Major League Baseball Simulation Analysis

The goal is to simulate the Major League Baseball season in near real time hundreds of thousands of times to get odds for various events occurring. What are the odds that the San Francisco Giants will repeat as World Series Champions? Which team is most likely to be the worst team in the AL West? Simulations let us make educated guesses to these questions. How the odds change as the season progresses also can be instructive.

So, how is this done? There are many ways. Full baseball simulators, like Diamond Mind Baseball as an example, are most accurate. The fully simulate the actual mechanics of baseball using individual players competing virtually. The accuracy comes at the expense of complexity and speed. Simulating a season 10000 times in Diamond Mind Baseball takes a couple of weeks!

Alternatively, a rating system for each player could be introduced and those player's abilities could be used to drive a lightweight simulation for each game. This has the advantage of using individual player abilities while avoiding the complexity of simulating each baseball pitch or plate appearance.

The approach simodds.com uses is a greater level of abstraction. Since the outcome of the season is based on wins, and since wins occurs at the level of the individual team within each game, all individual information is discarded and teams are compared directly against each other. For a specific game this makes the predictions less accurate. Pitching matchups, for example, certainly have a great influence on any individual game and will weaken the predictive power. Howeber, over many games these fluctuations even out and the team's relative strengths can be compared with good accuracy. This has many advantages in speed and simplicity. All that is needed to predict a team's future is the team's current rating and the outcomes of each game as they occur. Multiple hundreds of thousands of simulations can be run in less than an hour with this approach.

Research

To simulate the games the schedule for the season is required. Also, some algorithm must be determined for generating runs for each team according a weighted random number generator. For the soccer simulations simodds.com uses a Poisson generator. For baseball we need to looks at the data and see what is the best fit. Also, ELO ratings are used to compare team's relative strengths. Each sport has different parameters for updating ELO ratings. We need to determine the most accurate combination of these parameters and apply them to historical baseball results. This research is using all baseball games since 1990, loaded from Retrosheet.

<u>Run distribution</u>

To start, lets run some quick metrics to make sure the data looks reasonable. We would expect the home team to score more runs per game:

	Average Runs
Home Team	4.692845
Away Team	4.561664

We would expect the home team to win more games:

	Winning Percentage
Home Team	0.539
Away Team	0.461

So the quick metrics look fine. This is no guarantee that the data is accurate but I'm not going to review 50,000 plus games by hand!

Before loading the run information into R for analysis of the distribution, first I looked at the average runs per out for extra inning versus standard games for the away teams. I had a suspicion that extra inning games would have a lower runs per out ratio. Sure enough:

	Average Runs Per Out
Nine Innings or Less	0.08574
Extra Inning Game0.073023689274993232367	0.07302

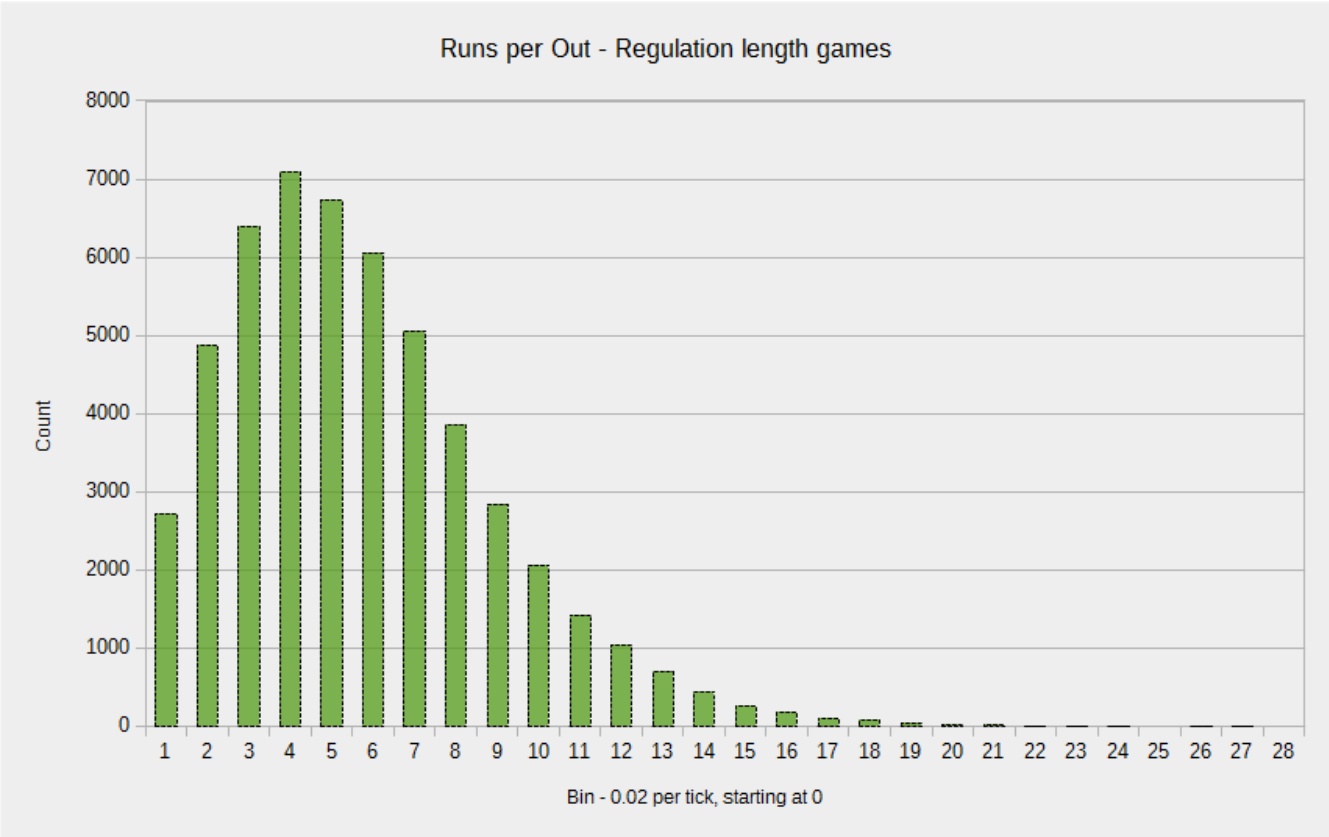
Extra inning games have less runs per out than a regular nine inning game by a significant margin.

If I include extra inning games, the expected outcome of the runs per game will be slightly skewed. Instead I will use only regulation length games for the runs analysis and then apply the different run rate in the simulation algorithm only if a tie needs to be broken.

What about leagues? Should I care about whether or not the games are in the AL versus the NL. We know the AL has more offense than the NL already. If there was no inter-league play then I would not worry about it. But there is... so I better at least take a look. Here are the average runs per out for the home teams.

	Average Runs Per Out
NL Only	0.0883
AL Only	0.0938
NL Home, AL Away	0.0879
AL Home, NL Away	0.0925

This makes it clear that games played in the AL parks have a large increase in runs scored, as would be expected with the DH rule. However, the difference when an NL teams plays in an AL park versus AL only games is noticeable but is the equivalent of about 0.07 runs per game. I suspect this would be significant statistically but to account for that in the sim I would need to make the assumption that the NL is inferior to the AL in quality. Although that has been true the last several years, I would prefer for the ELO ratings to reveal this advantage rather than hard code it into the simulation. So, only the home and away distributions will be used in the sims.

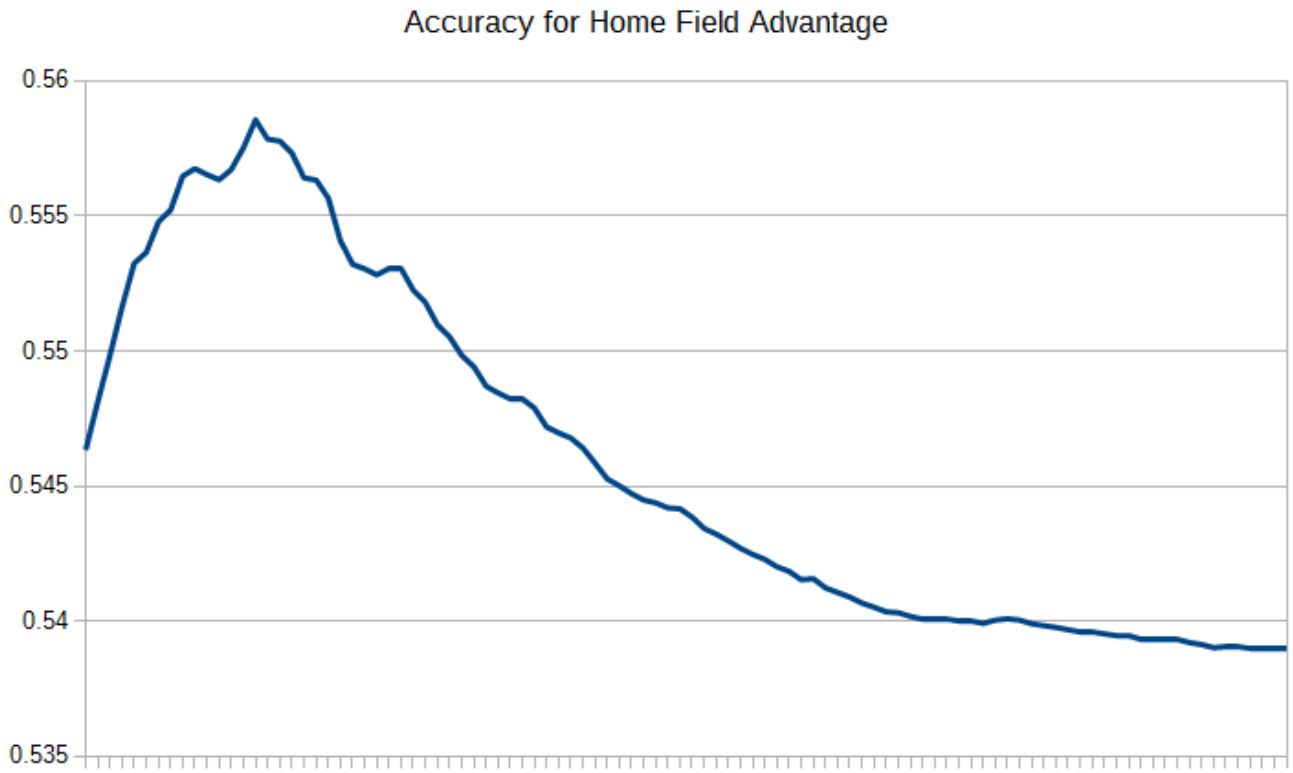


ELO Analysis

The next step is evaluating the best matching ELO calculation parameters for baseball. Each sport has different best values for variables like home field advantage or the regression to the mean between season for each team (to account for things like trades or free agency). To tease out the influence of these values we first need to write a bare bones ELO calculation tool. All teams will start in early 1990 with a baseline ELO rating of 1700. Real life games are then fed into the calculator in chronological order. The ELOs are adjusted based upon the result of each game. The predictions from the ELO ratings for both teams are compared to each other and the real life result. A success or failure is tallied. At the end, the overall success rate is shown as an error term. The goal is to find the ELO parameters that minimize this error term.

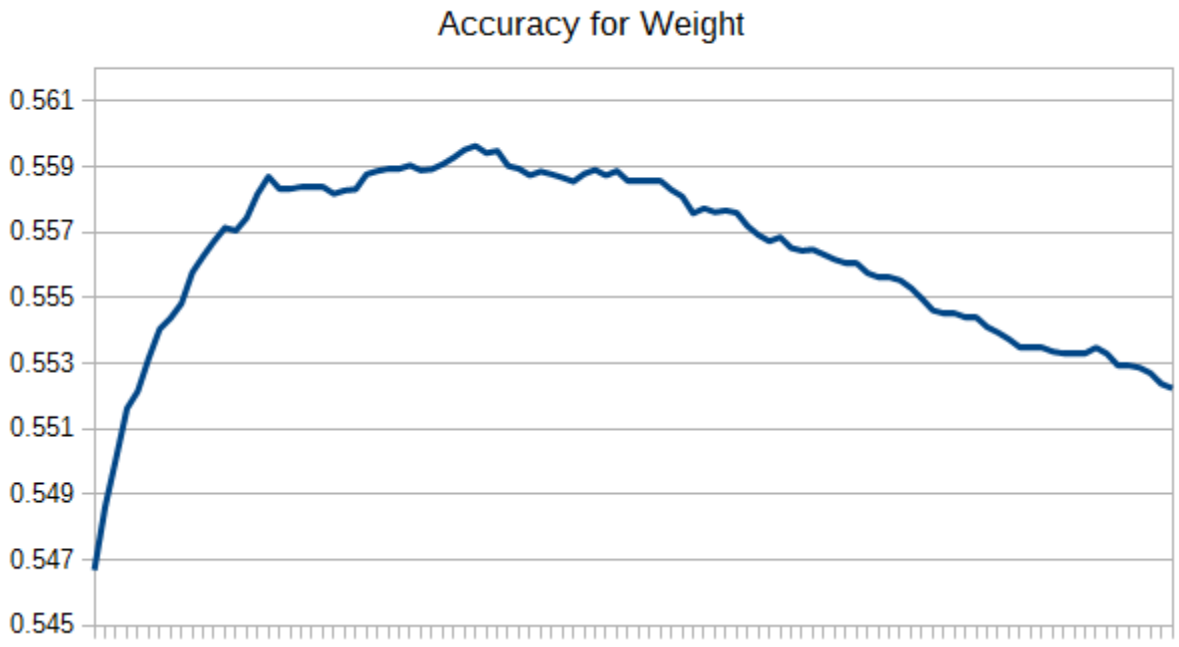
Home Field Advantage

Home Field Advantage (HFA) is a significant factor for baseball simulations. Below is a graph showing the accuracy of prediction for a wide range of HFA values.



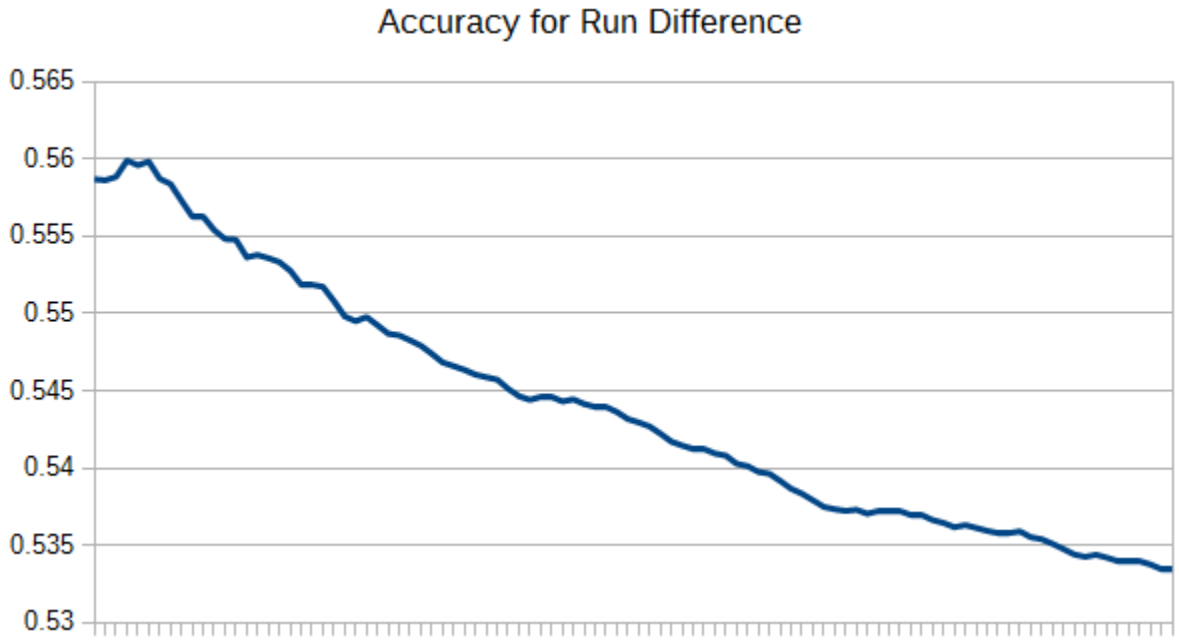
Weight

The weight is defined as the importance assigned to each individual game. The higher the importance, the more the ELO rating will change for any particular result. Too little importance and real trends in the team's performances will lag and be useless. Too much importance and changes look like trends but they aren't real. A relative light weight is optimal for baseball, which isn't a big surprise considering the large number of games per season.



Run Differential

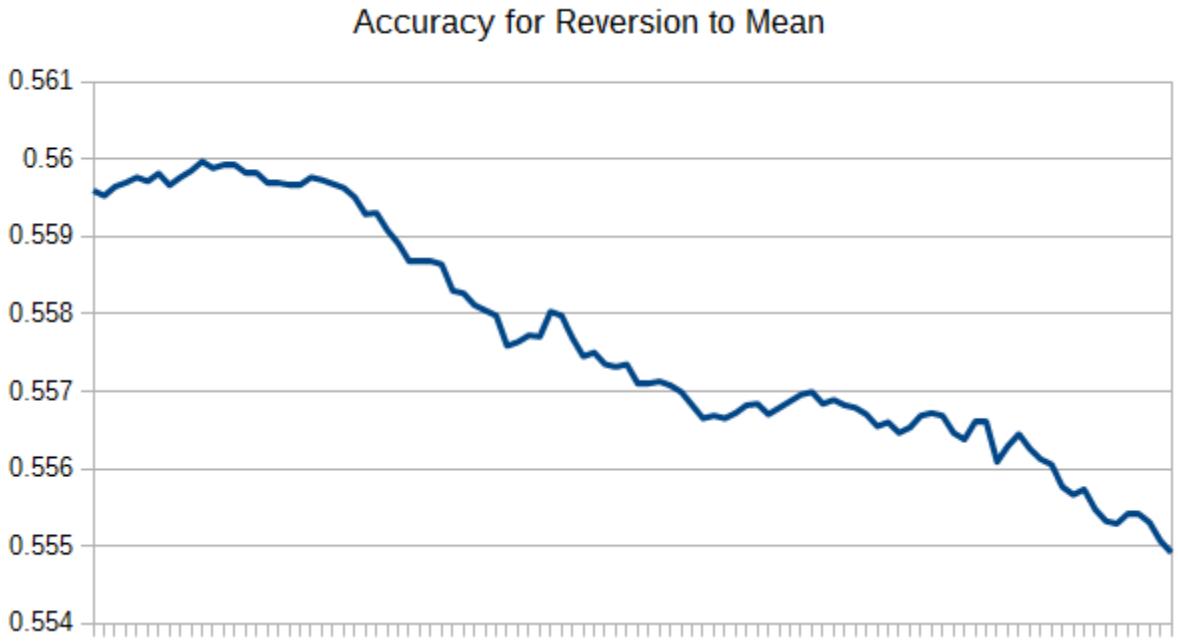
Run differential measures how much the weight for a particular game should change based upon the margin of victory. Usually teams that average a larger margin of victory is considered a stronger team. This may be the most obvious statement of the year. Since HFA and weight have already been optimized, any remaining variation should be in differential and between season reversion to the mean.



The benefit from run differential is minimal.

Reversion to mean between seasons

The biggest weakness of the ELO approach is that it is a lagging indicator. The team's results determine the ELO rating, but if major changes take place within the team it will be awhile before the effect is known. In a baseball simulation, if a player gets injured for the season remainder they can be taken out of the lineup. In ELO simulations, the difference are only revealed over time through the performances. Although trades do take place within the season, between season changes can be significant. This run will test how much of a reversion to the mean ELO (1700) should take place each off season. 0.0 means that the end of season ELO is retained. 1.0 means that each team starts with a fresh 1700 rating.



There is a slight improvement to make a minor reversion to the mean between season. However, the change is really small, which means teams are more robust between season than we might expect intuitively.

Genetic Algorithm

At this point the tweaking is almost done. The prediction success rate is now a 0.560, which is quite a bit better than always picking the home team, for example. There is likely a finer degree of tuning that could be done. One approach, which I used for the NFL simulations, is to use a genetic algorithm to randomly evolve the best local solution that can be found. Doing this with the NFL gained another 10% in accuracy. I'm going to wait to do this for baseball, however. It is a time consuming process and with so many baseball games in a season I don't expect it to be as productive.