (0,2). Finally, The D matrices are are the probabilities to transitioning to the absorbing three-out states. The white submatrices contain all zeros because you cannot reduce the number of outs from state i to state j.

Figure 1: Submatrices of the transition matrix



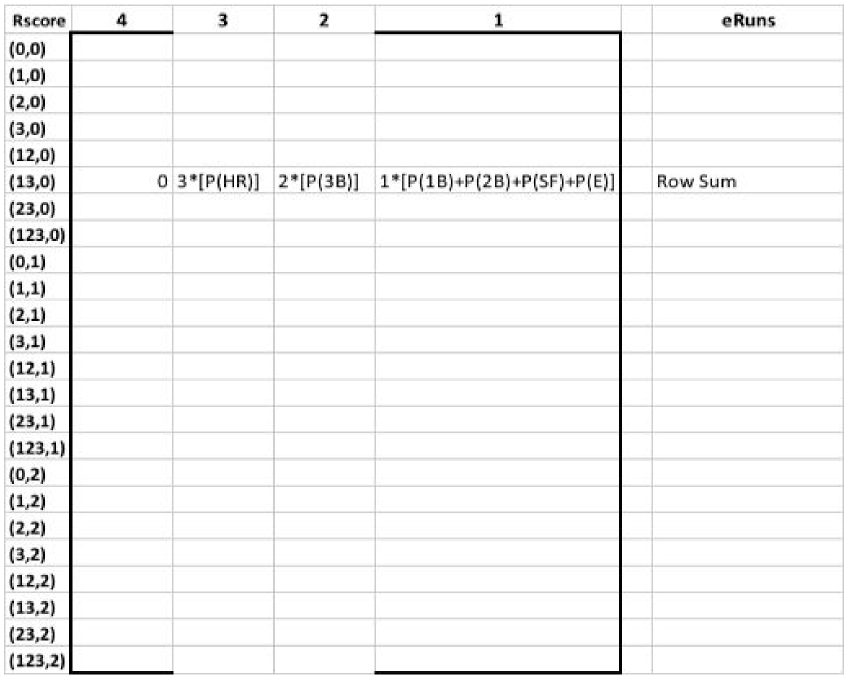
IV. Expected Runs

After the P matrix is fully constructed, the first task is to use it to find the expected runs for a half-inning. First, the **E** matrix is computed using the fundamental matrix equation **E = (I-Q)**$^{-1}$ where **Q** is a 24x24 submatrix of **P**, with the absorbing three-out states are removed (row and column) and **I** is the 24x24 identity matrix. Matrix **E** is then a 24x24 matrix whose values represent the expected number of visits to each state, starting from each state until the 3rd out occurs. Number of plate appearances before the absorbing state are calculated by the row sums of **E**.

Then, the **Rscore** matrix is a 24x4 matrix that contains the weighted probabilities of scoring a number of runs starting in each state. For example, in Figure 2 the row at state (13,0) will be 4(0), 3[P(HR)], 2[P(3B)], and 1[P(1B)+P(2B)+P(SF)+P(E)]. Where 4, 3, 2, and 1 represent scoring that many runs after one plate appearance and the probabilities of that happening. Note that it is impossible to score 4 runs in one plate appearance with two runners on base, and the other runs are possible with the respective probabilities.

The **eRuns** matrix is then computed by taking the row sums of **Rscore.** These values are the expected number of runs to be scored starting from each of the states after one plate appearance. Figure 2 is an example of how these matrices are constructed.

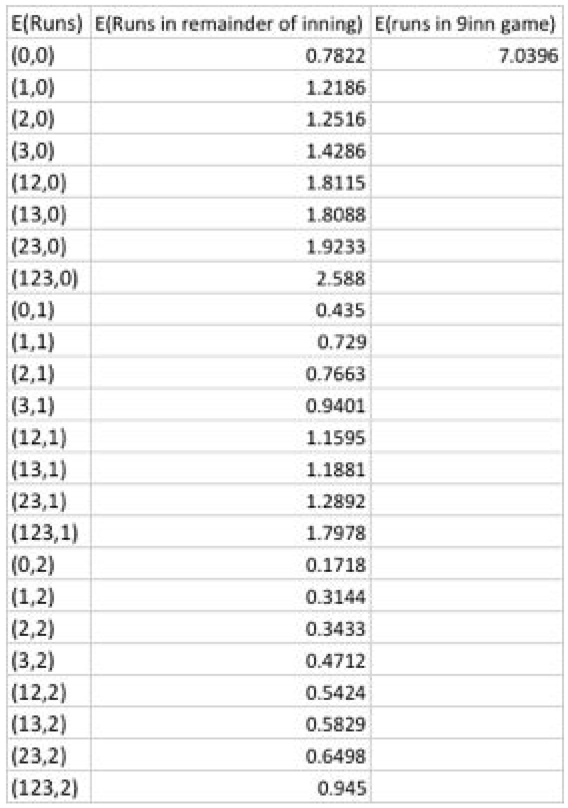
Figure 2: Example of the Rscore and eRuns matrices



Rscore	4	3	2	1		eRuns
(0,0)						
(1,0)						
(2,0)						
(3,0)						
(12,0)						
(13,0)	0	3*[P(HR)]	2*[P(3B)]	1*[P(1B)+P(2B)+P(SF)+P(E)]		Row Sum
(23,0)						
(123,0)						
(0,1)						
(1,1)						
(2,1)						
(3,1)						
(12,1)						
(13,1)						
(23,1)						
(123,1)						
(0,2)						
(1,2)						
(2,2)						
(3,2)						
(12,2)						
(13,2)						
(23,2)						
(123,2)						

Now that we have the expected number of visits to each state starting at each state in an inning, and the expected number of runs after one plate appearance from each state, we can find the expected number of runs for starting from each state for an inning. This 24x1 matrix is called **E(Runs)**. It is calculated as **E(Runs)**= **E*eRuns.** This makes intuitive sense based off the what the values of **E** and **eRuns** represent. Since every inning starts in the (0,0) state, you can multiply this value in **E(Runs)** by 9 to find the expected number of runs in a full 9 inning game. Figure 3 is what **E(Runs)** could potentially look like.

Figure 3: Example of a E(Runs) matrix based off of Mike Trout's career averages



E(Runs)	E(Runs in remainder of inning)	E(runs in 9inn game)
(0,0)	0.7822	7.0396
(1,0)	1.2186	
(2,0)	1.2516	
(3,0)	1.4286	
(12,0)	1.8115	
(13,0)	1.8088	
(23,0)	1.9233	
(123,0)	2.588	
(0,1)	0.435	
(1,1)	0.729	
(2,1)	0.7663	
(3,1)	0.9401	
(12,1)	1.1595	
(13,1)	1.1881	
(23,1)	1.2892	
(123,1)	1.7978	
(0,2)	0.1718	
(1,2)	0.3144	
(2,2)	0.3433	
(3,2)	0.4712	
(12,2)	0.5424	
(13,2)	0.5829	
(23,2)	0.6498	
(123,2)	0.945	

Assuming you are in state (12,0), then you expect 1.81 runs for the rest of the inning. Additionally, starting in (0,0) state for all 9 innings, you expect to score 9 * 0.7822 = 7.04 runs in a game. The expected number of runs using this Markov chain process are also called Markov Runs.

Now that we know how to find the number of Markov Runs in a baseball game, how can we put this information to use?

V. Player Analysis

First, I used the Markov Runs for player analysis. Hypothetically, if a batter were to hit every time during a 9 inning game, what is the expected number of runs they would produce? Table 2 shows these results for some of the best offensive years in MLB history.

Table 2: Markov Runs for historic offensive seasons

Name and Year	Significance	Markov Runs
Barry Bonds 2001	73 HR	17.55
Ichiro Suzuki 2004	262 Hits	7.20
Alex Rodriguez 2001	$252 Million Man	8.73
Bryce Harper 2015	2015 MVP	11.49
Babe Ruth 1923	The Big Bambino	18.81

For Barry Bonds in 2001 when he broke the HR record with 73 HR's, if he were to have every plate appearance for his team, then he produces 17.55 Markov runs. Not only does his 73 HR contribute to this massive number, but also his 177 walks. Ichiro Suzuki broke the most hits in a season record in 2004 with 262, but "only" produces 7.2 Markov runs. This is due to the characteristics of Ichiro as a hitter. Ichiro is known as a slash hitter who lives off of the single, which isn't nearly productive as extra base hits. Also, Ichiro does not walk a lot, so if he doesn't get a hit, then he most likely doesn't reach base. Alex Rodriguez had a massive year after signing his \$252 Million contract with Rangers in 2001, but again, his limited number of walks results in lower Markov runs then one would expect. Bryce Harper, the man put up a godly .460 OBP due to his high number of walks, along with his 42 HR, would produce 11.49 Markov runs. Lastly, out of historical respect, I evaluate Babe Ruth's 1923 MVP season. Speaking of OBP, Babe Ruth had a career high .545, paired with 41 HR and 45 2B, which results in an insane 18.81 Markov runs. However, one quick thing to note about Babe Ruth's number is that double plays were not a statistic back in 1923, so that inflates his number. We all know Babe Ruth's legendary love of beer and hot dogs, so I have a feeling he hit into quite a few of those in his heyday.

Besides looking at the some of the most impressive offensive seasons to date, Markov runs are a strong tool for evaluating a player's offensive productivity on his team. OPS is a statistic just as popular as batting average, which consists of OBP+SLG%. For clarification, SLG% is slugging percentage, the total number of bases divided by at-bats for a hitter. It is used to measure the power of a hitter.

$$\mathrm{SLG} = \frac{(\mathit{1B}) + (2 \times \mathit{2B}) + (3 \times \mathit{3B}) + (4 \times HR)}{AB}$$

Also, OBP is the percentage of the time a hitter reaches base safely in a number of plate appearances.

$$OBP = \frac{H + BB + HBP}{AB + BB + HBP + SF}$$

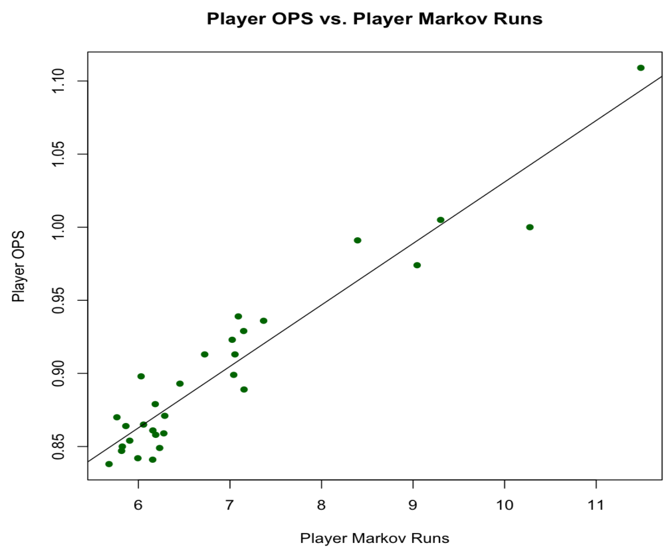
Therefore, OPS measures not only how often a batter gets on base, but also the impact of how he gets on base. It turns out that Markov runs are highly correlated with the OPS statistic. Table 3 shows the top 5 ranks in OPS for the 2015 season along with their respective Markov runs.

Table 3: Top 5 OPS players and Top 5 Markov Runs players

Name	OPS Rank	OPS	Markov Runs	Markov Runs Rank
Bryce Harper	1	1.109	11.49	1
Paul Goldschmidt	2	1.005	9.30	3
Joey Votto	3	1.000	10.28	2
Mike Trout	4	.991	8.39	5
Miguel Cabrera	5	.974	9.04	4

As you can see, the top 5 in OPS are also the top 5 in Markov runs. There is some slight variability in the order, but overall OPS and Markov runs tell the same story. Figure 4 is a graph showing the relationship between Markov runs and OPS for the top 30 hitters in 2015.

Figure 4: Player OPS vs. Player Markov Runs



VI. Team Analysis

Along with player analysis, you can also use Markov runs to find the productivity of a team. All of the above examples of Markov runs were for strictly offensive productivity. However, a team needs to be evaluated not only by their hitting, but their pitching as well. All of the stats used for hitting are also used to evaluate pitching. For example, 1B for hitting statistics is the number of singles the batter hits, but in pitching the same 1B statistic is the number of singles the pitcher gives up to opposing batters. Therefore, no changes need to be made to the data for pitchers and the same Markov runs process applies. Instead of finding the expected number of runs scored for a batter, we find the expected number of runs allowed for a pitcher. Once we have the Markov runs for a team’s offense (runs scored) and the Markov runs for a team’s pitching (runs allowed) we can determine the number of games a team can win in a season.

VII. Win Expectancy

Bill James, known as the Godfather of Sabermetrics, constructed an expected win percentage formula using runs scored and runs allowed for a team (Baseball Reference Bullpen):

$$E(WIN\%) = \frac{R^2}{R^2+RA^2} \ .$$

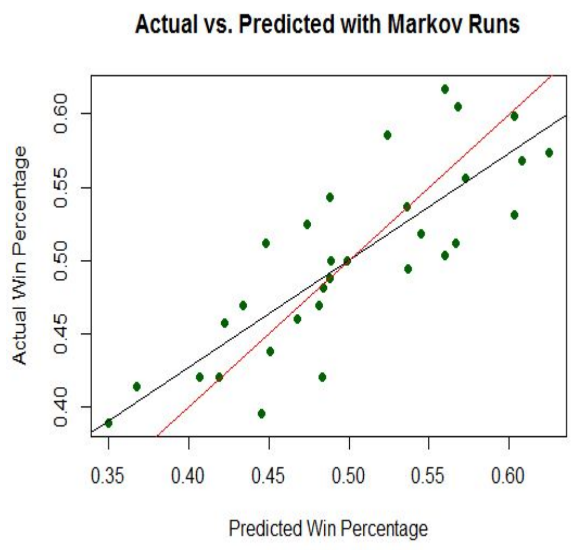
This formula was created by Bill James before computers and before the easy access to vast amounts of baseball data. Therefore, further analysis shows that a more accurate version of the formula is (Baseball Reference Bullpen):

$$E(WIN\%) = \frac{R^{1.81}}{R^{1.81}+RA^{1.81}}$$

Using my Markov runs scored and Markov runs allowed for a team as the parameters for the above formula, I found an expected win percentage for every Major League Baseball team using 2015 statistics. The first graph below shows the relationship between expected win percentage with Markov runs versus actual win percentage for the teams in 2015. The second graph shows the same relationship, except instead of using Markov runs for R and RA, the actual runs and runs allowed for the teams in 2015 are used.

Figure 5: Predicted Win Pct vs. Actual Win Pct with Markov Runs (5a),

Predicted Win Pct vs. Actual Win Pct with Observed Runs (5b)



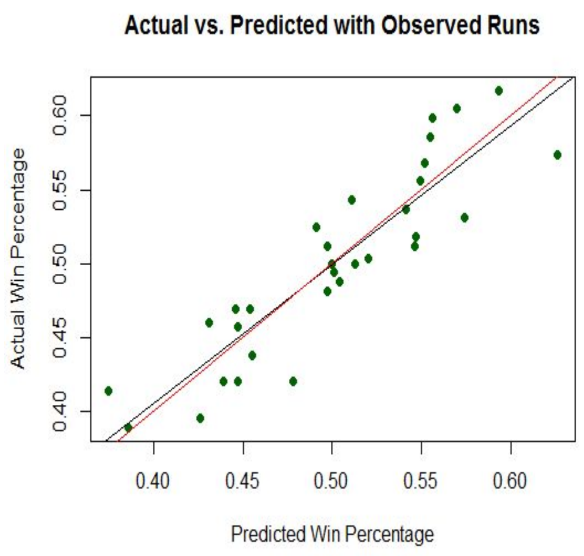


Table 4a: Regression Output for Figure 5a

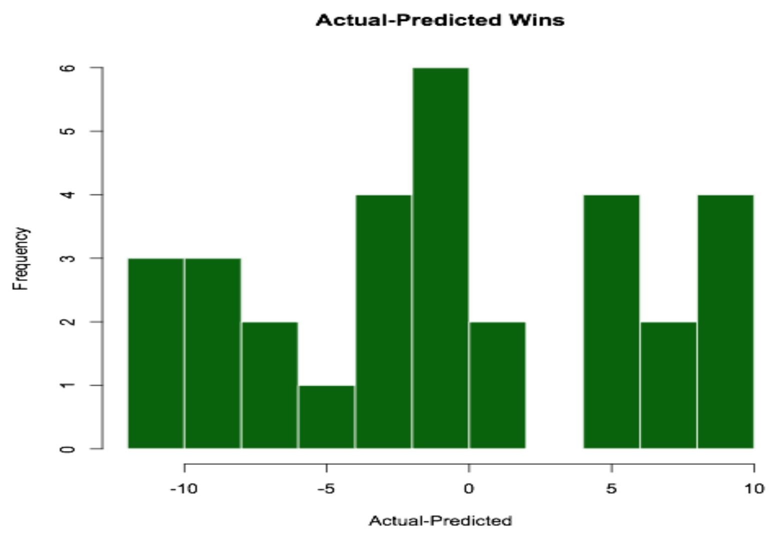
Markov Runs	Estimate	Std. Error	t value	p-value
Intercept	0.135	0.047	2.839	0.008
E(Win %)	0.73	0.094	7.778	<0.001
R^2	0.676			
Resid Std Error	0.037			

Table 4b: Regression Output for Figure 5b

Actual Runs	Estimate	Std. Error	t value	p-value
Intercept	0.027	0.044	0.628	0.535
E(Win %)	0.94	0.086	10.937	<0.001
R^2	0.805			
Resid Std Error	0.028			

Using the Markov runs to predict winning percentage accounts for roughly 68% of the variation in the actual winning percentage. On the other hand, using actual runs to predict winning percentage, we see that it accounts for nearly 81% of the variation in the actual winning percentage. The red lines in Figure 5a and 5b have an intercept of 0 and a slope 1. These denote a perfect prediction of winning percentage based on expected winning percentage, and are used to compare how close each respective model is to a perfect prediction. As expected, using actual runs scored and runs allowed will predict the actual win percentage of the teams better, but the two are not much different. Lastly, I include a graph that shows the distribution of the actual number of wins of the teams minus the predicted number of wins for a full 162 game season.

Figure 6: Distribution of Actual Wins minus Predicted Wins using Markov Runs



The distribution is centered around 0 which is reassuring, and the mean absolute difference was 5.4 wins. Therefore, using Markov runs and the revised pythagorean theorem of baseball formula to predict the number of wins in a 162 game season was only 5.4 games off from the actual number of wins on average. Some of the unusual observations include the Kansas City Royals (+10) and the Cleveland Indians (-10).

VIII. Conclusion

The game of baseball contains strategic situations that are not accounted for by a Markov process. However, the process effectively simulates the basic realities of a baseball game, which can be used to estimate winning percentage. Minor adjustments can be applied to Markov chains to help replicate a closer reality. Some of these include stolen bases, situational hitting, or even using pinch hitters at certain innings in the game. The effect of these inclusions may be minimal, but they are in fact part of a real baseball game.

There are numerous applications of Markov chains in baseball. Including but not limited to trade analysis, and simulation of batting orders. If you wanted to trade player A for player B, you would hope that your team improves. Therefore, retract player A's stats from your team averages and input player B's statistics. Does your team's Markov runs increase or decrease? If they decrease, then you might want to ask for another player in return, otherwise you risk losing more games in the future. In my Markov chain model, I am either running through the same player P matrix or a team's P matrix. However, it is possible to run through 9 different P matrices to simulate a batting order. You can put these matrices in an optimal batting order that produces the highest number of Markov runs. This idea can be used to test whether a pitcher batting 8th in the batting order instead of 9th will actually produce more runs for your team, and thus more wins.

IX. Bibliography

Albert, Jim and Mac Marchi. *Analyzing Baseball Data with R.* Boca Raton, FL. Taylor and Francis Group, 2013. Print

Pankin, Mark D. ”Baseball as a Markov Chain” Markov Baseball Models Theory. http://www.pankin.com/markov/intro.htm. 10 Sept. 2012.

Sports Reference LLC. Baseball-Reference.com - Major League Statistics and Information. http://www.baseball-reference.com/. 19 Sept. 2012

Sports Reference LLC. Baseball-Reference.com - Bullpen "Pythagorean Theorem of Baseball." http://www.baseball-reference.com/bullpen/Pythagorean_Theorem_of_Baseball

Tango, Tom M., Mitchel G. Lichtman, and Andrew E. Dolphin. *The Book: Playing The Percentages In Baseball*. Washington, D.C.: Potomac, 2007. Print.

Tesar, Naomi. “Estimating Expected Runs Using a Markov Model for Baseball”. http://www.edsolio.com/media/2/265/files/Tesar_FinalDraft.pdf.

X. R Code

Batting Code

```
teams.batting= read.csv("MLBBatting2015.csv", header=TRUE, nrow=31)

# Prob function to convert raw data to probabilities
prob.batting= function(data){
data$P.BB.HBP.= (data$BB+ data$HBP)/data$PA
data$P.1B.= (data$H- data$X2B-data$X3B- data$HR)/data$PA
data$P.2B.= data$X2B/data$PA
data$P.3B.= data$X3B/data$PA
data$P.HR.= data$HR/data$PA
data$P.DP.= data$GDP/data$PA
data$P.E.= (data$AB-data$SO)*(1-0.984)/data$PA
data$P.Out.= 1-(data$P.BB.HBP.+data$P.1B.+data$P.2B.+data$P.3B.+data$P.HR.+data$P.E.)
data$P.1Out.= data$P.Out.-data$P.DP.
data$P.SF.= data$SF/data$PA

# Extrabase is the probability of advancing from 2nd base to home on a single
data$P.Att.2nd.0out. = 0.4673
data$P.Att.2nd.1out. = 0.5927
data$P.Att.2nd.2out. = 0.9079
data$P.Safe.2nd.= 0.96

prob= data[, c(1,30:43)]

return(prob)
}

prob.batting=prob.batting(teams.batting)

# End of prob function

# trans function to compute the transition matrix and expected runs matrix
trans.batting= function(prob){
  A0=
matrix(data=c(prob.batting$P.HR.[i],prob.batting$P.BB.HBP.[i]+prob.batting$P.1B.[i]+prob.batting$P.E.[i],prob.batting$P.2B.[i],prob.batting$P
.3B.[i],0,0,0,0,

prob.batting$P.HR.[i],0,0,prob.batting$P.3B.[i],prob.batting$P.BB.HBP.[i]+prob.batting$P.1B.[i]+prob.batting$P.E.[i],0,prob.batting$P.2B.[i],0,

prob.batting$P.HR.[i],prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.0out.[i]*prob.batting$P.Safe.2nd.[i],prob.batting$P.2B.[i],prob.batting$P.3B.
[i],prob.batting$P.BB.HBP.[i],prob.batting$P.1B.[i]*(1-prob.batting$P.Att.2nd.0out.[i])
                +prob.batting$P.E.[i],0,0,

prob.batting$P.HR.[i],prob.batting$P.1B.[i]+prob.batting$P.E.[i],prob.batting$P.2B.[i],prob.batting$P.3B.[i],0,prob.batting$P.BB.HBP.[i],0,0,

prob.batting$P.HR.[i],0,0,prob.batting$P.3B.[i],prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.0out.[i]*prob.batting$P.Safe.2nd.[i],0,prob.batting$
P.2B.[i],prob.batting$P.BB.HBP.[i]+prob.batting$P.1B.[i]*(1-prob.batting$P.Att.2nd.0out.
                [i])+prob.batting$P.E.[i],

prob.batting$P.HR.[i],0,0,prob.batting$P.3B.[i],prob.batting$P.1B.[i]+prob.batting$P.E.[i],0,prob.batting$P.2B.[i],prob.batting$P.BB.HBP.[i],

prob.batting$P.HR.[i],prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.0out.[i]*prob.batting$P.Safe.2nd.[i],prob.batting$P.2B.[i],prob.batting$P.3B.
[i],0,prob.batting$P.1B.[i]*(1-prob.batting$P.Att.2nd.0out.[i])+prob.batting$P.E.[i],0,prob
                $P.BB.HBP.[i],

prob.batting$P.HR.[i],0,0,prob.batting$P.3B.[i],prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.0out.[i]*prob.batting$P.Safe.2nd.[i],0,prob.batting$
```

```
P.2B.[i],prob.batting$P.BB.HBP.[i]+prob.batting$P.1B.[i]*(1-prob.batting$P.Att.2nd.0out.
        [i])+prob.batting$P.E.[i]),
      nrow=8, ncol=8, byrow=TRUE)

 A1=
matrix(data=c(prob.batting$P.HR.[i],prob.batting$P.BB.HBP.[i]+prob.batting$P.1B.[i]+prob.batting$P.E.[i],prob.batting$P.2B.[i],prob.batting$P
.3B.[i],0,0,0,0,

prob.batting$P.HR.[i],0,0,prob.batting$P.3B.[i],prob.batting$P.BB.HBP.[i]+prob.batting$P.1B.[i]+prob.batting$P.E.[i],0,prob.batting$P.2B.[i],0,

prob.batting$P.HR.[i],prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.1out.[i]*prob.batting$P.Safe.2nd.[i],prob.batting$P.2B.[i],prob.batting$P.3B.
[i],prob.batting$P.BB.HBP.[i],prob.batting$P.1B.[i]*(1-prob.batting$P.Att.2nd.1out.[i])
        +prob.batting$P.E.[i],0,0,

prob.batting$P.HR.[i],prob.batting$P.1B.[i]+prob.batting$P.E.[i],prob.batting$P.2B.[i],prob.batting$P.3B.[i],0,prob.batting$P.BB.HBP.[i],0,0,

prob.batting$P.HR.[i],0,0,prob.batting$P.3B.[i],prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.1out.[i]*prob.batting$P.Safe.2nd.[i],0,prob.batting$
P.2B.[i],prob.batting$P.BB.HBP.[i]+prob.batting$P.1B.[i]*(1-prob.batting$P.Att.2nd.1out.
        [i])+prob.batting$P.E.[i],

prob.batting$P.HR.[i],0,0,prob.batting$P.3B.[i],prob.batting$P.1B.[i]+prob.batting$P.E.[i],0,prob.batting$P.2B.[i],prob.batting$P.BB.HBP.[i],

prob.batting$P.HR.[i],prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.1out.[i]*prob.batting$P.Safe.2nd.[i],prob.batting$P.2B.[i],prob.batting$P.3B.
[i],0,prob.batting$P.1B.[i]*(1-prob.batting$P.Att.2nd.1out.[i])+prob.batting$P.E.[i],0,prob
        $P.BB.HBP.[i],

prob.batting$P.HR.[i],0,0,prob.batting$P.3B.[i],prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.1out.[i]*prob.batting$P.Safe.2nd.[i],0,prob.batting$
P.2B.[i],prob.batting$P.BB.HBP.[i]+prob.batting$P.1B.[i]*(1-prob.batting$P.Att.2nd.1out.
        [i])+prob.batting$P.E.[i]),
      nrow=8, ncol=8, byrow=TRUE)


 A2=
matrix(data=c(prob.batting$P.HR.[i],prob.batting$P.BB.HBP.[i]+prob.batting$P.1B.[i]+prob.batting$P.E.[i],prob.batting$P.2B.[i],prob.batting$P
.3B.[i],0,0,0,0,

prob.batting$P.HR.[i],0,0,prob.batting$P.3B.[i],prob.batting$P.BB.HBP.[i]+prob.batting$P.1B.[i]+prob.batting$P.E.[i],0,prob.batting$P.2B.[i],0,

prob.batting$P.HR.[i],prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.2out.[i]*prob.batting$P.Safe.2nd.[i],prob.batting$P.2B.[i],prob.batting$P.3B.
[i],prob.batting$P.BB.HBP.[i],prob.batting$P.1B.[i]*(1-prob.batting$P.Att.2nd.2out.[i])
        +prob.batting$P.E.[i],0,0,

prob.batting$P.HR.[i],prob.batting$P.1B.[i]+prob.batting$P.E.[i],prob.batting$P.2B.[i],prob.batting$P.3B.[i],0,prob.batting$P.BB.HBP.[i],0,0,

prob.batting$P.HR.[i],0,0,prob.batting$P.3B.[i],prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.2out.[i]*prob.batting$P.Safe.2nd.[i],0,prob.batting$
P.2B.[i],prob.batting$P.BB.HBP.[i]+prob.batting$P.1B.[i]*(1-prob.batting$P.Att.2nd.2out.
        [i])+prob.batting$P.E.[i],

prob.batting$P.HR.[i],0,0,prob.batting$P.3B.[i],prob.batting$P.1B.[i]+prob.batting$P.E.[i],0,prob.batting$P.2B.[i],prob.batting$P.BB.HBP.[i],

prob.batting$P.HR.[i],prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.2out.[i]*prob.batting$P.Safe.2nd.[i],prob.batting$P.2B.[i],prob.batting$P.3B.
[i],0,prob.batting$P.1B.[i]*(1-prob.batting$P.Att.2nd.2out.[i])+prob.batting$P.E.[i],0,prob
        $P.BB.HBP.[i],

prob.batting$P.HR.[i],0,0,prob.batting$P.3B.[i],prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.2out.[i]*prob.batting$P.Safe.2nd.[i],0,prob.batting$
P.2B.[i],prob.batting$P.BB.HBP.[i]+prob.batting$P.1B.[i]*(1-prob.batting$P.Att.2nd.2out.
        [i])+prob.batting$P.E.[i]),
      nrow=8, ncol=8, byrow=TRUE)



 # creating B1 matrix: trans from 0 outs to 1 out
 B1= matrix(data=c(prob.batting$P.Out.[i],0,0,0,0,0,0,0,
         0,prob.batting$P.1Out.[i],0,0,0,0,0,0,
         0,prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.0out.[i]*(1-prob.batting$P.Safe.2nd.[i]),prob.batting$P.Out.[i],0,0,0,0,0,
         prob.batting$P.SF.[i],0,0,prob.batting$P.Out.[i]- prob.batting$P.SF.[i],0,0,0,0,
                0,0,0,0,prob.batting$P.1Out.[i]+prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.0out[i]*(1-prob.batting$P.Safe.2nd.[i]),0,0,0,
                0,prob.batting$P.SF.[i],0,0,0,prob.batting$P.1Out.[i]- prob.batting$P.SF.[i],0,0,

0,prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.0out.[i]*(1-prob.batting$P.Safe.2nd.[i]),prob.batting$P.SF.[i],0,0,0,prob.batting$P.Out.[i]-
prob.batting$P.SF.[i],0,

0,0,0,0,prob.batting$P.SF.[i]+prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.0out.[i]*(1-prob.batting$P.Safe.2nd.[i]),0,0,prob.batting$P.1Out.[i]-
prob.batting$P.SF.[i]),
          nrow=8, ncol=8, byrow=TRUE)

  # creating B2 matrix: trans from 1 out to 2 outs
  B2= matrix(data=c(prob.batting$P.Out.[i],0,0,0,0,0,0,0,
                0,prob.batting$P.1Out.[i],0,0,0,0,0,0,
                0,prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.1out.[i]*(1-prob.batting$P.Safe.2nd.[i]),prob.batting$P.Out.[i],0,0,0,0,0,
                prob.batting$P.SF.[i],0,0,prob.batting$P.Out.[i]- prob.batting$P.SF.[i],0,0,0,0,
                0,0,0,0,prob.batting$P.1Out.[i]+prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.1out[i]*(1-prob.batting$P.Safe.2nd.[i]),0,0,0,
                0,prob.batting$P.SF.[i],0,0,0,prob.batting$P.1Out.[i]- prob.batting$P.SF.[i],0,0,

0,prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.1out.[i]*(1-prob.batting$P.Safe.2nd.[i]),prob.batting$P.SF.[i],0,0,0,prob.batting$P.Out.[i]-
prob.batting$P.SF.[i],0,

0,0,0,0,prob.batting$P.SF.[i]+prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.1out.[i]*(1-prob.batting$P.Safe.2nd.[i]),0,0,prob.batting$P.1Out.[i]-
prob.batting$P.SF.[i]),
          nrow=8, ncol=8, byrow=TRUE)

  # creating C2 matrix: trans from 0 outs to 2 outs
  C2= matrix(data=c(0,0,0,0,0,0,0,0,
                prob.batting$P.DP.[i],0,0,0,0,0,0,0,
                0,0,0,0,0,0,0,0,
                0,0,0,0,0,0,0,0,
                0,0,0,prob.batting$P.DP.[i],0,0,0,0,
                prob.batting$P.DP.[i],0,0,0,0,0,0,0,
                0,0,0,0,0,0,0,0,
                0,0,0,prob.batting$P.DP.[i],0,0,0,0),
          nrow=8, ncol=8, byrow=TRUE)

  # absorbing states
  D1= matrix(0, nrow=8, ncol=1)
  D2= matrix(data=c(0,prob.batting$P.DP.[i],0,0,prob.batting$P.DP.[i],prob.batting$P.DP.[i],0,prob.batting$P.DP.[i]), nrow=8, ncol=1)
  D3= matrix(c(prob.batting$P.Out.[i],
                                                prob.batting$P.Out.[i],

prob.batting$P.Out.[i]+prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.2out[i]*(1-prob.batting$P.Safe.2nd.[i]),
                                                prob.batting$P.Out.[i],

prob.batting$P.Out.[i]+prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.2out[i]*(1-prob.batting$P.Safe.2nd.[i]),
                                         prob.batting$P.Out.[i],
                                         prob.batting$P.Out.[i]+prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.2out[i]*(1-prob.batting$P.Safe.2nd.[i]),

prob.batting$P.Out.[i]+prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.2out[i]*(1-prob.batting$P.Safe.2nd.[i])),
                              nrow=8, ncol=1)
               D4= matrix(1, nrow=1, ncol=1)

  abs0= matrix(c(0,0,0,0,0,0,0,0), nrow=1, ncol=8, byrow=TRUE)
  abs1= matrix(c(0,0,0,0,0,0,0,0), nrow=1, ncol=8, byrow=TRUE)
  abs2= matrix(c(0,0,0,0,0,0,0,0), nrow=1, ncol=8, byrow=TRUE)

  # zero matrices
  zero= matrix(0, nrow=8, ncol=8)

  # creating overall transisition matrix by combining above matrices
  trans= matrix(c(rbind(A0,zero,zero,abs0),rbind(B1,A1,zero,abs1),rbind(C2,B2,A2,abs2), rbind(D1,D2,D3,D4)),nrow=25, ncol=25,
             dimnames= list(c("(0,0)","(1,0)","(2,0)","(3,0)","(12,0)","(13,0)","(23,0)",
                           "(123,0)","(0,1)","(1,1)","(2,1)","(3,1)","(12,1)","(13,1)","(23,1)","(123,1)",

"(0,2)","(1,2)","(2,2)","(3,2)","(12,2)","(13,2)","(23,2)","(123,2)","(x,3)"),c("(0,0)","(1,0)","(2,0)","(3,0)","(12,0)","(13,0)","(23,0)",
                    "(123,0)","(0,1)","(1,1)","(2,1)","(3,1)","(12,1)","(13,1)","(23,1)","(123,1)",
                    "(0,2)","(1,2)","(2,2)","(3,2)","(12,2)","(13,2)","(23,2)","(123,2)","(x,3)")))

  R1=matrix(c(0,0,0,1*prob.batting$P.HR.[i],0,
         0,0,2*prob.batting$P.HR.[i],1*prob.batting$P.3B.[i],0,

0,0,2*prob.batting$P.HR.[i],1*(prob.batting$P.2B.[i]+prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.0out.[i]*prob.batting$P.Safe.2nd.[i]),0,

0,0,2*prob.batting$P.HR.[i],1*(prob.batting$P.3B.[i]+prob.batting$P.2B.[i]+prob.batting$P.1B.[i]+prob.batting$P.E.[i]+prob.batting$P.SF.[i]),0,

0,3*prob.batting$P.HR.[i],2*prob.batting$P.3B.[i],1*(prob.batting$P.2B.[i]+prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.0out.[i]*prob.batting$
P.Safe.2nd.[i]),0,
         0,3*prob.batting$P.HR.[i],2*prob.batting$P.3B.[i],1*(prob.batting$P.1B.[i]+prob.batting$P.E.[i]+prob.batting$P.SF.[i]),0,

0,3*prob.batting$P.HR.[i],2*(prob.batting$P.2B.[i]+prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.0out.[i]*prob.batting$P.Safe.2nd.[i]),1*(prob.b
atting$P.1B.[i]*prob.batting$P.Att.2nd.0out.[i]*(1-prob.batting$P.Safe.2nd.[i])+prob.batting$P.1B.[i]*(1-prob.batting$P.Att.2nd.0out.[i])+prob.b
atting$P.E.[i]+prob.batting$P.SF.[i]),0,

4*prob.batting$P.HR.[i],3*prob.batting$P.3B.[i],2*(prob.batting$P.2B.[i]+prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.0out.[i]*prob.batting$P.
Safe.2nd.[i]),
1*(prob.batting$P.BB.HBP.[i]+prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.0out.[i]*(1-prob.batting$P.Safe.2nd.[i])+prob.batting$P.1B.[i]*(1-p
rob.batting$P.Att.2nd.0out.[i])+prob.batting$P.E.[i]+prob.batting$P.SF.[i]),0),
        ncol=5,byrow=TRUE)

  R2=matrix(c(0,0,0,1*prob.batting$P.HR.[i],0,
         0,0,2*prob.batting$P.HR.[i],1*prob.batting$P.3B.[i],0,

0,0,2*prob.batting$P.HR.[i],1*(prob.batting$P.2B.[i]+prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.1out.[i]*prob.batting$P.Safe.2nd.[i]),0,

0,0,2*prob.batting$P.HR.[i],1*(prob.batting$P.3B.[i]+prob.batting$P.2B.[i]+prob.batting$P.1B.[i]+prob.batting$P.E.[i]+prob.batting$P.SF.[i]),0,

0,3*prob.batting$P.HR.[i],2*prob.batting$P.3B.[i],1*(prob.batting$P.2B.[i]+prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.1out.[i]*prob.batting$
P.Safe.2nd.[i]),0,
         0,3*prob.batting$P.HR.[i],2*prob.batting$P.3B.[i],1*(prob.batting$P.1B.[i]+prob.batting$P.E.[i]+prob.batting$P.SF.[i]),0,

0,3*prob.batting$P.HR.[i],2*(prob.batting$P.2B.[i]+prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.1out.[i]*prob.batting$P.Safe.2nd.[i]),1*(prob.b
atting$P.1B.[i]*prob.batting$P.Att.2nd.1out.[i]*(1-prob.batting$P.Safe.2nd.[i])+prob.batting$P.1B.[i]*(1-prob.batting$P.Att.2nd.1out.[i])+prob.b
atting$P.E.[i]+prob.batting$P.SF.[i]),0,

4*prob.batting$P.HR.[i],3*prob.batting$P.3B.[i],2*(prob.batting$P.2B.[i]+prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.1out.[i]*prob.batting$P.
Safe.2nd.[i]),
1*(prob.batting$P.BB.HBP.[i]+prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.1out.[i]*(1-prob.batting$P.Safe.2nd.[i])+prob.batting$P.1B.[i]*(1-p
rob.batting$P.Att.2nd.1out.[i])+prob.batting$P.E.[i]+prob.batting$P.SF.[i]),0),
        ncol=5,byrow=TRUE)

  R3=matrix(c(0,0,0,1*prob.batting$P.HR.[i],0,
         0,0,2*prob.batting$P.HR.[i],1*prob.batting$P.3B.[i],0,

0,0,2*prob.batting$P.HR.[i],1*(prob.batting$P.2B.[i]+prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.2out.[i]*prob.batting$P.Safe.2nd.[i]),0,
         0,0,2*prob.batting$P.HR.[i],1*(prob.batting$P.3B.[i]+prob.batting$P.2B.[i]+prob.batting$P.1B.[i]+prob.batting$P.E.[i]),0,

0,3*prob.batting$P.HR.[i],2*prob.batting$P.3B.[i],1*(prob.batting$P.2B.[i]+prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.2out.[i]*prob.batting$
P.Safe.2nd.[i]),0,
         0,3*prob.batting$P.HR.[i],2*prob.batting$P.3B.[i],1*(prob.batting$P.1B.[i]+prob.batting$P.E.[i]),0,

0,3*prob.batting$P.HR.[i],2*(prob.batting$P.2B.[i]+prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.2out.[i]*prob.batting$P.Safe.2nd.[i]),1*(prob.b
atting$P.1B.[i]*prob.batting$P.Att.2nd.2out.[i]*(1-prob.batting$P.Safe.2nd.[i])+prob.batting$P.1B.[i]*(1-prob.batting$P.Att.2nd.2out.[i])+prob.b
atting$P.E.[i]),0,

4*prob.batting$P.HR.[i],3*prob.batting$P.3B.[i],2*(prob.batting$P.2B.[i]+prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.2out.[i]*prob.batting$P.
Safe.2nd.[i]),
1*(prob.batting$P.BB.HBP.[i]+prob.batting$P.1B.[i]*prob.batting$P.Att.2nd.2out.[i]*(1-prob.batting$P.Safe.2nd.[i])+prob.batting$P.1B.[i]*(1-p
rob.batting$P.Att.2nd.2out.[i])+prob.batting$P.E.[i]),0),
        ncol=5,byrow=TRUE)

  Rscore= matrix(rbind(R1,R2,R3),nrow=24,ncol=5)
  eruns=matrix(rowSums(Rscore), nrow=24, dimnames= list(c("(0,0)","(1,0)","(2,0)","(3,0)","(12,0)","(13,0)","(23,0)",
                              "(123,0)","(0,1)","(1,1)","(2,1)","(3,1)","(12,1)","(13,1)","(23,1)","(123,1)",
                              "(0,2)","(1,2)","(2,2)","(3,2)","(12,2)","(13,2)","(23,2)","(123,2)"),c("Exp Runs for Half-Inn")))

  list(trans,eruns)
}
# End of trans function

# Start of script

nplayers.batting = nrow(prob.batting)
trans.store.batting=array(NaN,c(25,25,nplayers.batting),dimnames= list(c("(0,0)","(1,0)","(2,0)","(3,0)","(12,0)","(13,0)","(23,0)",
                              "(123,0)","(0,1)","(1,1)","(2,1)","(3,1)","(12,1)","(13,1)","(23,1)","(123,1)",

"(0,2)","(1,2)","(2,2)","(3,2)","(12,2)","(13,2)","(23,2)","(123,2)","(x,3)"),c("(0,0)","(1,0)","(2,0)","(3,0)","(12,0)","(13,0)","(23,0)",
                              "(123,0)","(0,1)","(1,1)","(2,1)","(3,1)","(12,1)","(13,1)","(23,1)","(123,1)",
                              "(0,2)","(1,2)","(2,2)","(3,2)","(12,2)","(13,2)","(23,2)","(123,2)","(x,3)")))
R.store.batting=array(NaN,c(24,5,nplayers.batting))
Exp.runs.store.batting=matrix(NaN,nrow=24,ncol=nplayers.batting)

for (i in 1: nplayers.batting){
temp.batting=trans.batting(prob.batting)
temp.trans.batting=temp.batting[[1]]
temp.eruns.batting=temp.batting[[2]]

# creating E: rowsums(E)= expected number of batters at each starting state for the inning
I=diag(24)
Q=temp.trans.batting[-25,-25]
E=solve(I-Q)

# Expected Runs for rest of inning at starting state: mult by 9 for full game
Exp.Runs= E%*%temp.eruns.batting
Nine.inn=Exp.Runs*9
Nine.inn

trans.store.batting[,,i]=temp.trans.batting
R.store.batting[,,i]=temp.eruns.batting
Exp.runs.store.batting[,i]=Nine.inn
}


compare.batting=cbind(teams.batting[,c(1,4)],Exp.runs.store.batting[1,], teams.batting$R/162, teams.batting$R)
compare.batting$Exp.runs.162.batting= Exp.runs.store.batting[1,]*162
compare.batting$Percent.change= round(((compare.batting$Exp.runs.162.batting-teams.batting$R)/compare.batting$Exp.runs.162.batting)*100,
3)
mean.batting=mean(abs(compare.batting$Percent.change))
```

Pitching Code

```
teams.pitching= read.csv("MLBPitching2015.csv", header=TRUE, nrow=31)
```

```
# Prob function to convert raw data to probabilities
prob.pitching= function(data){
data$P.BB.HBP.= (data$BB+ data$HBP)/data$PA
data$P.1B.= (data$H- data$X2B-data$X3B- data$HR)/data$PA
data$P.2B.= data$X2B/data$PA
data$P.3B.= data$X3B/data$PA
data$P.HR.= data$HR/data$PA
data$P.DP.= data$GDP/data$PA
data$P.E.= (data$AB-data$SO)*(1-0.984)/data$PA
data$P.Out.= 1-(data$P.BB.HBP.+data$P.1B.+data$P.2B.+data$P.3B.+data$P.HR.+data$P.E.)
data$P.1Out.= data$P.Out.-data$P.DP.
data$P.SF.= data$SF/data$PA

# Extrabase is the probability of advancing from 2nd base to home on a single
data$P.Att.2nd.0out. = 0.4673
data$P.Att.2nd.1out. = 0.5927
data$P.Att.2nd.2out. = 0.9079
data$P.Safe.2nd.= 0.96

prob= data[, c(1,28:41)]

return(prob)
}

prob.pitching=prob.pitching(teams.pitching)

# End of prob function

# trans function to compute the transition matrix and expected runs matrix
trans.pitching= function(prob){
  A0=
matrix(data=c(prob.pitching$P.HR.[i],prob.pitching$P.BB.HBP.[i]+prob.pitching$P.1B.[i]+prob.pitching$P.E.[i],prob.pitching$P.2B.[i],prob.pit
ching$P.3B.[i],0,0,0,0,

prob.pitching$P.HR.[i],0,0,prob.pitching$P.3B.[i],prob.pitching$P.BB.HBP.[i]+prob.pitching$P.1B.[i]+prob.pitching$P.E.[i],0,prob.pitching$P.2
B.[i],0,

prob.pitching$P.HR.[i],prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.0out.[i]*prob.pitching$P.Safe.2nd.[i],prob.pitching$P.2B.[i],prob.pitchin
g$P.3B.[i],prob.pitching$P.BB.HBP.[i],prob.pitching$P.1B.[i]*(1-prob.pitching$P.Att.2nd.0out.[i])
                              +prob.pitching$P.E.[i],0,0,

prob.pitching$P.HR.[i],prob.pitching$P.1B.[i]+prob.pitching$P.E.[i],prob.pitching$P.2B.[i],prob.pitching$P.3B.[i],0,prob.pitching$P.BB.HBP.[i]
,0,0,

prob.pitching$P.HR.[i],0,0,prob.pitching$P.3B.[i],prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.0out.[i]*prob.pitching$P.Safe.2nd.[i],0,prob.pi
tching$P.2B.[i],prob.pitching$P.BB.HBP.[i]+prob.pitching$P.1B.[i]*(1-prob.pitching$P.Att.2nd.0out.
                              [i])+prob.pitching$P.E.[i],

prob.pitching$P.HR.[i],0,0,prob.pitching$P.3B.[i],prob.pitching$P.1B.[i]+prob.pitching$P.E.[i],0,prob.pitching$P.2B.[i],prob.pitching$P.BB.HB
P.[i],

prob.pitching$P.HR.[i],prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.0out.[i]*prob.pitching$P.Safe.2nd.[i],prob.pitching$P.2B.[i],prob.pitchin
g$P.3B.[i],0,prob.pitching$P.1B.[i]*(1-prob.pitching$P.Att.2nd.0out.[i])+prob.pitching$P.E.[i],0,prob
                              $P.BB.HBP.[i],

prob.pitching$P.HR.[i],0,0,prob.pitching$P.3B.[i],prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.0out.[i]*prob.pitching$P.Safe.2nd.[i],0,prob.pi
tching$P.2B.[i],prob.pitching$P.BB.HBP.[i]+prob.pitching$P.1B.[i]*(1-prob.pitching$P.Att.2nd.0out.
                              [i])+prob.pitching$P.E.[i]),
          nrow=8, ncol=8, byrow=TRUE)

  A1=
matrix(data=c(prob.pitching$P.HR.[i],prob.pitching$P.BB.HBP.[i]+prob.pitching$P.1B.[i]+prob.pitching$P.E.[i],prob.pitching$P.2B.[i],prob.pit
ching$P.3B.[i],0,0,0,0,
prob.pitching$P.HR.[i],0,0,prob.pitching$P.3B.[i],prob.pitching$P.BB.HBP.[i]+prob.pitching$P.1B.[i]+prob.pitching$P.E.[i],0,prob.pitching$P.2
B.[i],0,

prob.pitching$P.HR.[i],prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.1out.[i]*prob.pitching$P.Safe.2nd.[i],prob.pitching$P.2B.[i],prob.pitchin
g$P.3B.[i],prob.pitching$P.BB.HBP.[i],prob.pitching$P.1B.[i]*(1-prob.pitching$P.Att.2nd.1out.[i])
                        +prob.pitching$P.E.[i],0,0,

prob.pitching$P.HR.[i],prob.pitching$P.1B.[i]+prob.pitching$P.E.[i],prob.pitching$P.2B.[i],prob.pitching$P.3B.[i],0,prob.pitching$P.BB.HBP.[i]
,0,0,

prob.pitching$P.HR.[i],0,0,prob.pitching$P.3B.[i],prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.1out.[i]*prob.pitching$P.Safe.2nd.[i],0,prob.pi
tching$P.2B.[i],prob.pitching$P.BB.HBP.[i]+prob.pitching$P.1B.[i]*(1-prob.pitching$P.Att.2nd.1out.
                        [i])+prob.pitching$P.E.[i],

prob.pitching$P.HR.[i],0,0,prob.pitching$P.3B.[i],prob.pitching$P.1B.[i]+prob.pitching$P.E.[i],0,prob.pitching$P.2B.[i],prob.pitching$P.BB.HB
P.[i],

prob.pitching$P.HR.[i],prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.1out.[i]*prob.pitching$P.Safe.2nd.[i],prob.pitching$P.2B.[i],prob.pitchin
g$P.3B.[i],0,prob.pitching$P.1B.[i]*(1-prob.pitching$P.Att.2nd.1out.[i])+prob.pitching$P.E.[i],0,prob
                        $P.BB.HBP.[i],

prob.pitching$P.HR.[i],0,0,prob.pitching$P.3B.[i],prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.1out.[i]*prob.pitching$P.Safe.2nd.[i],0,prob.pi
tching$P.2B.[i],prob.pitching$P.BB.HBP.[i]+prob.pitching$P.1B.[i]*(1-prob.pitching$P.Att.2nd.1out.
                        [i])+prob.pitching$P.E.[i]),
         nrow=8, ncol=8, byrow=TRUE)


 A2=
matrix(data=c(prob.pitching$P.HR.[i],prob.pitching$P.BB.HBP.[i]+prob.pitching$P.1B.[i]+prob.pitching$P.E.[i],prob.pitching$P.2B.[i],prob.pit
ching$P.3B.[i],0,0,0,0,

prob.pitching$P.HR.[i],0,0,prob.pitching$P.3B.[i],prob.pitching$P.BB.HBP.[i]+prob.pitching$P.1B.[i]+prob.pitching$P.E.[i],0,prob.pitching$P.2
B.[i],0,

prob.pitching$P.HR.[i],prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.2out.[i]*prob.pitching$P.Safe.2nd.[i],prob.pitching$P.2B.[i],prob.pitchin
g$P.3B.[i],prob.pitching$P.BB.HBP.[i],prob.pitching$P.1B.[i]*(1-prob.pitching$P.Att.2nd.2out.[i])
                        +prob.pitching$P.E.[i],0,0,

prob.pitching$P.HR.[i],prob.pitching$P.1B.[i]+prob.pitching$P.E.[i],prob.pitching$P.2B.[i],prob.pitching$P.3B.[i],0,prob.pitching$P.BB.HBP.[i]
,0,0,

prob.pitching$P.HR.[i],0,0,prob.pitching$P.3B.[i],prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.2out.[i]*prob.pitching$P.Safe.2nd.[i],0,prob.pi
tching$P.2B.[i],prob.pitching$P.BB.HBP.[i]+prob.pitching$P.1B.[i]*(1-prob.pitching$P.Att.2nd.2out.
                        [i])+prob.pitching$P.E.[i],

prob.pitching$P.HR.[i],0,0,prob.pitching$P.3B.[i],prob.pitching$P.1B.[i]+prob.pitching$P.E.[i],0,prob.pitching$P.2B.[i],prob.pitching$P.BB.HB
P.[i],

prob.pitching$P.HR.[i],prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.2out.[i]*prob.pitching$P.Safe.2nd.[i],prob.pitching$P.2B.[i],prob.pitchin
g$P.3B.[i],0,prob.pitching$P.1B.[i]*(1-prob.pitching$P.Att.2nd.2out.[i])+prob.pitching$P.E.[i],0,prob
                        $P.BB.HBP.[i],

prob.pitching$P.HR.[i],0,0,prob.pitching$P.3B.[i],prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.2out.[i]*prob.pitching$P.Safe.2nd.[i],0,prob.pi
tching$P.2B.[i],prob.pitching$P.BB.HBP.[i]+prob.pitching$P.1B.[i]*(1-prob.pitching$P.Att.2nd.2out.
                        [i])+prob.pitching$P.E.[i]),
         nrow=8, ncol=8, byrow=TRUE)


 # creating B1 matrix: trans from 0 outs to 1 out
 B1= matrix(data=c(prob.pitching$P.Out.[i],0,0,0,0,0,0,0,
            0,prob.pitching$P.1Out.[i],0,0,0,0,0,0,
            0,prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.0out.[i]*(1-prob.pitching$P.Safe.2nd.[i]),prob.pitching$P.Out.[i],0,0,0,0,0,
            prob.pitching$P.SF.[i],0,0,prob.pitching$P.Out.[i]- prob.pitching$P.SF.[i],0,0,0,0,
            0,0,0,0,prob.pitching$P.1Out.[i]+ prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.0out[i]*(1-prob.pitching$P.Safe.2nd.[i]),0,0,0,
                0,prob.pitching$P.SF.[i],0,0,0,prob.pitching$P.1Out.[i]- prob.pitching$P.SF.[i],0,0,

0,prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.0out.[i]*(1-prob.pitching$P.Safe.2nd.[i]),prob.pitching$P.SF.[i],0,0,0,prob.pitching$P.Out.[i]-
prob.pitching$P.SF.[i],0,

0,0,0,0,prob.pitching$P.SF.[i]+prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.0out.[i]*(1-prob.pitching$P.Safe.2nd.[i]),0,0,prob.pitching$P.1Ou
t.[i]- prob.pitching$P.SF.[i]),
          nrow=8, ncol=8, byrow=TRUE)

  # creating B2 matrix: trans from 1 out to 2 outs
  B2= matrix(data=c(prob.pitching$P.Out.[i],0,0,0,0,0,0,0,
                0,prob.pitching$P.1Out.[i],0,0,0,0,0,0,
                0,prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.1out.[i]*(1-prob.pitching$P.Safe.2nd.[i]),prob.pitching$P.Out.[i],0,0,0,0,0,
                prob.pitching$P.SF.[i],0,0,prob.pitching$P.Out.[i]- prob.pitching$P.SF.[i],0,0,0,0,
                0,0,0,0,prob.pitching$P.1Out.[i]+prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.1out[i]*(1-prob.pitching$P.Safe.2nd.[i]),0,0,0,
                0,prob.pitching$P.SF.[i],0,0,0,prob.pitching$P.1Out.[i]- prob.pitching$P.SF.[i],0,0,

0,prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.1out.[i]*(1-prob.pitching$P.Safe.2nd.[i]),prob.pitching$P.SF.[i],0,0,0,prob.pitching$P.Out.[i]-
prob.pitching$P.SF.[i],0,

0,0,0,0,prob.pitching$P.SF.[i]+prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.1out.[i]*(1-prob.pitching$P.Safe.2nd.[i]),0,0,prob.pitching$P.1Ou
t.[i]- prob.pitching$P.SF.[i]),
          nrow=8, ncol=8, byrow=TRUE)

  # creating C2 matrix: trans from 0 outs to 2 outs
  C2= matrix(data=c(0,0,0,0,0,0,0,0,
                prob.pitching$P.DP.[i],0,0,0,0,0,0,0,
                0,0,0,0,0,0,0,0,
                0,0,0,0,0,0,0,0,
                0,0,0,prob.pitching$P.DP.[i],0,0,0,0,
                prob.pitching$P.DP.[i],0,0,0,0,0,0,0,
                0,0,0,0,0,0,0,0,
                0,0,0,prob.pitching$P.DP.[i],0,0,0,0),
          nrow=8, ncol=8, byrow=TRUE)

  # absorbing states
  D1= matrix(0, nrow=8, ncol=1)
  D2= matrix(data=c(0,prob.pitching$P.DP.[i],0,0,prob.pitching$P.DP.[i],prob.pitching$P.DP.[i],0,prob.pitching$P.DP.[i]), nrow=8, ncol=1)
  D3= matrix(c(prob.pitching$P.Out.[i],
                                        prob.pitching$P.Out.[i],

prob.pitching$P.Out.[i]+prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.2out[i]*(1-prob.pitching$P.Safe.2nd.[i]),
                                        prob.pitching$P.Out.[i],

prob.pitching$P.Out.[i]+prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.2out[i]*(1-prob.pitching$P.Safe.2nd.[i]),
                                  prob.pitching$P.Out.[i],

prob.pitching$P.Out.[i]+prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.2out[i]*(1-prob.pitching$P.Safe.2nd.[i]),

prob.pitching$P.Out.[i]+prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.2out[i]*(1-prob.pitching$P.Safe.2nd.[i])),
                        nrow=8, ncol=1)
  D4= matrix(1, nrow=1, ncol=1)

  abs0= matrix(c(0,0,0,0,0,0,0,0), nrow=1, ncol=8, byrow=TRUE)
  abs1= matrix(c(0,0,0,0,0,0,0,0), nrow=1, ncol=8, byrow=TRUE)
  abs2= matrix(c(0,0,0,0,0,0,0,0), nrow=1, ncol=8, byrow=TRUE)

  # zero matrices
  zero= matrix(0, nrow=8, ncol=8)

  # creating overall transisition matrix by combining above matrices
  trans= matrix(c(rbind(A0,zero,zero,abs0),rbind(B1,A1,zero,abs1),rbind(C2,B2,A2,abs2), rbind(D1,D2,D3,D4)),nrow=25, ncol=25,
              dimnames= list(c("(0,0)","(1,0)","(2,0)","(3,0)","(12,0)","(13,0)","(23,0)",
                        "(123,0)","(0,1)","(1,1)","(2,1)","(3,1)","(12,1)","(13,1)","(23,1)","(123,1)",

"(0,2)","(1,2)","(2,2)","(3,2)","(12,2)","(13,2)","(23,2)","(123,2)","(x,3)"),c("(0,0)","(1,0)","(2,0)","(3,0)","(12,0)","(13,0)","(23,0)",
                                        "(123,0)","(0,1)","(1,1)","(2,1)","(3,1)","(12,1)","(13,1)","(23,1)","(123,1)",
                                        "(0,2)","(1,2)","(2,2)","(3,2)","(12,2)","(13,2)","(23,2)","(123,2)","(x,3)")))

  R1=matrix(c(0,0,0,1*prob.pitching$P.HR.[i],0,
         0,0,2*prob.pitching$P.HR.[i],1*prob.pitching$P.3B.[i],0,

0,0,2*prob.pitching$P.HR.[i],1*(prob.pitching$P.2B.[i]+prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.0out.[i]*prob.pitching$P.Safe.2nd.[i]),0
,

0,0,2*prob.pitching$P.HR.[i],1*(prob.pitching$P.3B.[i]+prob.pitching$P.2B.[i]+prob.pitching$P.1B.[i]+prob.pitching$P.E.[i]+prob.pitching$P.S
F.[i]),0,

0,3*prob.pitching$P.HR.[i],2*prob.pitching$P.3B.[i],1*(prob.pitching$P.2B.[i]+prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.0out.[i]*prob.pit
ching$P.Safe.2nd.[i]),0,
         0,3*prob.pitching$P.HR.[i],2*prob.pitching$P.3B.[i],1*(prob.pitching$P.1B.[i]+prob.pitching$P.E.[i]+prob.pitching$P.SF.[i]),0,

0,3*prob.pitching$P.HR.[i],2*(prob.pitching$P.2B.[i]+prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.0out.[i]*prob.pitching$P.Safe.2nd.[i]),1*(
prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.0out.[i]*(1-prob.pitching$P.Safe.2nd.[i])+prob.pitching$P.1B.[i]*(1-prob.pitching$P.Att.2nd.0ou
t.[i])+prob.pitching$P.E.[i]+prob.pitching$P.SF.[i]),0,

4*prob.pitching$P.HR.[i],3*prob.pitching$P.3B.[i],2*(prob.pitching$P.2B.[i]+prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.0out.[i]*prob.pitc
hing$P.Safe.2nd.[i]),
1*(prob.pitching$P.BB.HBP.[i]+prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.0out.[i]*(1-prob.pitching$P.Safe.2nd.[i])+prob.pitching$P.1B.[i
]*(1-prob.pitching$P.Att.2nd.0out.[i])+prob.pitching$P.E.[i]+prob.pitching$P.SF.[i]),0),
        ncol=5,byrow=TRUE)

  R2=matrix(c(0,0,0,1*prob.pitching$P.HR.[i],0,
         0,0,2*prob.pitching$P.HR.[i],1*prob.pitching$P.3B.[i],0,

0,0,2*prob.pitching$P.HR.[i],1*(prob.pitching$P.2B.[i]+prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.1out.[i]*prob.pitching$P.Safe.2nd.[i]),0
,

0,0,2*prob.pitching$P.HR.[i],1*(prob.pitching$P.3B.[i]+prob.pitching$P.2B.[i]+prob.pitching$P.1B.[i]+prob.pitching$P.E.[i]+prob.pitching$P.S
F.[i]),0,

0,3*prob.pitching$P.HR.[i],2*prob.pitching$P.3B.[i],1*(prob.pitching$P.2B.[i]+prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.1out.[i]*prob.pit
ching$P.Safe.2nd.[i]),0,
         0,3*prob.pitching$P.HR.[i],2*prob.pitching$P.3B.[i],1*(prob.pitching$P.1B.[i]+prob.pitching$P.E.[i]+prob.pitching$P.SF.[i]),0,

0,3*prob.pitching$P.HR.[i],2*(prob.pitching$P.2B.[i]+prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.1out.[i]*prob.pitching$P.Safe.2nd.[i]),1*(
prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.1out.[i]*(1-prob.pitching$P.Safe.2nd.[i])+prob.pitching$P.1B.[i]*(1-prob.pitching$P.Att.2nd.1ou
t.[i])+prob.pitching$P.E.[i]+prob.pitching$P.SF.[i]),0,

4*prob.pitching$P.HR.[i],3*prob.pitching$P.3B.[i],2*(prob.pitching$P.2B.[i]+prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.1out.[i]*prob.pitc
hing$P.Safe.2nd.[i]),
1*(prob.pitching$P.BB.HBP.[i]+prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.1out.[i]*(1-prob.pitching$P.Safe.2nd.[i])+prob.pitching$P.1B.[i
]*(1-prob.pitching$P.Att.2nd.1out.[i])+prob.pitching$P.E.[i]+prob.pitching$P.SF.[i]),0),
        ncol=5,byrow=TRUE)

  R3=matrix(c(0,0,0,1*prob.pitching$P.HR.[i],0,
         0,0,2*prob.pitching$P.HR.[i],1*prob.pitching$P.3B.[i],0,

0,0,2*prob.pitching$P.HR.[i],1*(prob.pitching$P.2B.[i]+prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.2out.[i]*prob.pitching$P.Safe.2nd.[i]),0
,
         0,0,2*prob.pitching$P.HR.[i],1*(prob.pitching$P.3B.[i]+prob.pitching$P.2B.[i]+prob.pitching$P.1B.[i]+prob.pitching$P.E.[i]),0,

0,3*prob.pitching$P.HR.[i],2*prob.pitching$P.3B.[i],1*(prob.pitching$P.2B.[i]+prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.2out.[i]*prob.pit
ching$P.Safe.2nd.[i]),0,
         0,3*prob.pitching$P.HR.[i],2*prob.pitching$P.3B.[i],1*(prob.pitching$P.1B.[i]+prob.pitching$P.E.[i]),0,

0,3*prob.pitching$P.HR.[i],2*(prob.pitching$P.2B.[i]+prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.2out.[i]*prob.pitching$P.Safe.2nd.[i]),1*(
prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.2out.[i]*(1-prob.pitching$P.Safe.2nd.[i])+prob.pitching$P.1B.[i]*(1-prob.pitching$P.Att.2nd.2ou
t.[i])+prob.pitching$P.E.[i]),0,

4*prob.pitching$P.HR.[i],3*prob.pitching$P.3B.[i],2*(prob.pitching$P.2B.[i]+prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.2out.[i]*prob.pitc
hing$P.Safe.2nd.[i]),
1*(prob.pitching$P.BB.HBP.[i]+prob.pitching$P.1B.[i]*prob.pitching$P.Att.2nd.2out.[i]*(1-prob.pitching$P.Safe.2nd.[i])+prob.pitching$P.1B.[i
]*(1-prob.pitching$P.Att.2nd.2out.[i])+prob.pitching$P.E.[i]),0),
          ncol=5,byrow=TRUE)

  Rscore= matrix(rbind(R1,R2,R3),nrow=24,ncol=5)
  eruns=matrix(rowSums(Rscore), nrow=24, dimnames= list(c("(0,0)","(1,0)","(2,0)","(3,0)","(12,0)","(13,0)","(23,0)",
                              "(123,0)","(0,1)","(1,1)","(2,1)","(3,1)","(12,1)","(13,1)","(23,1)","(123,1)",
                              "(0,2)","(1,2)","(2,2)","(3,2)","(12,2)","(13,2)","(23,2)","(123,2)"),c("Exp Runs for Half-Inn")))

  list(trans,eruns)
}
# End of trans function

# Start of script
nplayers.pitching = nrow(prob.pitching)
trans.store.pitching=array(NaN,c(25,25,nplayers.pitching),dimnames= list(c("(0,0)","(1,0)","(2,0)","(3,0)","(12,0)","(13,0)","(23,0)",
                              "(123,0)","(0,1)","(1,1)","(2,1)","(3,1)","(12,1)","(13,1)","(23,1)","(123,1)",

"(0,2)","(1,2)","(2,2)","(3,2)","(12,2)","(13,2)","(23,2)","(123,2)","(x,3)"),c("(0,0)","(1,0)","(2,0)","(3,0)","(12,0)","(13,0)","(23,0)",
                              "(123,0)","(0,1)","(1,1)","(2,1)","(3,1)","(12,1)","(13,1)","(23,1)","(123,1)",
                              "(0,2)","(1,2)","(2,2)","(3,2)","(12,2)","(13,2)","(23,2)","(123,2)","(x,3)")))
R.store.pitching=array(NaN,c(24,5,nplayers.pitching))
Exp.runs.store.pitching=matrix(NaN,nrow=24,ncol=nplayers.pitching)

for (i in 1: nplayers.pitching){
temp.pitching=trans.pitching(prob.pitching)
temp.trans.pitching=temp.pitching[[1]]
temp.eruns.pitching=temp.pitching[[2]]

# creating E: rowsums(E)= expected number of batters at each starting state for the inning
I=diag(24)
Q=temp.trans.pitching[-25,-25]
E=solve(I-Q)

# Expected Runs for rest of inning at starting state: mult by 9 for full game
Exp.Runs= E%*%temp.eruns.pitching
Nine.inn=Exp.Runs*9
Nine.inn

trans.store.pitching[,,i]=temp.trans.pitching
R.store.pitching[,,i]=temp.eruns.pitching
Exp.runs.store.pitching[,i]=Nine.inn
}


compare.pitching=cbind(teams.pitching[,c(1,4)],Exp.runs.store.pitching[1,], teams.pitching$R/162, teams.pitching$R)
compare.pitching$Exp.runs.162.pitching= Exp.runs.store.pitching[1,]*162
compare.pitching$Percent.change=
round(((compare.pitching$Exp.runs.162.pitching-teams.pitching$R)/compare.pitching$Exp.runs.162.pitching)*100, 3)
mean.pitching=mean(abs(compare.pitching$Percent.change))
```

Winning Pct Code

```
teams.standings= read.csv("MLBStandings2015.csv", header=TRUE, nrow=31)

N= 100000
nteams=nrow(teams.standings)
pred.Win=NA
pred.Loss=NA
pred.win.pct=NA
pred.win.pct2=NA
Diff.wins= NA

for (i in 1:nteams){
	pred.win.pct[i]= round((Exp.runs.store.batting[1,i])^1.81/((Exp.runs.store.batting[1,i])^1.81+(Exp.runs.store.pitching[1,i])^1.81),3)
	pred.win.pct2[i]= round((teams.batting$R[i])^1.81/((teams.batting$R[i])^1.81+(teams.pitching$R[i])^1.81),3)
	pred.Win[i]= round(pred.win.pct[i]*162,0)
	pred.Loss[i]= round(162-pred.Win[i],0)
	Diff.wins[i]= teams.standings$W[i]-pred.Win[i]
}


compare.win.prob=cbind(teams.standings[,c(2,5:7)],pred.Win,pred.Loss,pred.win.pct,Diff.wins)
compare.win.prob$PT= round(162*teams.batting$R^2/(teams.batting$R^2+teams.pitching$R^2),0)
compare.win.prob$PTDiff.wins= teams.standings$W-compare.win.prob$PT

hist(compare.win.prob$Diff.wins,breaks=10, main="Actual-Predicted Wins", xlab="Actual-Predicted",
	col="dark green", border="white")

lm= lm(teams.standings$W.L.~pred.win.pct)
lm2= lm(teams.standings$W.L.~pred.win.pct2)
#summary(lm)
plot=plot(pred.win.pct, teams.standings$W.L., type="p",xlab="Predicted Win Percentage", ylab="Actual Win Percentage",
	col="dark green", pch=16, main="Actual vs. Predicted with Markov Runs")
abline(lm)

plot2=plot(pred.win.pct2, teams.standings$W.L., type="p",xlab="Predicted Win Percentage", ylab="Actual Win Percentage",
	col="dark green", pch=16, main="Actual vs. Predicted with Observed Runs")
abline(lm2)

#lm.win.pct=predict(lm)

mean.win.diff= mean(abs(Diff.wins))
mean.win.diff.PT= mean(abs(compare.win.prob$PTDiff.wins))
```