Structured Randomness in Baseball: Stochastic Modeling with Machine Learning and Monte Carlo Inference

Evan Goforth

August 27, 2025

Abstract

I introduce a Monte Carlo simulation framework to estimate the probability distributions of the number of hits in a single game for a given hitter and starting pitcher matchup. This model was built under the assumption that while the hitter-pitcher encounters are stochastic, their underlying structure can be modeled deterministically. My approach combines unsupervised and supervised machine learning with hierarchical simulation. First, I applied Gaussian Mixture Models (GMMs) to cluster pitches into perceptually (from the vantage point of the hitter) meaningful subtypes based on velocity and movement features, using information criteria (AIC/BIC) to select the optimal number of clusters. Second, I personalized the importance of these subtypes for each batter by training Random Forest regressors on the expected batting average (xBA), reweighting pitch features according to the performance profile of the hitter. Third, I developed a custom plate appearance simulation engine that models each pitch sequentially using a hybrid of Naïve Bayes classifiers and empirical outcome distributions. While I also experimented with more advanced architectures—including MLPs, RNNs, and XGBoost classifiers—the simpler empirical models consistently performed better from both an accuracy and stochastic perspective (important for Monte Carlo simulations), likely due to the limited number and recency of hitter- and pitcher-specific data points, which makes localized, data-driven methods more statistically robust. The resulting distribution of total hits, derived from a Monte Carlo simulation with 10,000 trials, emerges as a compound distribution (resembling a Poisson- binomial distribution where the number of trials is also a RV)—effectively a convolution of the distribution of opportunities (at-bats) and the distribution of successes (hits). The innovative aspect of this study is to fuse semi-deterministic structure, with defensible rules grounded in domain knowledge with a stochastic Monte Carlo framework. In general, this work was an optimization problem to test the limit of how much predictive signal machine learning can extract from open-source data under real-world sparsity while limiting algorithmically introduced noise, all in the realm of baseball.

Introduction and Motivation:

A central motivation for this work is the recognition that baseball performance is highly matchup-dependent. Hitters are not simply "good" or "bad" in absolute terms; rather, their swings are better suited to some pitch types than others. For instance, batters with flatter swing paths often fare better against four-seam fastballs, while those with more pronounced uppercut swings may match up more effectively against curveballs. The same logic applies to pitchers: a pitcher may not rank highly on leaguewide metrics, but their particular repertoire can neutralize specific hitters. Thus, the key analytical question becomes less about overall skill in isolation and more about the fit between a hitter's swing mechanics, mental approach, and a pitcher's arsenal. This relationship—where performance is conditional on the interplay of batter and pitcher attributes—provides the foundation for my modeling approach.

Modeling the duel between pitcher and batter is a core problem in sabermetrics and sports analytics. Each plate appearance (PA) is the result of a sequence of pitches and decisions, yielding a stochastic outcome (hit, out, walk, etc.). I posit that while outcomes are random, the structure governing these interactions can be treated as deterministic or at least stationary: given the count, pitcher, batter, and pitch characteristics, the probabilities of the next event are reproducible. This view is in line with the Markov chain models of baseball, which assume the game transitions through states (such as count or base/out states) with fixed probabilities. By modeling the sequence-of-pitches structure explicitly and layering on probabilistic outcomes, I aim to capture both the tactical nuance of each at-bat and the overall randomness of baseball results.

Early quantitative analyses of baseball adopted Markov processes to model innings and games (e.g., Lindsey, 1961; Howard, 1963), recognizing that "baseball games are readily modeled using Markov processes". In these models, each pitch or at-bat transition depends only on the current state (such as count) and not on prior history, producing a tractable but powerful representation of the game's deterministic backbone. My approach shares this philosophy of deterministic structure: I assume that if I properly condition on the relevant state (pitch type, location, count, hitter archetype, etc.), the outcome probabilities are essentially fixed. This allows the ability to decompose the complex batter–pitcher duel into modular components that can be learned from data and recombined in simulation.

In what follows, I detail five major components of my methodology, each building upon modern data and techniques:

Core Methodology

1.) Unsupervised Pitch Clustering via GMM: To represent the pitch repertoire in a data-driven way, I cluster pitches into subtypes based on their kinematic characteristics (velocity, horizontal break, vertical break). By doing so, I aim to capture what batters perceive as different pitch "shapes" or types beyond the traditional labels (fastball, slider, etc.). Prior research has shown that classifying pitch types can be challenging because the same pitch label (FF, SL, CH, etc.) can correspond to different trajectories across pitchers, and vice versa. Umemura et al. (2021) applied a variational Bayesian GMM to TrackMan pitch data and demonstrated the value of unsupervised learning in identifying unique pitch groupings. I adopt a similar approach, using a finite Gaussian mixture model to cluster pitches and selecting the number of mixture components by maximizing model fitness criteria (AIC/BIC), while not creating artificial data sparsity due to overclassification. This approach lets the data determine how many distinct pitch sub-categories exist, rather than imposing a fixed number a priori. As noted in an earlier study, model-based clustering with mixtures can "identify the best model and the number of clusters" automatically via Bayesian Information Criterion, avoiding arbitrary choices of cluster count. The result of this step is that each raw pitch is assigned to a GMM cluster (with some probability or hard assignment), effectively creating new pitch subtypes (e.g., a pitcher's four-seam fastball might be split into two clusters representing a "fast, rising four-seamer" vs. an "average speed, slightly tailing four-seamer"). Although the differences in these clusterings are fine, so are the differences in baseball, the game of inches.

2.) Personalized Pitch Subtype Weights via Random Forests: Not all batters respond to pitch types and subtypes in the same way. After clustering pitches, I incorporate hitter-specific performance against those clusters using a supervised learning approach. For each batter (and each broad pitch category), I train a Random Forest regressor to predict the expected batting average (xBA) outcome of a pitch, using the pitch's measured features (velocity, vertical break, horizontal break, and pitch type) as inputs. The use of Random Forests is motivated by their ability to capture nonlinear interactions and to provide measures of variable importance. Breiman (2001) introduced Random Forests as an ensemble of decision trees that yields robust predictions and internal estimates of feature importance. In our context, I leverage these properties to discern which pitch features or clusters most influence the success of a given batter. For example, a Random Forest model might learn that two hitters might post similar overall statistics, yet against four-seam fastballs one may be more sensitive to changes in velocity while the other responds more strongly to variations in vertical break. By training separate GMM models per batter, I personalize the interaction: effectively, each batter now has a custom weighting over the pitch feature space. This step can be seen as a form of "personalized log5" or individualized matchup modeling, analogous in spirit to the Bayesian hierarchical models that allow each player to have their own parameters. Indeed, Doo and Kim (2018) used a hierarchical Bayesian extension of the traditional log5 odds model to estimate matchup probabilities for each batter–pitcher pair, improving predictions by pooling information across players while retaining individual differences. My Random Forest personalization plays a similar role: it uses the data of each batter's historical performance (in terms of xBA) to adjust how much each pitch characteristic matters for that batter.

3.) At-Bat Simulation Engine (Hybrid Naïve Bayes + Empirical Model): Modeling an at-bat requires carefully defining which features matter most. As a former player, I recognize that many factors influence an outcome — pitch type, count, sequencing, and hitter tendencies among them. In designing my simulation engine, I wrestled with how to encode this reality into a tractable set of predictive models that would maximize signal while limiting noisy features.

One of the biggest battles was over sequencing. Pitching coaches place great emphasis on pitch sequences to disrupt hitter timing, and intuitively, the previous series of pitches carries signal about the next pitch. My initial goal was to capture this sequential information using a recurrent neural network (RNN). However, the data available for individual pitcher–hitter matchups was too sparse to allow for effective training without sacrificing predictive stability elsewhere. In the spirit of balancing signal and noise, I made the pragmatic choice to omit explicit sequence history as a feature. While I concede that sequencing plays a role in reality, the cost of modeling it directly outweighed the benefit under data constraints.

Instead, I defined features module by module, ensuring that each model only used features that were absolutely necessary. When unsure whether a feature added value, I tested it empirically by comparing model performance with and without the feature using Macro F1 and Micro F1 scores. Ultimately, three features proved most useful across modules: pitch cluster, pitch zone, and hitter archetype. Many other candidate features were discarded to reduce noise and avoid diluting the predictive signal.

The only feature I constructed independently, meaning not already pre-packaged in the data, was hitter archetype. The hitter archetype feature is designed to capture the manner in which pitchers alter their approach depending on hitter profile. Archetypes were defined along three axes: (i) patient vs. aggressive (tendency to chase), (ii) power vs. contact (likelihood of extra-base damage), and (iii) elite vs. non-elite (overall potency measured by wOBA). Patient hitters reduce the utility of chase pitches, power hitters force pitchers to avoid high-damage zones, and elite hitters often require tailored, weakness-specific strategies. These categories were built using statistical thresholds and grounded in baseball intuition.

With these features established, I divided the at-bat simulation into four predictive modules. Each module outputs a probability distribution for the next state, and together they form a probabilistic state machine that plays out each pitch until the plate appearance ends in a terminal outcome (hit, out, walk, etc.).

a.) Pitch Selection Module: The first module predicts which pitch cluster will be thrown. I implemented a hybrid Naïve Bayes / stochastic lookup approach conditioned on the current count and hitter archetype. This reflects realistic tendencies such as fastballs in hitter's counts or breaking balls with two strikes. Pitch types are pre-clustered into groups earlier in the pipeline, ensuring that the model works with consistent categories.

b.) Pitch Zone Module: Conditioned on the pitch cluster as well, the Pitch Zone Module predicts the zone in or around the strike zone to which the pitch will be located. This model also follows a Naïve Bayes / lookup hybrid and is conditioned on pitch cluster, count, and hitter archetype. Separating type and location into distinct modules improves interpretability and flexibility, while still allowing their interaction to be captured through conditional dependencies.

c.) Pitch Outcome Module: The Pitch Outcome Model determines the result of a pitch once it is thrown. Possible outcomes include ball, called strike, swinging strike, foul, hit-by-pitch (HBP), or ball in play (BIP). Using pitch cluster and count as inputs, the model applies Naïve Bayes and empirical lookup distributions calibrated from historical data. Each pitch thus transitions the simulation state: the count updates if the plate appearance continues, or the sequence terminates if the pitch is put in play or results in a strikeout/walk.

d.) XBA Module: The Expected Batting Average (xBA) Model provides a hit probability for balls in play. Due to sparsity in the data, I employed a tiered lookup table rather than a more complex statistical model. The lookup first searches for xBA given the full feature set (count, zone, and pitch cluster). If insufficient data exists, it falls back to (zone + pitch cluster), and if necessary, to the global xBA for all hitters given the feature set. This ensures every BIP has a probabilistic outcome while grounding estimates in data-rich contexts when possible.

4.) Hierarchical Backoff and Quality Gates: One challenge in this fine-grained simulation is data sparsity. The model needs probabilities for very specific scenarios (e.g., a 2-2 slider on the low-outside corner thrown by a particular pitcher to a particular batter). To handle this, I implement a hierarchical backoff strategy inspired by techniques in language modeling and Bayesian shrinkage. At the most specific level, if I have sufficient data for a given situation (say, batter X seeing a pitch of subtype Y in zone Z at count C), I use those empirical probabilities. If not (data too sparse), I "back off" to a more general level – for example, use data for all batters against pitch subtype Y in zone Z at count C, or batter X against any pitch in zone Z, etc. I establish quality gates (thresholds of data quantity or quality) that determine when to trust a fine-grained probability versus when to revert to a broader distribution. This hierarchical approach is analogous to the Bayesian hierarchical models that use partial pooling: it preserves specific effects when data support them and borrows strength from the aggregate when they do not. For instance, our model might know that batter X, a notorious fastball hitter, has a high contact rate on high fastballs specifically – if enough samples validate this – but if batter X has never seen a rare pitch subtype from a certain pitcher, the model will fall back to league-average outcomes for that subtype. By combining empirical frequency tables at multiple levels of granularity, I ensure the simulation produces plausible outcomes even for uncommon scenarios. This is crucial for realism: the engine won't overfit to noisy small-sample stats, and it won't entirely miss the unique tendencies of specific players. Notably, this approach resonates with the methodology of Doo and Kim (2018), who extended the traditional log5 model by treating the fixed coefficients as prior information in a Bayesian framework, thus "retaining the advantages of [simple models] while complementing their disadvantages" in data-sparse matchups. Similarly, our quality-gated backoff ensures that when detailed data is lacking, the model's behavior defaults to reasonable priors (broader averages or context-free probabilities).

5.) Termination Logic: The pitch-by-pitch simulation continues until the plate appearance ends. I explicitly simulate events like balls and strikes accruing (with walks occurring after four balls, strikeouts after three strikes without contact, noting that foul balls with two strikes do not count as strikeouts). If a ball is put in play, I determine if it results in a hit or an out. Here, I incorporate the hitter's personalized hit probability (xBA) for that specific pitch, as estimated by the stochastic lookup table. In other words, when contact occurs, I treat the probability of a hit as $p_{\text{hit}} = \text{xBA}(\text{batter, pitch features})$ from the personalized model. I then draw a Bernoulli outcome: with probability p_{hit} it's a hit (single, or I could even extend to extra-base hit probabilities), otherwise it's an out in play. This usage of xBA ensures that the quality of contact (or likelihood of success) reflects the match-up and pitch attributes: a well-located, effective pitch will have a low xBA, whereas a hanging curveball might have a high xBA for a good hitter, and the simulation will produce hits accordingly. If an out or hit occurs, the PA terminates; if a walk or strikeout occurs, it terminates as well.

Full-Game Monte Carlo Simulation and Compound Distribution of Hits:

I scale from the at-bat engine to full games by allocating plate appearances (PAs) between the opposing starter and bullpen, rather than simulating inning states. The flow is:

a.) Starter longevity (IP | opponent). I model the starting pitcher's innings pitched (IP) via OLS:

$$\mathbb{E}[\text{IP}] \;=\; \beta_0 + \beta_1 \cdot \text{wOBA}_{\text{opponent}}\,.$$

This captures mean reversion in IP and ties the only easily modeled systematic driver (opponent quality) to starter longevity. In each simulation, I sample IP around the OLS prediction using an empirical SD (truncated to $[0, 9]$ in regulation).

b.) Batters faced (BF | IP). Conditional on the sampled IP, I model batters faced with a Poisson regression. Equivalently,

$$\log \mathbb{E}[\text{BF} \mid \text{IP}] \;=\; \gamma_0 + \gamma_1 \cdot \text{IP} \quad\Longleftrightarrow\quad \mathbb{E}[\text{BF} \mid \text{IP}] \;=\; \exp\Big(\gamma_0 + \gamma_1 \cdot \text{IP}\Big),$$

and I draw a stochastic BF accordingly. This yields a simulation-specific pair (IP, BF) for the starter.

c.) Map BF to a hitter's PAs vs the starter (lineup arithmetic). Given lineup slot $s \in \{1, \ldots, 9\}$ (with $s = 1$ the leadoff), the hitter's plate appearances against the starter are computed without inning states as

$$\text{PA}_{\text{vs SP}} \;=\; \max\left\{0,\; \left\lfloor \frac{\text{BF} - s}{9} \right\rfloor + 1\right\}.$$

This counts how many times slot s appears among the first BF batters.

d.) Bullpen innings and batters faced (Monte Carlo). Let the pitching team be T with home/away indicator $H \in \{0, 1\}$ ($H = 1$ if home). Let IP_{SP} be the starter's innings pitched (measured in thirds so $3\,\text{IP}_{\text{SP}}$ is an integer), and let $X \geq 0$ denote any extra defensive innings (set $X = 0$ if extras are not modeled). If $H = 0$ (away team pitching), draw

$$I_9 \sim \text{Bernoulli}(\pi_9), \qquad \pi_9 \;=\; \frac{w_T}{w_T + w_O};$$

if $H = 1$ (home), set $I_9 \equiv 1$. Define the team's defensive workload as

$$\text{IP}_{\text{team}} = \begin{cases} 9 + X, & H = 1, \\ 8 + I_9 + X, & H = 0. \end{cases}$$

Then bullpen innings and outs are

$$\mathrm{IP}_{\mathrm{BP}} \;=\; \max\{\,0,\, \mathrm{IP}_{\mathrm{team}} - \mathrm{IP}_{\mathrm{SP}}\,\}, \qquad O_{\mathrm{BP}} \;=\; 3\,\mathrm{IP}_{\mathrm{BP}}.$$

Each team T has a discrete empirical distribution $\mathcal{D}_T$ on $\{1, 2, \dots\}$ for "batters faced to record one out." Let $K_j^{(\cdot)}$ be i.i.d. draws from $\mathcal{D}_T$. Then

$$\mathrm{BF}_{\mathrm{SP}} \;=\; \sum_{j=1}^{3\,\mathrm{IP}_{\mathrm{SP}}} K_j^{(\mathrm{SP})}, \qquad \mathrm{BF}_{\mathrm{BP}} \;=\; \sum_{j=1}^{O_{\mathrm{BP}}} K_j^{(\mathrm{BP})}.$$

Let $s \in \{1, \dots, 9\}$ be the hitter's lineup slot ($s = 1$ leadoff). Plate appearances versus the starter:

$$\mathrm{PA}_{\mathrm{vs\,SP}}(s \mid \mathrm{BF}_{\mathrm{SP}}) \;=\; \max\left\{\, 0,\, 1 + \left\lfloor \frac{\mathrm{BF}_{\mathrm{SP}} - s}{9} \right\rfloor \right\}.$$

The first batter to face the bullpen is

$$r \;=\; 1 + \Big(\mathrm{BF}_{\mathrm{SP}} \bmod 9\Big) \;\in\; \{1, \dots, 9\},$$

and the cyclic distance from that batter to slot s is

$$d(s, r) \;=\; (\, s - r\,) \bmod 9 \;\in\; \{0, \dots, 8\}.$$

Over the next $\mathrm{BF}_{\mathrm{BP}}$ batters, plate appearances versus the bullpen are

$$\mathrm{PA}_{\mathrm{vs\,BP}}(s \mid \mathrm{BF}_{\mathrm{SP}}, \mathrm{BF}_{\mathrm{BP}}) \;=\; \max\left\{\, 0,\, 1 + \left\lfloor \frac{\mathrm{BF}_{\mathrm{BP}} - 1 - d(s, r)}{9} \right\rfloor \right\}.$$

Finally, total plate appearances:

$$\mathrm{PA}_{\mathrm{total}}(s) \;=\; \mathrm{PA}_{\mathrm{vs\,SP}}(s \mid \mathrm{BF}_{\mathrm{SP}}) \;+\; \mathrm{PA}_{\mathrm{vs\,BP}}(s \mid \mathrm{BF}_{\mathrm{SP}}, \mathrm{BF}_{\mathrm{BP}}).$$

e.) At-bat simulation (per PA). For each PA, I invoke the at-bat engine with the appropriate pitcher context. The engine uses pitch-type and location models, an outcome model (ball/strike/foul/HBP/BIP), and an xBA model on balls in play to determine a *hit* vs. *no hit*. Because this study is hits-focused, I do not simulate baserunning or run states. This at-bat simulation is only for the PA against the SP. For PA against the RP, due to the fact that it is quite difficult to predict exactly which RP will be used in game, I simply used the xHits/PA to stochastically model the number of hits per PA assuming that the hit distribution converges to the hitter's global mean against the entirety of the bullpen. This is an example of opting for simplicity, since modeling each bullpen pitcher versus the hitter would be quite complex and noisy.

f.) Game & season Monte Carlo. A single game draw yields a hit count for the target hitter. Repeating across many simulations produces the distribution of total hits, from which I compute probabilities (e.g., $P(\text{hits} \geq 1)$). The same setup scales to multiple games for season-level summaries. Optionally, include extra-innings probability to add another PA cycle when tied. This process for factoring in the presence of extra-innings was mentioned above.

This formulation removes inning-by-inning loops, ties bullpen exposure directly to sampled starter longevity, uses OLS(IP $\sim$ wOBA$_{\text{opponent}}$) and Poisson(BF | IP), and computes the hitter's PAs vs the starter by lineup arithmetic with the remainder vs the bullpen.

An important insight from our simulation is that the total hits in a game follow a compound distribution that arises from the random nature of opportunities and successes. In simpler terms, the number of hits H can be seen as the sum of hits in each at-bat, and the number of at-bats itself is a random variable depending on how innings progress (more hits and walks can extend innings, while quick outs shorten them). Formally, I might write:

$$H = \sum_{i=1}^{N} X_i,$$

where N is the number of official at-bats in the game (itself random) and X_i is an indicator (1 if hit, 0 if out) for at-bat i. Even if I assume a constant number of plate appearances N, H is a Poisson-binomial or (if p is constant and N fixed) a Binomial(N, p) random variable. N has its own distribution driven by the game context. Due to the fact neither N nor p is fixed, the distribution of H is therefore a mixture over N — a classic compound distribution. Prior analytic work has tackled a similar problem for runs scored. Rosner, Mosteller, & Youtz (1996), for example, modeled the distribution of runs per inning as a convolution of two components: the distribution of the number of batters faced in the inning, and the distribution of runs given a certain number of batters. They found that the number of batters faced by a pitcher in an inning was well-fit by a (slightly adjusted) negative binomial distribution, while the runs-scored given that many batters could be modeled by a truncated binomial whose success probability varies with the number of batters (to account for diminishing returns of loading the bases, etc.). In our context of hits, an analogous statement can be made: the total number of at-bats (or plate appearances) in a game can be modeled (for a given team offense and opposing pitching) by some distribution – often roughly Poisson or negative binomial due to the structure of outs and innings. Then, given a certain number of at-bats $N = n$, the number of hits H is binomial with parameters n and the average hit probability in those at-bats (which may not be constant if the lineup's hitting probability changes as the game goes on, but one could integrate over that). The overall distribution of hits is "compound" because it is the result of this two-stage random process (random N, and random hits given N). Our Monte Carlo simulation naturally captures this compound randomness: in some simulated games, pitchers might face the minimum 27 batters (yielding lower H typically), while in others, extra batters come up due to hits and walks, extending the game and giving more opportunities for hits.

The Monte Carlo approach allows us to bypass making strict analytical assumptions about these distributions; instead, I can empirically observe the simulated distribution of H. Some aspects of the hitter-pitcher dynamic are much harder to explicitlty model, but can be indirectly encoded into the Monte Carlo simulation. The flexibility of Monte Carlo lets us incorporate such complexities that are hard to handle in closed-form.

Backtest Proxy

Motivation. A full outcomes backtest—mapping simulated hit-count PMFs to realized box scores over many games—would be ideal but is costly in data engineering and compute. As a practical alternative, I use *published two-way "batter hits" lines* as an external probability benchmark. The aim is not to reproduce those lines, but to check that the model's probabilities are close to a sophisticated external forecast while preserving independent structure. Essentially, let these external forecasts serve as a trusted benchmark as to what the simulated model probabilities should be approximately.

Protocol. For each hitter–pitcher–date, collect two-way lines at common thresholds (0.5, 1.5; occasionally 2.5). Convert quotes to implied probabilities (q_O, q_U) and remove overround (vig) by proportional renormalization:

$$p_O^* = \frac{q_O}{q_O + q_U}, \qquad p_U^* = \frac{q_U}{q_O + q_U}.$$

Map the simulation PMF $P(H = k)$ to line-level probabilities:

$$\text{Over } 0.5: \ P(H \geq 1), \quad \text{Over } 1.5: \ P(H \geq 2), \quad \text{Under } 0.5: \ P(H = 0), \quad \text{Under } 1.5: \ P(H \leq 1),$$

and compute per-quote error $\Delta = p_{\text{model}} - p^*_{\text{line}}$.

Results:

N **(quotes)**	**MdAE**	**MAE**	**RMSE**	**Share** $\leq$ **10pp**	**Bias (mean)**
213	0.033	0.0405	0.0526	0.93	+0.0022

Bias (median)	$q_{0.05}(\Delta)$	$q_{0.95}(\Delta)$	**Tail diff.**
+0.0048	−0.0888	+0.0891	0.0003

Analysis: Because published lines embed microstructure, exact equality is neither expected nor desirable. What matters for face validity is (i) consistently small, robust errors; (ii) near-perfect symmetry around zero; and (iii) deviations that make clear baseball sense. With MdAE and MAE both below 0.05, differences at the event-probability level are uniformly small. The bias—both median and mean—< 0.005 is compelling evidence of symmetry about the center, indicating no material systematic skew in either direction. Moreover, $|q_{0.95} - q_{0.05}| < 0.001$ is a stringent indicator of tail symmetry as well. Under realistic time/compute constraints, this constitutes a credible *proxy backtest*: the simulator is close to where it should be and different where domain logic predicts.

Takeaway: The proxy comparison shows a bottom-up, pitch-level simulator that is *plausibly aligned* with an external probability benchmark (MdAE $\approx$ 3.3 pp; 76% within 5 pp; 93% within 10 pp; symmetric tails) while preserving independent, interpretable structure.

Conclusion

I developed a simulation-first, pitch-level framework to estimate the full distribution of *hits* for a *single hitter* in a given pitcher-hitter matchup. The design fuses semi-deterministic structure (quality-gated backoff, lineup arithmetic) with a stochastic Monte Carlo engine to extract reliable signal from sparse, open-source data while controlling variance. A key innovation is *hitter-tailored pitch clustering*: Gaussian Mixture Models (AIC/BIC-selected) define perceptually meaningful subtypes, then supervised reweighting (Random Forests) personalizes those subtypes to each batter's response profile. A modular at-bat simulator (pitch selection, location, outcome, and xBA-on-contact) converts pitch context into calibrated hit/no-hit probabilities, yielding a compound distribution of total hits across simulated plate appearances. The result is an interpretable, deployable methodology that is rigorous enough for research and pragmatic enough for decision support.

Next steps and uses.

- **Validation.** Compare simulated player-level PMFs to realized game outcomes (per hitter, per date) to assess calibration and sharpness; expand beyond proxy checks to full empirical backtesting over rolling windows.
- **Applications.** (i) Pre-game *hitter–starter* matchup evaluation; (ii) scouting and player development via *cluster-targeted* strengths/weaknesses; (iii) long-term guidance for either approach or swing adjustments based on pitch subtypes most associated with success or failure.

This model and the probabilistic framework that helped create it began from a realization that there is a deterministic structure that underlies all pitcher and hitter matchups. This is something I've seen (from the perspective of a hitter) and many baseball people will tell you that they know the guys they are excited to face and they know the guys they aren't. Long story short, there's a reason for that. Each step of this model aimed to capture all of the signal and determinism that exists from open-source pybaseball Statcast data. This combination of determinism and signal was then implemented into a stochastic Monte Carlo framework to let the determinism appear, while accounting for the inherently random parts of baseball such as a pitcher missing their spot or a hitter just missing their favorite pitch. There is still much work to be done, but this is an addition to the current work being done to find the answer to the optimization problem of how much predictive signal machine learning can extract from open-source data under real-world data sparsity while limiting algorithmically introduced noise.

References

Albert, J. (2006). Pitching, hitting, and scoring: A Bayesian approach to understanding baseball. *Journal of Quantitative Analysis in Sports, 2*(1). https://doi.org/10.2202/1559-0410.1007

Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Doo, W., & Kim, H. (2018). Modeling the probability of a batter/pitcher matchup event: A Bayesian approach. *PLOS ONE, 13*(10), e0204874. https://doi.org/10.1371/journal.pone.0204874

Howard, R. A. (1963). Dynamic programming of baseball. *Operations Research, 11*(4), 537–549. https://doi.org/10.1287/opre.11.4.537

Lindsey, G. R. (1961). The progress of the score during a baseball game. *Journal of the American Statistical Association, 56*(296), 703–728. https://doi.org/10.1080/01621459.1961.10482115

Rosner, B., Mosteller, F., & Youtz, C. (1996). Modeling pitcher performance and the distribution of runs per inning in Major League Baseball. *The American Statistician, 50*(4), 279–291. https://doi.org/10.2307/2684938

Umemura, K., Yanai, T., Nagata, Y. (2021). Application of VBGMM for pitch type classification: Analysis of TrackMan's pitch tracking data. Japanese Journal of Statistics and Data Science, 4(1), 41–71. https://doi.org/10.1007/s42081-020-00079-8