# MIT 2.086: Simulation of a Baseball Game

*Note: This document assumes you are familiar with the game of baseball. If you are unfamiliar with the rules of the game, please first read a basic introduction to how baseball works (for example, [http://entertainment.howstuffworks.com/baseball.htm](http://entertainment.howstuffworks.com/baseball.htm))*

In your second assignment, you built a code to simulate a single at-bat given a player's probability of hitting a single, double, triple, or home run. It is possible to use this code to simulate an entire game of baseball. To do so, we just need to write some additional code to keep track of things like baserunners, outs, and runs scored. Here's the basic outline of how such a simulator might work

```
runs = 0
for i = 1:9
    outs = 0
    while outs < 3
        Simulate an at bat
        Was an out made? If so, increment number of outs
        Advance baserunners
        Did any runners score? If so, add number of runs scored to variable "runs"
    end
end
```

To run this code, we need to figure out what should go in the while loop? The MATLAB file "baseball_game.m" contains some intuitive rules for advancing runners. When a single is hit, all runners who are on base advance one base. When a double is hit, all runners who are on base advance two bases. Give "baseball_game.m" a look—do you understand the logic?

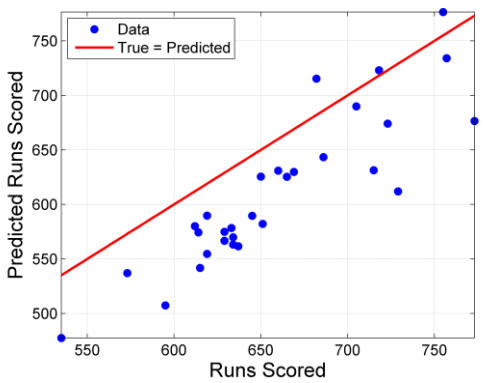
We tested this code using data from each of the 30 Major League Baseball teams during the 2014 season. For each team, we counted the average probability of a single, double, triple, and home run. We then simulated a bunch of games for each team, allowing us to estimate the number of runs the team would score over the course of a single season. This is done in "simulateAllTeams.m". Since there are 162 games in a season, all we do to estimate the total runs scored in a season is simulate a bunch of games, find the average number of runs scored, and multiply this number by 162.

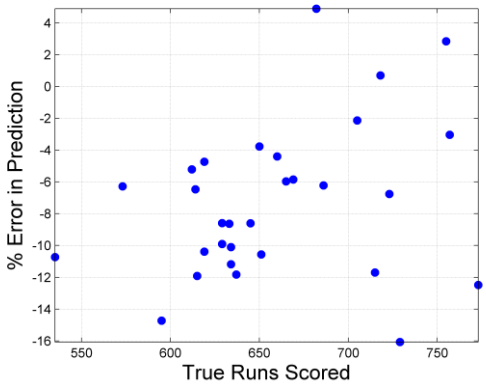


We can then compare these *predicted* runs scored with the actual number of runs the team scored in 2014. We see that our simulator isn't bad, but teams score about 15-20% less in our simulator than they do in real life.

One reason for this might be that our rules for advancing runners are too simple. For example, often a runner on second will score when a single is hit (rather than stopping at third). Similarly, a runner who starts at first when a double is hit will often score. Also, we have the issue of walks (when a pitcher throws too many balls)—in this case the batter will advance to first, but baserunners only advance on walks if they are forced to (for instance, if there is a runner on second, no runner on first, and the batter walks, the runner on second does not get to go to third).

The code "baseball_game_withWalks.m" simulates a baseball game with some of these more subtle baserunning rules. Now, in addition to the probability of an out, single, double, triple, or home run, we must also provide the probability that a batter walks. You'll see that "baseball_game_withWalks.m" is a little more complicated, but look at the code and see if you can follow what is happening. The script "simulateAllTeams_walks.m" estimates the average runs scored for all 30 Major League Baseball teams from their probabilities of getting a single, double, etc.



Using the new advancement rules, the simulator is much more accurate in predicting the number of runs a team would score over the course of the season. It still underestimates, but now we underestimate by 6-7% versus the 15-20% underestimation we saw using the rudimentary rules for advancing baserunners.

***Why does any of this matter?*** Computer experimentation is fast, cheap, and void of real world consequences. These computer simulations can inform important, *real-world* decisions. Suppose you are the owner of the Red Sox. With the astronomical rise in player salaries, if you'd like to hire a star player, this could easily cost you over *100 million dollars* over the life of the contract you offer this player. Would you be better off buying a single superstar player, or 2 above average players for 50 million dollars each? One way you might answer this question is to use a simulation procedure like we've described, and look at the number of runs you would score with each option (option 1: single superstar, option 2: two above average players).

There are a number of other questions such simulation might be able to answer, such as:

1. How does *bunting* affect the average number of runs my team scores in an inning
2. What percent success rate is required to make attempting to *steal a base* a beneficial strategy?

There are a variety of other tasks/questions you could look into, such as making the simulator more realistic.