Digital Twins: An Application to Baseball Decision-Making

Matt Eisenhauer

1. Introduction

Professional baseball has undergone profound transformations in recent years, not only in how the game is played, but how it is strategized. The integration of new technologies such as high-speed cameras and radar systems has allowed for organizations to leverage data at a level of granularity never seen before. The introduction of MLB's Statcast system in 2015 [1], deployed in all 30 Major League stadiums, has enabled baseball to go far beyond the common statistics that have long dominated strategy and in-game decision making.

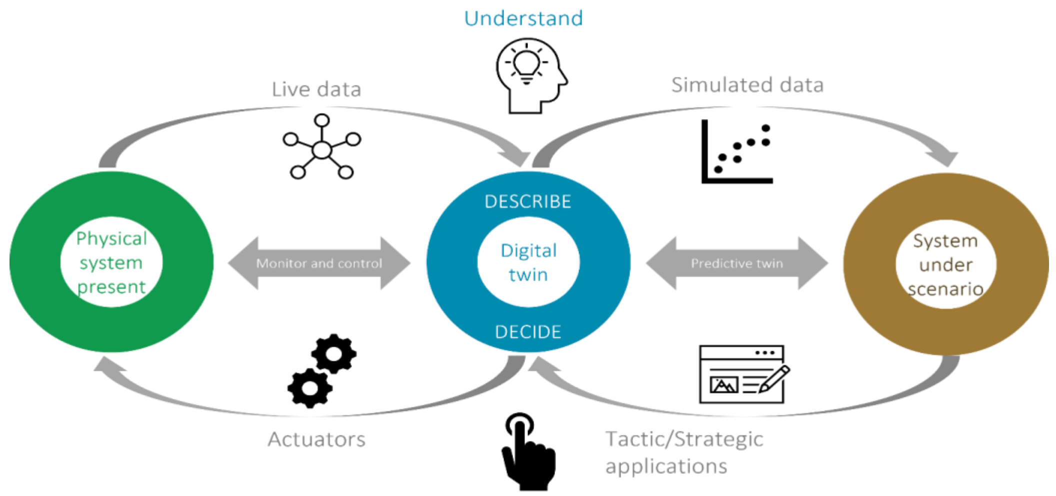
The Statcast system is comprised of a combination of Doppler radar technology and twelve high-frame-rate cameras, strategically positioned across MLB stadiums to enable pitch, hit, player, and bat tracking [1]. The system now captures metrics such as spin rate for every pitch, exit velocity for every batted ball, sprint speed for base runners, and exact player locations throughout the course of a game. In the 2023 season alone, Statcast captured more than 725,000 pitches and 125,000 batted balls, quantifying all aspects of professional baseball in a way never possible before [1]. The focus of this paper will rely on Statcast's live player tracking capabilities and their application as a Digital Twin for organizations to enhance in-game decision making.

2. Background

Digital Twins, first proposed by Dr. Michael Grieves in 2003, was originally defined as a virtual model of a physical entity, intended to optimize life-cycle management of a given product [2]. At its core, Grieves outlined the Digital Twin made up of three core components, working together to create a powerful feedback loop for analysis and decision making:

1. **The Physical System:** This is the real-world entity being modeled. Early applications of the Digital Twin focused on manufacturing, supply chain, or even structural health of a spacecraft used by Grieves and his counterparts at NASA [2].
2. **The Virtual Model:** This virtual environment is not just a static 3D model of the physical entity, but a dynamic representation that accounts for the physical rules, constraints, and probabilistic outcomes of variable events. It aims to remain an exact replica of reality, mirroring the physical system as events unfold.
3. **The Data Link:** This is the continuous, two-way data flow between physical and virtual environments. Modern tracking technology, whether it be physical sensors (tagging), or high-frame-rate cameras, like Statcast's, provide the real-

time data from the physical environment to power the virtual model. The most powerful data links reflect this data simultaneously within the virtual to ensure a high-speed replica of currently reality is maintained. In turn, this provides decision makers the ability to carry out 'what-if' scenarios, offering decision making insights to influence actions in the real-world.



[5] Figure 1 - Digital twin diagram showing operational applications on the left and tactical and strategic applications on the right. From "Digital Twins for the Water Sector," KWR Water Research Institute, www.kwrwater.nl/en/projecten/digital-twins-for-the

The use-cases for Digital Twins have proved endless, with industries such as construction, manufacturing, healthcare, and aerospace leveraging the advancement of Internet of Things (IoT), cloud computing, and artificial intelligence to replicate physical environments in a virtual world [3]. The wide array of applications have in turn resulted in varying definitions of the Digital Twin. While the core components remain the same, different goals of a twin model are likely to result in different functionalities.

Kritzinger et al. [6] aimed to classify the Digital Twin based on data integration levels: the Digital Model, Digital Shadow, and Digital Twin. In this hierarchy, the Digital Model relies solely on manual data exchange, while a Digital Shadow relies on a one-way automated data link from the physical environment to reflect changes within the virtual. Conversely, the Digital Twin is defined by a fully automated bi-directional dataflow, where the virtual environment not only mirrors in the physical, but can control or influence it directly.

For the purpose of this analysis, the proposed Digital Twin will likely fall somewhere between Kritzinger's definition of a Digital Model and Digital Shadow. The intent here is to display the potential value a Digital Twin can bring to Major League Baseball organization's while operating within technological constraints.

3. Problem Overview

While the Statcast system has evolved to become a robust repository of baseball data points, this paper aims to extract value specifically from its player tracking capabilities.

Because Statcast delivers its data in real-time, one can begin to define exactly how the system can be leveraged as a Digital Twin. Here, the Twin's physical system is a live baseball game – accounting for players, ballpark dimensions, and in-game events.

The data link and virtual environment are both enabled by Statcast itself. Not only do Statcast's high-frame-rate cameras deliver event data in real-time, they also have proven their capability to deliver a true 3D model of a game unfolding as it happens. In 2023, Major League Baseball unveiled its Gameday 3D experience powered by Statcast, allowing fans at home to interact with a 3D replica of a live game from any conceivable angle in the park [4]. While Gameday 3D is purely for fan entertainment, it speaks to the capability of Statcast's ability to capture all physical aspects of a game in real-time.

Additionally, Statcast data made public via MLB's Baseball Savant website furthers the capability of such a model to allow for simulations of in-game scenarios. The platform captures metrics such as ball flight, exact player location and the kinematics behind their movements, and variable outcomes through its historical data. This analysis will focus on the combination of the two – leveraging Statcast's historical data to understand player tendencies with the intent to marry it with the Digital Twin, resulting in a robust model specifically built for in-game decision making.

3.1 Defining the Scenario: The Stolen Base Attempt

To construct a functional digital twin, a well-defined, data-rich scenario is required. This project will focus on baserunning scenarios, allowing managers to instantaneously simulate potential outcomes to assist with in-game decision making. Specifically, this tutorial will limit analysis to a singular event – the stolen base attempt.

For simplicity, we will further constrain to only focusing on a baserunner attempting to steal second base. In this context, a stolen base occurs when a runner on first base attempts to advance to second as the pitcher begins his motion. As the pitch is delivered, the catcher may throw the ball to second base, where it is received by an infielder. If the infielder catches the ball and tags the runner before the runner touches second base, the runner is called out. If the runner reaches second base safely before being tagged, the stolen base is successful, and the runner assumes his new position at second. Using Statcast data, we can develop a simulation of this event and augment the model given specific tendencies of a given runner, pitcher, and catcher.

3.2 Defining Value with Run Expectancy

Before modeling the stolen base event, it's useful to understand where such a model can deliver value. One way to quantify the value of a stolen base is through the Run Expectancy Matrix (RE24). Pioneered by operations researcher George Lindsey in 1963 and refined by modern statisticians [8], RE24 measures the average number of runs a team is expected to score during the remainder of an inning for each of the 24 base-out states [7]. There are three possible out counts (0, 1, or 2) and eight possible baserunner scenarios, yielding 24 base-out combinations in total. An example RE24 matrix is shown below. Note that the below matrix uses a run environment of 4.15 runs per game, as iterations of RE24 will change based on annual league scoring tendencies and ballpark-specific factors.

Runners	0 Outs	1 Out	2 Outs
Empty	0.461	0.243	0.095
1 _ _	0.831	0.489	0.214
_ 2 _	1.068	0.644	0.305
1 2 _	1.373	0.908	0.343
_ _ 3	1.426	0.865	0.413
1 _ 3	1.798	1.14	0.471
_ 2 3	1.92	1.352	0.57
1 2 3	2.282	1.52	0.736

[7] Figure 2. RE24 Matrix (Run Environment = 4.15 Runs Per Game). Source: FanGraphs Library.

To see how the matrix works, consider the “0 Outs” column. Historically, teams with the bases empty and no outs can expect to score, on average, 0.461 runs that inning. Putting a runner on first with no outs increases the expectancy to 0.831 and advancing that runner to second raises expectancy further to 1.068 runs.

This progression allows the calculation of exact risk-reward outcome of a stolen base attempt. In the 0-out scenario, a successful steal transitions the state from *Runner on 1st* to *Runner on 2nd*, yielding a net positive run expectancy ($\Delta RE_{Success}$):

$$\Delta RE_{Success} = 1.068 - 0.831 = +\mathbf{0.237}\ runs$$

However, the penalty for failure is more severe. Being thrown out transitions the state from *Runner on 1st, 0 Outs* (0.831) to *Bases Empty, 1 Out* (0.243), resulting in a loss ($\Delta RE_{Failure}$):

$$\Delta RE_{Failure} = 0.243 - 0.831 = -\mathbf{0.588}\ runs$$

Because the penalty for failure (-0.588) outweighs the reward for success (+0.237), a runner requires a high probability of success to justify the attempt. We can calculate the break-even success rate (P_{BE}) where the expected value of the move is neutral:

$$P_{BE} = \frac{|\Delta RE_{Failure}|}{|\Delta RE_{Failure}| + \Delta RE_{Success}} = \frac{0.588}{0.588 + 0.237} \approx \mathbf{71.3\%}$$

This calculation demonstrates that a runner must be safe more than 71.3% of the time to add value to the team. The high threshold underscores the necessity of the proposed Digital Twin model: rather than guessing, a manager needs precise, real-time probability estimate to determine if the specific conditions (pitcher speed, lead distance, catcher arm) exceed this critical 71.3% break-even point.

4. Model Definition

The below section outlines the simulation model, defining entities, attributes, data sources, and the simulation process.

4.1 Entities and Attributes

The model consists of four primary entities. The attributes for these entities are initialized using historical data from the “Physical System” (MLB Statcast), serving as the data link in the digital twin architecture. Table 1 summarizes the mapping between input attributes and simulation variables.

Table 1. *Model Entities and Simulation Variables*

Entity	Input Attribute (Data Source)	Variable	Description
Baserunner	Lead Distance (First Move)	Lead Distance	d_{lead}
	5ft Split Time	Reaction Time	t_{react}
	Full Split Array (5-90ft)	Run Time	t_{run}
Pitcher	Pitcher Stealing Runs	Delivery Time	t_{pitch}
Catcher	Pop Time to 2B	Pop Time	t_{pop}
System	Global Constant	Tag Time	t_{tag}

4.2 Attribute Derivation and Model Assumptions

Simulation input attributes were derived from publicly available Statcast data via MLB’s Baseball Savant [10]. Due to limitations in public data granularity, this simulation operates on specific model assumptions to illustrate what is possible for professional organizations, given their access to raw Statcast data. Explanations for each entity and attribute as well as the assumptions / limitations that come with them are explained below.

A. Baserunner:

- **Lead Distance (d_{lead}):** The initial distance from first base at the start of the pitcher’s delivery.
    - **Data Source:** Average Distance from First Base at Pitcher’s First Move (Statcast).

    - **Assumption:** While Statcast provides the mean (μ_{lead}), the standard deviation (σ_{lead}) is derived based on a "Risk/Reward" heuristic. We assume runners with longer leads accept higher variance (greater threats to steal):
        - Aggressive $(\mu_{lead} > 12.0ft)$: $\sigma_{lead} = 1.0ft$
        - Standard $(11.0 \leq \mu_{lead} \leq 12.0ft)$: $\sigma_{lead} = 0.5ft$
        - Conservative $(\mu_{lead} < 11.0ft)$: $\sigma_{lead} = 0.3ft$
- **Reaction Time $(\boldsymbol{t_{react}})$:** The delay between the pitcher's first move and the runner's kinematic start.
    - **Data Source:** 5ft Split Time (seconds), Home Plate to First Base (Statcast).
    - **Assumption:** 5ft Split acts as a proxy for a baserunner's reaction time, assuming runner's ability to get out of the batter's box upon contact reflects their initial acceleration ('burst') on a stolen base attempt.
        - Elite Burst (Top 25%): $N(0.18, 0.02)$
        - Average Burst: $N(0.20, 0.05)$
        - Poor Burst (Bottom 25%): $N(0.24, 0.08)$
- **Run Time $(\boldsymbol{t_{run}})$:** The duration required to traverse the effective distance to second base $\left(d_{eff} = 90ft - d_{lead}\right)$.
    - **Data Source:** 90ft Splits, captured in 5ft increments (Statcast).
    - **Assumption**: Split times are used as a lookup table to calculate runner's total service time, based on d_{eff} 5 to 90ft Split times.

B. Pitcher:

- **Delivery Time $(\boldsymbol{t_{pitch}})$:** The time from the pitcher's first move to the ball reaching the catcher.
    - **Data Source:** Pitcher Stealing Runs (Statcast).
    - **Assumption:** Time to Plate data is not publicly available. Instead, we use Pitcher Stealing Runs, a Statcast-developed metric that assigns value to a pitcher's ability to limit a runner advancing on the basepaths. Pitchers are assigned a tier based on this value (Elite / Average / Slow), developed from public scouting evaluations [9].
        - Elite (75th Percentile): $N(1.25,\ 0.05)$
        - Average: $N(1.35,\ 0.08)$
        - Slow (25th Percentile): $N(1.55, 0.10)$
    - **Limitation:** This model utilizes effective delivery time and does not distinguish between LHP/RHP pickoff moves.

C. Catcher:

- **Pop Time $(\boldsymbol{t_{pop}})$:** The time from pitch reception to the catcher's throw reaching the glove at second base.
    - **Data Source**: Average Pop Time to 2B (Statcast).

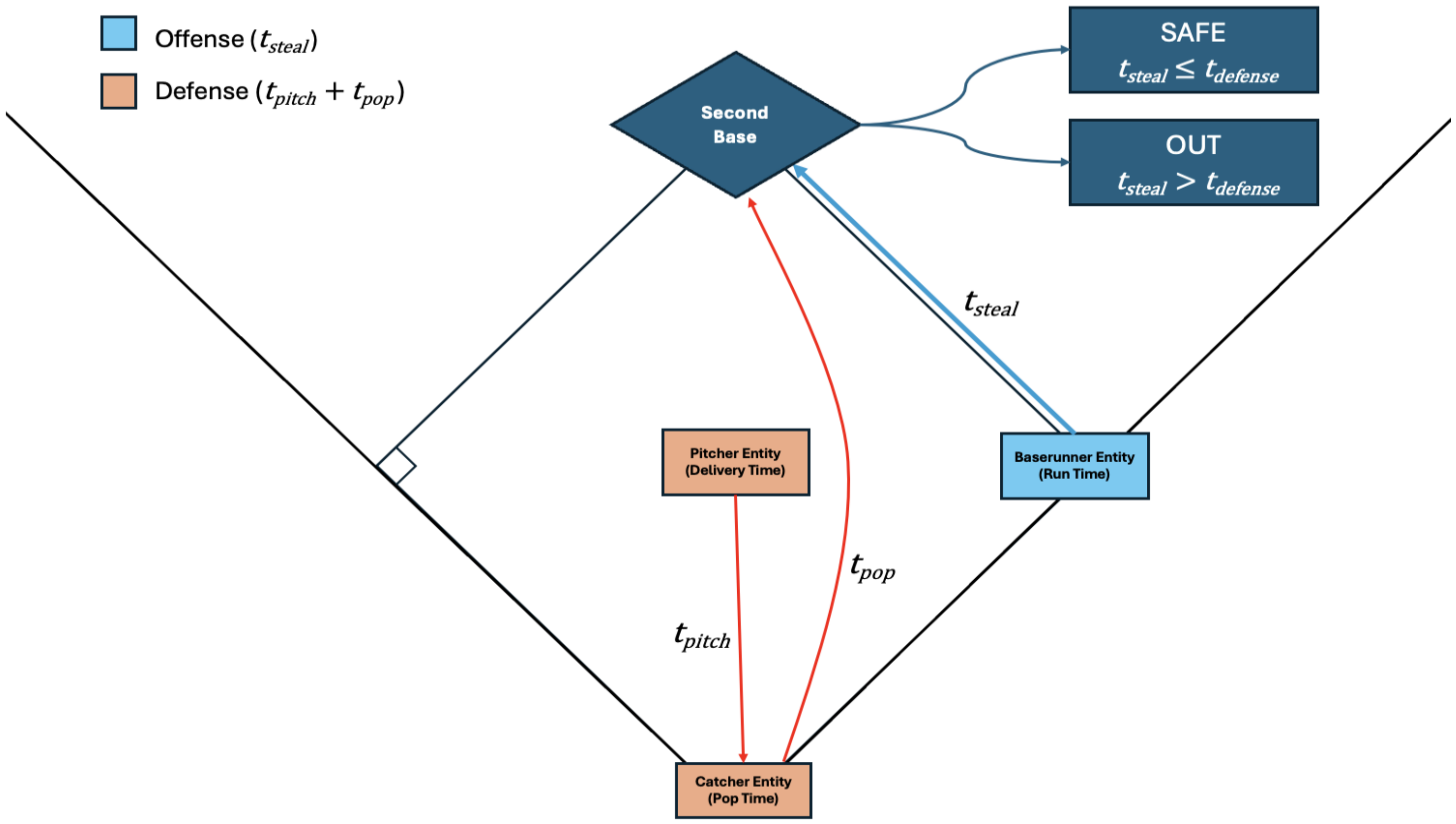
- **Assumption:** Statcast provides the mean (μ_{pop}), but not variance. We assume faster catchers possess more consistent exchange / throw metrics.
  - Elite (Top 25%): $\sigma_{pop} = 0.03s$
  - Average: $\sigma_{pop} = 0.08s$
  - Poor (Bottom 25%): $\sigma_{pop} = 0.10s$

D. System (Middle Infielder):

- **Tag Time (t_{tag}):** A stochastic delay representing the fielder's application of the tag.
  - **Assumption:** Sampled from $N(0.15, 0.05)$ and added to the total defensive service time to account for the physical action required to record an out after the ball arrives.

4.3 The Simulation Process

The simulation executes as a discrete-event system where the stolen base event is modeled as two asynchronous, parallel processes – Offense and Defense - competing for a single resource (Second Base), as shown in Figure 3.


Offense (t_{steal})
Defense ($t_{pitch} + t_{pop}$)
Second Base
SAFE $t_{steal} \leq t_{defense}$
OUT $t_{steal} > t_{defense}$
t_{steal}
Pitcher Entity (Delivery Time)
Baserunner Entity (Run Time)
t_{pop}
t_{pitch}
Catcher Entity (Pop Time)

1. **Initialization (t=0) -** At t = 0, the system state is initialized by loading the specific attributes for the Baserunner (R), Pitcher (P), and Catcher (C).

    - The Pitcher Entity enters the *Delivery* state
    - The Runner Entity enters the *Reaction state*
    - The System Clock begins

2. **The Defensive Process -** The defensive timeline $(t_{defense})$ is modeled as a linear sequence of three stochastic service times. These events must occur in strict sequential order – the catcher cannot receive the ball until the pitcher delivers it.

    2.1 Pitcher Delivery $(\boldsymbol{t_{pitch}})$: The system samples a delivery duration from $N(\mu_{ttd}, \sigma_{ttd})$ based on the pitcher's assigned run game tier defined in Section 5.2.
    2.2 Catcher Exchange & Throw $(\boldsymbol{t_{pop}})$: Upon completion of the delivery, the Catcher entity processes the ball. A service time is sampled from $N\left(\mu_{pop}, \sigma_{pop}\right)$.
    2.3 Tag Application $(\boldsymbol{t_{tag}})$: A final stochastic delay is added, representing the fielder's catch-and-tag, sampled from $N(0.15, 0.05)$.

The total defensive service time is calculated as:

$$t_{defense} = t_{pitch} + t_{pop} + t_{tag}$$

Figure 3. Stolen Base Digital Twin - The Simulation Process.

3. **The Offensive Process -** The offensive timeline (t_{steal}) is modeled as a physics-based lookup rather than a sequential server.

    3.1 Reaction Phase: The runner's initial delay (t_{react}) is sampled based on their "Burst Tier".
    3.2 Kinematic Phase: The simulation calculates the effective distance to run (d_{eff}) by subtracting the runner's specific lead distance from the 90-foot basepath:

$$d_{eff} = 90 - d_{lead}$$

    3.3 Interpolation: The run duration (t_{run}) is derived by linearly interpolating d_{eff} against the runner's empirical 90ft split array.

The total offensive service time is calculated as:

$$t_{steal} = t_{react} + t_{run} - 0.15$$

(Note: A static adjustment of -0.15s is applied based on the observed differences of a stolen base start vs. batter's box exit.)

4. **Termination and Outcome -** The simulation terminates when both offensive and defensive processes have resolved. The outcome is determined by comparing the total service times:

    - **SAFE:** If $t_{steal} \leq t_{defense}$, the baserunner occupies the resource before the Tag Event completes. (*Note: If $t_{steal} = t_{defense}$, the runner is called safe, following the common baseball interpretation "the tie goes to the runner".)*
    - **OUT:** If $t_{steal} > t_{defense}$, the Tag Event completes before the Runner arrives.

The Digital Twin operates by running this simulation *N* times (e.g., *N* = 10,000) for a specific set of initial conditions. The probability of success $\left(P_{Safe}\right)$ is defined as:

$$P_{Safe} = \frac{\sum_{i=1}^{N} I(Safe_i)}{N}$$

Where $I(Safe_i)$ is an indicator function equal to 1 if the runner is safe in iteration *i,* and 0 otherwise. This probability is then compared against the break-even threshold (71.3%) to render a 'Go/No-Go' decision.

5. Implementation Strategy and Example Usage

This iteration of the Stolen Base Digital Twin was implemented as a tutorial to review how MLB teams can leverage existing infrastructure to make in-game decisions on stolen base attempts. Recognizing it's limitations, the structure of the twin is defined below:

- **Physical System:** Any stolen base opportunity, with a runner on first base, using data from the 2025 MLB season.
- **Virtual Environment:** The simulation is written in Python, based on user input of pitcher / catcher / baserunner combinations, allowing for execution during in-game situations.
- **Data Link:** The model ingests historical data from 2025 MLB season to determine system variables. Because the simulation is data-driven and not hardcoded, it remains capable of handling real-time data and future Statcast updates.

Figure 4 displays example simulation output, featuring a matchup between known speedster Chandler Simpson of the Tampa Bay Rays and Boston Red Sox battery Greg Weissert and Connor Wong. After 10,000 trials, the simulation predicts Simpson has an

83.6% success probability, exceeding the calculated 71.3% break-even threshold, so the decision is to steal.

```
--- SIMULATION MATCHUP ---
Runner:  Simpson, Chandler (Lead: 11.0ft | 5ft Burst: 0.53s)
Pitcher: Weissert, Greg (Avg TTD: 1.35s) | Class: Average)
Catcher: Wong, Connor (Avg Pop: 1.97s | Class: Average)

PROBABILITY SAFE: 83.6%
DECISION: STEAL IS ON
```

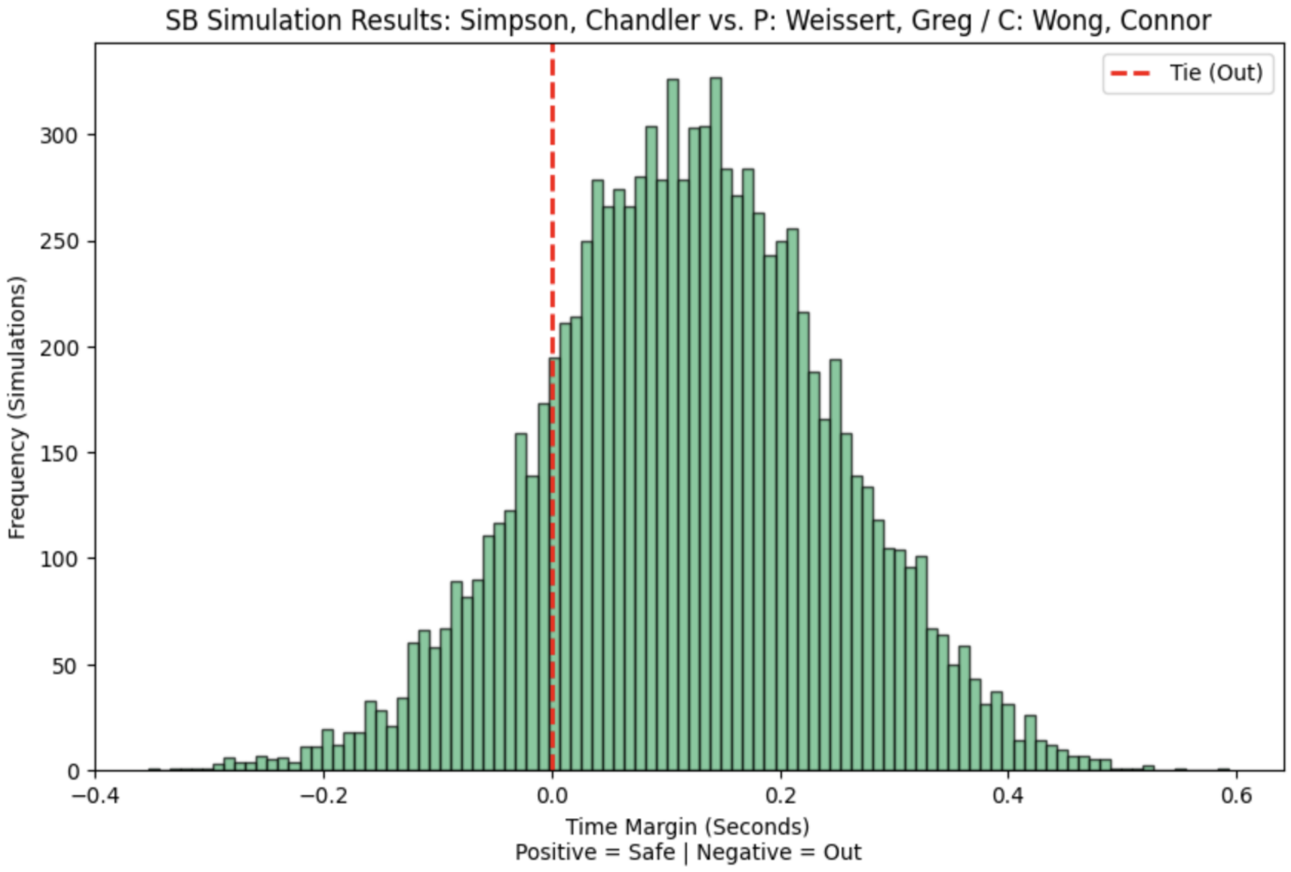


Figure 4. Simulation Output - Stolen Base Success Probability (N=10,000 Simulations)

6. Model Verification and Validation

6.1 Statistical Verification

To further verify numerical assumptions used in the simulation, video analysis via the OpenCV Python library confirmed tiered distributions were within appropriate ranges. As an example, Figure 5 displays frame-by-frame captures of Houston Astros' pitcher Spencer Arrighetti to verify delivery time. With Arrighetti starting this delivery at $t = 1.806s$ and the pitch caught at $t = 3.512s$, we can calculate delivery time as $3.512 - 1.806 = \mathbf{1.706s}$. Arrighetti was classified as 'Slow' ($N(1.55, 0.10)$), so while the classification matches reality, distribution may be conservative in this instance.

[10] *Figure 5 - Spencer Arrighetti vs. New York Mets (03/29/2025) - Time to Plate Analysis. Source: Major League Baseball Advanced Media, Baseball Savant.*

6.2 Validation and Analysis

While true model validation would require excessive compute power and exceeds the scope of this project, we opted to review six stolen base attempts from the 2025 season to assess model accuracy. The results (Table 2) feature a diverse combination of elite, average, and poor tiers to stress-test the simulation.

Table 2. *Simulation Validation – Players are color coordinated based on tier (green = elite, yellow = average, red = poor)*

Date	Pitcher	Catcher	Baserunner	Probability Safe	Recommendation	Actual Outcome
4/25/25	Yoshinobu Yamamoto - LAD	Will Smith - LAD	Oneil Cruz - PIT	8.6%	HOLD	SAFE
5/1/25	Kodai Senga - NYM	Luis Torrens - NYM	Corbin Carroll - AZ	3.3%	HOLD	OUT
7/12/25	Kevin Gausman - TOR	Tyler Heinemen - TOR	Lawrence Butler - ATH	52.6%	HOLD	SAFE
7/12/25	Kevin Gausman - TOR	Tyler Heinemen - TOR	Lawrence Butler - ATH	52.6%	HOLD	OUT
8/19/25	Mitch Keller - PIT	Joey Bart - PIT	Myles Straw - TOR	86.5%	STEAL	SAFE
9/21/25	Greg Weissert - BOS	Connor Wong - BOS	Chandler Simpson - TB	83.6%	STEAL	SAFE

The model correctly aligned with real-world outcome in four out of six cases [10]. The validation set highlighted two distinct insights:

- **Model Calibration (The Butler Case):** Rows 3 and 4 feature Lawrence Butler attempting to steal against Gausman / Heinemen battery twice in the same game. The model predicted a **52.6%** success probability – a statistical 'toss-up'. In reality, Butler was safe once and out once. This split outcome underlines the model's ability in identifying marginal, high-variance matchups where the decision to steal is statistically indifferent.
- **Model Limitation (The Cruz Case):** Row 1 shows the model issuing a strong "HOLD" recommendation (8.6% success probability) for Oneil Cruz, yet in reality he was safe. Video review of the play reveals Cruz stole on a 77-mph off-speed pitch. This exposes a key model limitation: the current Digital Twin does not factor in pitch velocity, with slower pitches allowing the runner more time to reach second base. Additionally, Cruz's 6'7" frame allows for stride lengths that likely outperform the assigned burst tier for "average" runners.

7. Limitations and Future Considerations

While initial validations showed signs of simulation accuracy, there are notable model limitations worth considering.

- **Left-Handed Pitchers (LHP):** When considering a pitcher's ability to 'hold' runners, delivery time does not capture the full picture. Left-handed pitchers have an inherent advantage over righties, with the potential for the mid-delivery pickoff from a lefty arm. Left-handed pitchers may rely less on fast delivery times due to their pickoff advantage, and while delivery times in this simulation are assumed based on overall run prevention metrics, a more robust model should incorporate pitcher handedness.

- **Pitch Selection:** As stated, when reviewing The Cruz Case, the model does not account for pitch type or velocity. While these metrics are encapsulated in delivery time, a more robust model should consider a pitcher's arsenal when predicting stolen base outcomes. For example, a pitcher who features a high dose of mid-70s curveballs is likely easier to steal on than a 100+ mph flamethrower. Ideally, future simulation iterations model delivery time as a conditional variable, where pitch-type is predicted based on situational data, and delivery time is sampled accordingly (e.g., $N(\mu_{fastball}, \sigma_{fastball})$ or $N(\mu_{offspeed}, \sigma_{offspeed})$).

- **Weather:** A true Digital Twin should consider the physical system in its entirety – including weather. A model that considers weather may find that rain results in higher success probability if a pitcher's control is affected, or lower due to poor field conditions for runners.

The expectation is that with increased data granularity, model accuracy is expected to improve. The simulation can account for more factors to truly replicate the physical system on a larger scale. Organizations should look to leverage the raw data provided from Statcast as well as internal metrics if considering implementing their own Digital Twin.

8. Conclusion

This project set out to first explore Digital Twins as a means of simulation and to demonstrate its practical application to professional baseball. As the sport undergoes rapid technological advancement, the opportunity for simulation to support real-time, in-game decision-making continues to expand. Thanks to Statcast's ability to capture the game's physics at a granular level, we can now move beyond static historical analysis and begin to formulate dynamic Digital Twin use cases for MLB organizations.

The stolen base model developed here proves that this paradigm is viable even when restricted to public data. By leveraging historical physics data from Baseball Savant, this analysis demonstrates insights that not only assist with managerial strategy, but tie directly to maximizing Run Expectancy.

Most notably, this tutorial demonstrated the successful application of discrete-event modeling to a game that is inherently continuous. By converting kinematic physics into stochastic service times, we created a lightweight, high-speed decision tool without sacrificing validity. While this model relies on specific proxies for missing variables (such as runner reaction time), expanding this approach to incorporate real-time sensor feeds would further reduce uncertainty. Ultimately, this project illustrates that the future of baseball strategy lies not just in collecting data, but in simulating the future before the pitch is thrown.

9. References

[1] Major League Baseball. (n.d.). *Statcast glossary*. *MLB.com*. www.mlb.com/glossary/statcast

[2] Grieves, M. (2025). Digital twin phases and futures. *Digital Engineering, 6*, 100053. https://doi.org/10.1016/j.dte.2025.100053 (Directory of Open Access Journals)

[3] Attaran, M., & Celik, B. G. (2023). Digital twin: Benefits, use cases, challenges, and opportunities. *Decision Analytics Journal, 6*, 100165. https://doi.org/10.1016/j.dajour.2023.100165 (RWU Docs)

[4] Lemire, J. (2023, June 27). MLB unveils Gameday 3D experience for viewers. *Sports Business Journal*.

https://www.sportsbusinessjournal.com/Daily/Issues/2023/06/27/Technology/mlb-gameday-3d/ (Sports Business Journal)

[5] KWR Water Research Institute. (n.d.). *Digital twins for the water sector*. KWR Water Research Institute. Retrieved November 26, 2025, from https://www.kwrwater.nl/en/projecten/digital-twins-for-the-water-sector/ (KWR)

[6] Kritzinger, W., Karner, M., Traar, G., Henjes, J., & Sihn, W. (2018). Digital twin in manufacturing: A categorical literature review and classification. *IFAC-PapersOnLine, 51*(11), 1016–1022. https://doi.org/10.1016/j.ifacol.2018.08.474 (Publica)

[7] Clemens, B. (2025, May 29). The run expectancy matrix, reloaded for the 2020s. *FanGraphs*. https://blogs.fangraphs.com/the-run-expectancy-matrix-reloaded-for-the-2020s/ (FanGraphs Baseball)

[8] Lindsey, G. R. (1963). An investigation of strategies in baseball. *Operations Research, 11*(4), 477–501. https://doi.org/10.1287/opre.11.4.477 (IDEAS/RePEc)

[9] Delivery time and pop time significance. (2013, March 19). *Baltimore Sports and Life*. https://baltimoresportsandlife.com/delivery-time-and-pop-time-significance/ (baltimoresportsandlife.com)

[10] Major League Baseball Advanced Media. (n.d.). *Baseball Savant Data and Video*. Baseball Savant. https://baseballsavant.mlb.com