Stolen Base Predictor

Ronan Hwang

In this project, I created a program that predicts whether or not a base runner steals second base successfully based on the runner, the runner's jump, the pitch velocity, and the catcher. To predict the stolen base outcome, I compared the time it takes for the runner to get to second base with the time it takes for the ball to get from the pitcher's hand to second base. The program will take in the input of the pitch velocity, the baserunner, the baserunner's jump, and the catcher to generate a prediction.

The first step in this process was to calculate the time it takes for the ball to go from the pitcher's hand to second base. To do so, I will take the sum of the catcher's pop time and the time it takes for the pitch to reach home plate. Specifically, the user-inputted pitch velocity will be converted so that a time interval is given for the pitch(mph) to go from the pitcher's hand to home plate (60.6 feet). With that time, I will add that with the catcher's pop time. To determine the pop time of the user-inputted catcher, I decided to use the competitive pop time average of the catcher on stolen base attempts to second base. One summing the two values together, the next step is to calculate the time it takes for the baserunner to reach second base given the jump, which I will call the "Runner time".

For this program, I will determine the jump to be the distance that the base runner gets off first base when the pitch is thrown. To calculate the Runner's time, I subtracted 60 feet(the distance from 1B to 2B) from the final jump distance, and calculated the time it takes for the runner to get to second base in that remaining distance. This time will be calculated with the runner's sprint speed, where the sprint speed along with the remaining distance will give us the

total time. To calculate the jump, I will grab all of the runner's stolen base data, and get the minimum and maximum distances that the baserunner got prior to pitch release. With that, the new jump will be the lowest jump that the base runner had added with the jump range multiplied by the user-inputted jump divided by 10. It will look like this:

```
new_jump = (jump_range * (jump / 10)) + min_jump
```

Then, I will subtract 60 from the new_jump to calculate the remaining distance that the base runner has to cover. To calculate the time it would take for the baserunner to cover the remaining distance. I graphed the running distance splits of the baserunner from 0 to 90 feet and proceeded to use that trendline as an estimate as to how long it would take for the runner to cover the remaining distance. Another approach that I could have done is to get the sprint speed of the baserunner and use the speed, distance formula to calculate the time.

In the future, something I will need to take into account is the location of the throw to second base. A ball that may get to second base a little high will add crucial milliseconds to the total fielding time, which may affect the stolen base outcome. Another thing that will not be taken into account is the release point for the pitcher. Currently, the pitch velocity time to the plate is calculated by getting the time for the pitch over 60.6 feet. However, I did not take into account how far in front of the rubber a pitcher releases a ball. For example, A 95 mph fastball from Sonny Gray will take longer to reach the plate than a 95 mph fastball from Chris Sale because of the difference in release point and release distance from the rubber.

The goal for this project is to continue to find ways to make this stolen base program more accurate and find ways to take into account some of the external factors that come into play when stealing a base.