Building a Context-Neutral Player Value Model for Dynasty League Baseball

An MMDBL Advanced Analytics Department White Paper

Why build this at all

Dynasty League Baseball (DLB) resolves everything on cards and charts: a hitter's outcome against a given pitcher lives in a fixed 0-499 roll space (3 10-sided dice), a pitcher's in the 500-999 range, and the two combine – along with fielding, baserunning, and park charts – into what actually happens on the tabletop game field. That's a strength – every outcome traces to a printed number – and also the reason a simple "what's this guy worth" question doesn't have an obvious answer. A .280 hitter with no power on a team that runs a lot isn't obviously more or less valuable than a .240 slugger who never takes an extra base, and neither projection systems built for Major League Baseball (MLB) nor real-world Wins Above Replacement (WAR) translate cleanly onto a dice-and-chart universe with its own steal mechanics, its own error rates, its own player card probabilities, and its own distance hit tables, to name just a few of the game's commonly referenced artifacts.

This paper documents a value model built by the MMDBL Advanced Analytics Department (MAAD) to answer that question anyway: one number per player, denominated in runs above or below a Dynasty League Baseball-average replacement at the same position and role, built up from the actual card mechanics rather than borrowed from MLB. The short version of the approach is old and well-worn in sabermetrics – solve the run-expectancy matrix, derive linear weights, hold everything except the skill you're measuring at league average, and add the pieces back together. The long version — and the more interesting one – is what happens when that approach is applied to every roll in a DLB game, with every roll representing a new probability to factor into the run expectancy equation. To accomplish this required the input of all player card data points, the outcomes of all charts and tables referred to in the process of game play and, of course, the rulebook which clarifies items not fully resolvable on the cards and charts themselves. We know that a player who hits .300 is a better hitter than one who hits .250. And we know that a player who hits 30 homers has more power than one who hits 15. These tried-and-true counting stats tell some of the story but not all of it. Baseball value is the sum of the parts, and the parts include everything a player might do to add value – or subtract it – in a DLB game and over the course of a DLB season.

This paper, and the modeling story it tells, is intended to crack the proverbial code and guide DLB managers to a fulsome understanding of player value within the context of the DLB tabletop and game. The model is built on the latest player cards, charts and rulebook published as of February 1, 2026, so if the online game includes probabilistic outcomes not aligned with these game artifacts, then there may be variations to its applicability. That said, everything the company says about the online game on social media suggests the online version is fully aligned with the tabletop version. If that's the case, then the modeling approach and its output should apply uniformly to online gamers.

Despite what the model output may suggest, the decisions made by a team's GM in constructing a roster, and those made by the manager on game day, have profound impact on game and season

outcomes. A well-constructed model can point you to the best players but it is up to the GM and manager – one in the same in DLB leagues – to make it all come together.

A brief history: from box scores to run expectancy

None of the methods in this paper are new. They're about 60 years old, and the story of where they came from is worth telling; both because it explains why this particular approach was chosen and because every serious DLB gamer building their own evaluation system is standing on the same ground.

Baseball stats have been counted since the game has been played — Henry Chadwick invented the box score in the 1850s, and batting average, RBIs, and pitcher wins were the standard measures of value for a century afterward. The trouble, which took a surprisingly long time for anyone to formalize, is that these counting stats don't actually answer the question a GM or manager cares about: how many runs does a given event add to my team's expected total, in the specific situation it happens in? A double with a runner on second scores a run immediately; the same double with the bases empty doesn't. Batting average treats both swings identically.

The first person to answer that question rigorously wasn't a baseball insider at all. George Lindsey was a Canadian defense operations researcher with a passion for the game, and in the late 1950s he and his father sat through hundreds of games – in person, by radio, by television – charting exactly what happened from every combination of baserunners and outs. In 1963 he published the results in the academic journal *Operations Research* in a paper titled *"An Investigation of Strategies in Baseball."* Buried in it was the first known table of expected runs scored for an inning, broken out by all 24 combinations of baserunners and outs – the exact structure this paper's model calls RE24, explained in the next section. Lindsey used it to answer practical questions (when is a sacrifice bunt worth it, when is an intentional walk correct) that managers had been arguing about on instinct for decades.

Lindsey's work sat mostly unnoticed outside a small circle for another two decades. Pete Palmer, a statistician who did much of his early analysis on IBM punch cards during off-hours at his day job, picked up the same underlying idea in the 1970s and pushed it further: instead of just building the run-expectancy table, he used it to derive a fixed run value for *every* offensive event – a walk, a single, a stolen base, an error – so that a player's entire seasonal line could be converted into one number, runs above or below average. In 1984, Palmer and the historian John Thorn published *The Hidden Game of Baseball*, which named this approach *Linear Weights* and made the case, at length, for why it described the game better than batting average or RBIs ever had.

Running in parallel – and eventually becoming the public face of all of it – was Bill James. Starting in 1977, James, at the time working a night-watchman job, began self-publishing an annual booklet called the *Baseball Abstract*, arguing with essay after essay that the stats everyone used to judge players didn't measure what people thought they measured. Around 1980 he coined a name for the whole enterprise: sabermetrics, built from the acronym of the Society for American Baseball Research (SABR, founded in 1971), which he defined simply as "the search for objective knowledge about baseball." James, Palmer and researcher Dick Cramer had already co-founded SABR's

Statistical Analysis Committee in 1974, and through the 1980s their combined work – Lindsey's run expectancy, Palmer's linear weights, James's relentless questioning of the traditional stat line – became the shared foundation of a field that had mostly been individual researchers publishing in isolation.

For another twenty years this stayed a niche interest for a certain kind of obsessive fan. That changed in 2003 with Michael Lewis's best-selling book *Moneyball*, which told the story of the Oakland Athletics using advanced statistical evaluation principles assessing player value to compete against teams with three times their payroll. The A's applications of these methodologies made the business case for front office adoption in a way academic papers hadn't, and within a decade every organization in the sport had analytics departments built on the same theoretical foundation. Today, sites like FanGraphs and Baseball Reference publish wRC+, WAR, and FIP as ordinary, everyday numbers. All three trace directly back to Lindsey's run-expectancy table and Palmer's linear weights, just refined and re-derived many times over.

This model is doing exactly what Lindsey and Palmer did, pointed at a different target. Instead of building the run-expectancy table from real major-league play-by-play, it builds the identical 24-state structure from DLB's own data – the same dice, the same cards, the same rulebook and charts every Dynasty gamer already owns – and derives its own linear weights from that. Every number in this paper is a direct descendant of a method a Canadian defense researcher worked out listening to games on the radio in 1959, just pointed at a different manifestation of baseball.

The model's output explained in simplest terms

Before digging into the modeling effort, let's level-set on what the output is actually designed to show. The core concept is that each player adds or subtracts to their respective team's ability to score runs or prevent runs.

For position players, three skillsets drive this contribution – 1) hitting, 2) baserunning, and 3) fielding. For each position player – versus lefties, versus righties, and overall –the model outputs a Hit Score, a Run Score and a Field Score. The Hit Score and Run Score are added, and then weighted based on their real-life MLB usage, to produce an Expected Offensive Contribution metric. The ratio of plate appearances versus lefties and righties, respectively, is proportional to the MLB average for the hitter's handedness.

These last two points are important for contextualizing the output – overall usage and where that usage is deployed – is hard-wired based on MLB averages. The model does not attempt to weight optimal usage and deployment that, ideally, a player's DLB manager will pursue. Expected Offensive Contribution, like MLB's Wins Above Replacement (WAR) metric, is a cumulative measure. WAR baselines to a replacement player, whereas this model's Expected Offensive Contribution baselines to the DLB context-neutral average in which a player neither adds nor subtracts from their team's value. In other words, the baseline is zero, which is well-above MLB replacement player value. Finally, the position player evaluation is completed by adding Field Score to Expected Offensive Contribution to derive Total Expected Contribution.

For pitchers, three skillsets drive their individual contribution to run prevention – 1) stuff and command, 2) run game control, and 3) fielding. Stuff and command is captured in the Pitch Score which, as for hitters, is decomposed into a score versus lefthanders and a separate score versus righthanders. These scores are combined and weighted based on MLB usage and MLB-average deployment based on handedness to produce the Stuff & Command Contribution metric. This is the foundational metric for pitchers as it answers, "how well does the guy pitch?" Separately, a Run Game Control Score and a Field Score are calculated for each pitcher and added to Stuff & Command Contribution to produce Total Expected Contribution.

You probably noticed the ultimate metric for both position players and pitchers is named "Total Expected Contribution." This is by design so that position players and pitchers can be compared to determine who can be expected to contribute the most to a team's success. You probably also noticed that everything is additive. This also is by design so that it is easy to decompose Total Expected Contribution into its component parts and determine what's driving a player's value – or lack of value. For instance, Bobby Witt Jr. is an excellent hitter. But in the context of DLB, his baserunning composes 5.4% of his overall value and his defense accounts for 18.3% of Total Expected Contribution. That's roughly 24% of Witt's total value attributable to something other than his bat!

The appendix to this paper provides tables for the top 20 position players and top 20 pitchers across a number of these score components, and it also provides the MLB team rankings for the DLB 2025 card set. The ultimate team metric is Team Talent Score, which enables a DLB manager to say, "my players are better than your players." While this is all well and good, you'll want to read the "How much does the manager matter?" section beginning on page 18. As the saying goes, "hard work beats talent when talent doesn't work hard."

The engine: run expectancy from a 24-state Markov chain

Every baseball value model needs an answer to the question "how many runs is this event worth," and the answer depends on what state the game is in when the event happens — a double with the bases empty is worth less than the same double with a runner on second, because the second case drives in a run immediately. The standard sabermetric tool for handling this is a run-expectancy table: the average number of runs a team scores for the rest of the inning, starting from each possible combination of baserunners and outs. There are eight baserunner combinations (empty, first only, second only, and so on through bases loaded) and three out counts, for 24 total combinations – hence "RE24," shorthand in the sabermetric world for Run Expectancy across those 24 states. Once every state has a run-expectancy value attached to it, any event's true worth is just the *change* in expectancy it causes, averaged across every situation it could occur in – a walk with the bases empty and a walk with the bases loaded move the game to different states worth different amounts, and a proper linear weight for a walk blends across all of them.

DLB uses that same 24-state framework, but rather than importing MLB's real-world run values, this model builds the actual transition matrix from DLB's own validated data – hit distributions, range and error resolution, baserunner advancement, catcher and pitcher influence on stolen bases – and solves the RE24 table from scratch. Every player's value is then computed as the runs-above-

average impact of his own specific card, run through weights derived from *this* league's actual chart and rulebook mechanics.

This matters more than it sounds like it should, because it's the difference between *assuming* a double is worth about 0.7 runs (a reasonable estimate from real baseball) and *deriving* that a double is worth 0.63 runs in this specific tabletop game, given this league's actual distribution of who's on base when doubles happen and how DLB's own charts resolve them.

The weights that came out of the solve, for the run environment against right-handed pitching:

Event	Runs above average
Home Run	+1.46
Triple (gap)	+0.95
Double (gap, all fields)	+0.79
Double (line/wall)	+0.63
Single (ground, advances two)	+0.51
Single (ground/line)	+0.34
Walk / Hit By Pitch	+0.32
Long flyout	−0.19
Groundout (soft)	−0.23
High flyout	−0.25
Lineout / popout / strikeout	−0.25
Hard groundout	−0.33

That last row is the most interesting to a scout who's never opened a run-expectancy table: a hard groundout is the single worst outcome in the model, worse than a strikeout, because it's the band where a double play actually happens if there's a runner on first with fewer than two outs. A player's card doesn't just tell you how often he makes outs – it tells you *what kind*, and the kind matters.

The league's overall run environment came out to 4.23-to-4.36 runs per nine innings depending on which side of the platoon split you're looking at – in the range you'd expect for a card-driven simulation calibrated against modern offensive levels.

Five components, one number

The model prices five distinct skills and adds them together, and the addition is the whole point. Everything is computed *context-neutral* – a player's own card against a league-average opponent, league-average fielders, league-average baserunners, at a league-average park – which means every component isolates exactly one thing a player brings to the table, holding everything else fixed at zero. That's a real tradeoff (it won't tell you what a specific player did against a specific real opponent in a specific park), but it's the right one for a value model: you want to know what a player is worth in general, not what he happened to do last Tuesday.

Overall score components were discussed earlier, but let's take a closer look:

- **Hit Score / Pitch Score** — The batter's or pitcher's own card, run through the linear weights above, expressed per 650 plate appearances (PA) or nine innings pitched (IP). This is "traditional" offensive and pitching value in the RE24 sense, split by platoon side since DLB cards are.
- **Run Score** — A hitter's baserunning value, itself two things bolted together: how often he takes an extra base on somebody else's hit (driven by his Run rating and the league-average outfielder's arm), and how much value his stolen-base threat adds. Both pieces turned out to be much harder problems than they looked, and each gets its own section below.
- **Run Game Control (RGC) —** The pitching and catching mirror of Run Score: how much a pitcher's Hold rating and a catcher's arm suppress the running game, relative to a league-average battery, plus — new in Version 24 — a Wild Pitch component that shares its chart with Passed Ball.
- **Field Score —** Error rate, range, and (for outfielders) arm, each isolated the same context-neutral way; catchers get an additional Handling component, a steal-suppression arm component that lives in the same framework as Run Game Control, and — new in Version 24 — a Passed Ball component and a throwing-error-on-steal component, both described below.
- **Total Expected Contribution** — The sum. A player at exactly zero performs at league average in every one of these dimensions — not replacement level, not a bad player, exactly average. That distinction matters more than it sounds like it should; a lot of confusion about "why is this good pitcher showing such a low result" traces back to conflating zero with bad, and it's worth stating the definition plainly rather than assuming it's obvious.

Deep Drive: what a power grade actually buys you

Whether a fly ball clears the fence is the highest-leverage single resolution in the model – a home run is worth 1.46 runs and a long flyout is worth –0.19, the single widest gap on the whole card – and it took real work to get it right. The naive version of this mechanic is a flat conversion rate: any "Deep Drive" result becomes a home run some fixed percentage of the time, independent of the batter's power grade. That version existed briefly in this build, at a flat 40%, and it was wrong in a specific, findable way: it meant an F-power hitter and an A-power hitter turned the same near-miss fly ball into a home run at the same rate, which is obviously not how the printed Distance Hit and Deep Drive Location charts work.

The fix pulls the real charts – distance-hit distributions by power grade, deep-drive landing sectors by batter type (pull vs. spray, left vs. right) – and resolves each batter's own power grade against his own spray tendency, at a specific park's actual wall distances.

The corrected conversion curve:

Power Grade	P(Deep Drive → Home Run)
A	39.0%
B	22.3%
C	10.6%
D	3.7%
F	0.4%

An F-power hitter converts a deep drive into a home run roughly one time in 250. Under the old flat-40% assumption, he'd have done it two out of five times – a wild overstatement of the little guy's power that would have quietly inflated Hit Score for every low-power player in the league.

The park itself went through its own revision. Every one of those numbers above was originally computed at Comerica Park, the first stadium chart input in the early versions of this build – a reasonable practical default, but not a neutral one, and Comerica specifically leans pitcher-friendly with unusually deep gaps. The model now runs against a **Dynasty League Average Ballpark**: wall distances averaged, unweighted, across all 30 real MLB parks (so no single team's park quirks bias the evaluation, and no over-represented team in the data pool skews the "average" either). The averaged park is smaller than Comerica in almost every sector – left-center averages 397 feet against Comerica's 415 — which is exactly the direction you'd expect given Comerica's real-world reputation, and it moved every power-dependent number in the model by a small, real amount when it went in.

The steal-mechanics story

This is the part of the build that took the longest, went through the most complete rewrites, and is worth walking through in full, because the failure mode it uncovered generalizes past this specific model – and, as it turned out, generalized to two other mechanics inside this same model, covered later in this paper.

The anomaly. Hunter Brown's Run Game Control Score came back at –18.50 runs for a season, which is worse than replacement-level pitching, on a card that had produced a legitimately Cy Young-caliber season. That's not a small discrepancy to wave away; a –18.50 swing on a single secondary skill is larger than most players' entire seasonal contribution. The obvious hypothesis was a coding bug in how his specific ratings were being read. That hypothesis was wrong, and the actual answer took three separate rounds of debugging to find.

Round one: the baseline was wrong, but not by enough. The original attempt-frequency model used the single most common Hold grade in the league (B+) as the "league average" reference point for how often runners even attempt a steal. But the league's Hold-grade distribution isn't tightly clustered around B+ – it's spread out, with meaningful shares at every letter grade – so comparing Hunter Brown's actual C grade against a B+ reference overstated the gap. Switching to a

properly innings-pitched-weighted average across the real distribution closed part of the gap (his season total improved to roughly –13), but not nearly enough, and the fix didn't generalize: it was patching the reference point without touching the underlying mechanism.

Round two: the mechanism itself was wrong. The rulebook describes a step the original model never encoded: before the Attempt chart ever gets rolled, the defensive manager has to decide to hold or not hold a runner, then the offensive manager has to actively decide to send the runner and announce that intent to the defensive manager, and lastly the defensive manager has to decide to throw or concede the base. Anyone who has played the game known that this is the playing sequence but programming it in such a way that model's the expected manager's decision at each turn was the challenge. A steal attempt isn't automatic – it's gated behind a real strategic decision, made on the runner's own perceived success rate, and the Attempt chart itself (Hold grade times Lead rating, the mechanic that determines whether a "good jump" happens once someone's already going) only fires after that decision is made. The original model had every runner-on-base situation trigger an automatic Attempt-chart roll, which is equivalent to assuming every manager sends every runner every time, regardless of the odds. This was an obvious modeling flaw. Real managers don't do that, and neither, it turns out, should a value model – this was the actual source of Hunter Brown's problem, since his poor Hold *letter grade* (which only governs whether a jump succeeds *once someone's already going*) was driving the entire attempt frequency, when the number that should gate whether an attempt happens at all – his Hold *plus/minus*, which enters the runner's own success-rate calculation – was close to league average the whole time.

Round three: the deterrence paradox. Modeling the send-decision as a logistic function of the runner's success probability, centered on the base's breakeven rate, fixed the conceptual gap but broke something else. Naively comparing "value to the offense with this pitcher" against "value to the offense with a league-average pitcher" produces a mathematical artifact whenever the reference success rate sits below breakeven, which it does at league average: a great pitcher suppresses attempts almost to zero, and an offense that isn't attempting has an expected value near zero, which reads as *higher* (better for the offense) than the small-but-nonzero negative expected value at league average, where attempts still occasionally happen. The model was reading successful deterrence as if it helped the offense. The fix was to stop comparing raw expected values and instead decompose the effect into two independently signed terms – an attempt-suppression term weighted by the reference value's *magnitude*, not its sign, and an execution-suppression term weighted by the reference frequency – verified monotonic by direct sweep across the entire real range of Hold grade, Hold plus/minus, and catcher Arm ratings in the league, not just spot-checked on a couple of convenient players.

The last piece: whose runners are we suppressing? A great arm or a great Hold rating doesn't have one fixed value – its value depends entirely on who's actually on base. A catcher with a cannon arm barely matters against a lineup of non-threats; the same arm meaningfully suppresses a lineup full of burners. The final version of the model integrates every defense-side calculation across the real, plate-appearance-weighted distribution of runner types in the league – not one reference runner, the actual mix, where 25.9% of all plate appearances belong to the least-threatening runner grade and a real minority sit near or above the breakeven threshold where arm quality actually bites. That distributional integration is also what makes the monotonicity hold cleanly: a weighted

sum of individually monotonic functions is itself monotonic, so getting each runner-type calculation right in isolation was enough to get the aggregate right too.

Hunter Brown's steal-of-second-and-third Run Game Control contribution settled at a rounding error relative to a Cy Young case – which is what "people don't run all over him" should look like in the numbers.

Fielding matters – when it matters – which is less than you think

Field Score prices error rate, range, and outfield arm the same context-neutral way as everything else — real chart-derived category mixes (slow rollers, smashes up the middle, drilled liners, and so on for the infield; flares, gap shots, drives over the head for the outfield), routed through actual Range Result tables rather than assumed distributions. Catchers get a Handling component (their effect on the umpire's ball/strike-adjacent resolution) plus an arm-on-base stealing component built on the identical send-gate-and-distribution framework as Run Game Control.

How often does it even come up? Not as often as intuition suggests. An infield range check fires on 4.87% of plate appearances league-wide; an outfield range check, 1.94%. Error opportunities are rarer still – the error band occupies a fixed slice of the roll space, split so that every position except pitcher gets an equal 10% share of error-check chances, with the pitcher himself getting a double share (20%) since he's involved, in some fielding capacity, on more plays than anyone. That even split is worth pausing on: it means the *frequency* of an error opportunity doesn't differentiate shortstop from right field at all. Whatever gap exists between positions in how much errors matter comes entirely from *what happens once the check fires* – the actual error-rate table – not from the shortstop getting more chances to fail.

And that table has a real, validating shape. Error rate by position, at both the best (rating 100) and worst (rating 5) end of the scale:

Position	Error rate at best (100)	Error rate at worst (5)
Shortstop	25%	100%
Third Base	25%	100%
Second Base	13%	89%
First Base	5%	79%
Left / Center / Right Field	0%	38%
Catcher	0%	18%

Shortstop and third base are the only positions where even a rating-100 fielder still boots a real chunk of his tough chances – a quarter of them, in fact – while an elite outfielder essentially never makes an error and an elite catcher almost never does either. That's not a modeling choice; it falls directly out of the chart data, and it happens to match the sport's own folk wisdom that short and third are where the hardest, highest-variance plays live. It also supports a fact of the Dynasty game structure in which most of the outs a shortstop will convert are never tested for range or error as

they are simply baked into the game. If the game's construct required every ball hit to the shortstop to be checked for range and error, game play would be long and Dynasty would be the MLB tabletop game equivalent of Pro Football Phantasm – an abomination of a game that took half a day to play.

So how much does a great shortstop's glove actually buy you? Isolating range alone – holding error rate and everything else at league average – an A+ range shortstop is worth about **+2.88 runs per 650 plate appearances of playing time**, relative to a league-average (grade C) shortstop; an F-range shortstop costs about **–2.10**. That's a swing of about **5.0 runs** across the full grade spectrum from the range component alone. A right fielder's range swings a smaller **2.6 runs** (A+ to F) – consistent with a less demanding defensive spot – but arm, isolated from best to worst mark on the card, swings about **5.8 runs**, larger than either range or error on its own. One honest simplification is worth flagging: the model doesn't differentiate arm value by which outfield spot a player plays – it computes a single league-wide arm value and splits it evenly across left, center, and right, when in real baseball a right fielder's arm gets tested more often. The swing above is best read as the value of arm quality at a generic outfield spot, probably understating right field and overstating left.

Put next to the offensive numbers from earlier in this paper, that's a useful sense of scale: a single home run is worth 1.46 runs as one event, and an elite defensive shortstop's *entire season* of range advantage over a replacement-level one is worth about three of those. Fielding is real, position-dependent, and – at premium spots and on the arm axis specifically – bigger than a lot of DLB managers might assume. It's also, appropriately, smaller than a full season of plate appearances, which is exactly the shape a value model should produce.

The best validation of the arm-on-base stealing piece specifically came almost for free: isolating just that component, J.T. Realmuto's modeled arm value came out ahead of Cal Raleigh's, both positive, with a below-average defender showing correctly negative. Nobody built that ordering in by hand – it fell out of feeding real card ratings through a mechanism that had already been checked for correctness on its own terms, and it happens to match Realmuto's actual reputation as one of the best throwing catchers in the game.

The error-severity story: a chart, a rework, and a bug that had been there the whole time

Every error in this model was, for most of the build, priced identically: a reach-on-error was valued as if it were a single, no matter where on the field it happened. That was a real question worth asking rather than assuming – a ball that skips through an infielder's legs and a fly ball an outfielder completely loses in the sun are not the same play and treating them as interchangeable meant the model was almost certainly undervaluing outfield errors and overvaluing the precision of infield ones.

Getting the real chart. The DLB rulebook's own description of the error mechanic – roll to determine which position has the chance, roll again against the fielder's Error rating – gives no hint of a severity breakdown. The physical ERROR chart tells a different story: every outcome row is labeled with exactly how many bases it grants, from "1 base error" up through explicit 2-base and 3-

base outcomes, and the roll-range width behind each label varies by position and by rating. Transcribing that chart by eye from a photograph the first time through surfaced a real, structural inconsistency – at the best rating, several pitcher error rows appeared to overlap the catch-all “no error” row, which shouldn't be possible. Rather than guess which specific cell was misread and silently patch it, that inconsistency got flagged and corrected offline manually for a second pass rather than corrected on faith. A full, careful 20-column rework resolved it cleanly: those rows are genuinely blank at the elite rating – an excellent fielder simply never produces that specific error type – and the earlier transcription had every value shifted one column over. This underscores the different between structured and unstructured data. Most charts were fed into the modeling process as photographs (i.e., unstructured data that the computer attempted to convert to structured data), and many needed to be manually corroborated to keep the project moving forward with statistical precision. Player card data was fed into the process as structured data in the form of a fully corroborated Excel spreadsheet covering every rating on every card available in the DLB set for the 2025 season. This is key: assuming the charts and rulebook don’t change from season to season, player data for future seasons will be able to easily be fed into the model to produce new league-wide player baselines, determined the new season’s context-neutral values, and output player values across the rubric of metrics the model has been designed to produce.

What the corrected error chart actually shows. Shortstop, third base, second base, and pitcher all carry a severity mix that stays close to stable across the entire rating range – a bad shortstop makes far more errors than a good one, but the *kind* of error barely changes, staying around three-quarters single-equivalent at every rating. First base shows a real, if modest, shift, including genuine 3-base exposure at poor ratings that doesn't exist at all for an elite one. Outfield is the outlier, and it's a large one: at the best rating, outfield errors are rare enough to barely register; by the time you're down to a poor rating, they're consistently 45–52% triple-equivalent, not the mild single-equivalent bobble the old flat assumption assigned everywhere. Run through the model's actual linear weights, that reshapes the numbers from earlier in this paper: shortstop's error-component swing widens modestly from the old flat estimate to **1.22 runs** across the full rating range, while right field's more than doubles, to **1.09 runs** – smaller in absolute terms than shortstop's due to reduced frequency but closing the gap due to materially higher incidence severity. Remember, the goal of the value model for fielders is not to ascertain who commits more errors; it is to determine who adds the most value to the team’s ability to suppress runs.

A second, unrelated bug found along the way. Wiring the corrected severity data into the model required opening the function that prices every error, and doing that surfaced something that had nothing to do with severity at all: the function was being called with lowercase position keys for every hitter (‘ss’, ‘rf’, and so on) but the underlying error-rate table only has uppercase keys. That mismatch meant the lookup silently failed and returned zero for every single hitter, every time, in every version of this model built before Version 23 – the error component of Field Score had never actually applied to a hitter's evaluation at all, only to a pitcher's own fielding, where a separate part of the code happened to hardcode the case correctly. Fixing a one-character case mismatch changed Field Score for every hitter in the pool, some by a little, a few meaningfully. J.T. Realmuto's Field Score, for instance, moves from slightly negative to slightly positive once his own error rate is actually being counted.

Validating both together. With the real severity data and the case fix both in place, additivity holds with zero mismatches across all 1,143 players and Run Game Control's range is completely unchanged, confirming neither fix leaked into pitcher-side mechanics that shouldn't have moved. The clearest validation came from Field Score's new extremes: the top three qualified hitters by Field Score are Pete Crow-Armstrong, Ceddanne Rafaela, and Corbin Carroll – three widely recognized elite defensive outfielders – and the single worst Field Score in the entire 577-hitter pool belongs to Kyle Schwarber, who has one of the sport's most well-earned reputations as a defensive liability in left field. Nobody built that ordering in; it fell out of a corrected chart and a one-line bug fix.

Closing the catcher: Wild Pitch, Passed Ball, and the throwing error

Two of the three items the v23 draft flagged as open – passed balls and catcher errors on steal attempts – got closed out in Version 24, and both turned out to have the same shape: the chart data had already been transcribed into the codebase, correctly, well before either was ever turned into a run value. Neither was a missing chart. Both were a missing wire.

The Umpire band, which is 2.7% of all plate appearances resolved by its own d100 sub-roll, routes into four possible outcomes depending on the base state: park visibility, catcher Handling, umpire strike-zone framing, and, with runners on base, a Control chart crossing the pitcher's own Wild Pitch rating against the catcher's own Passed Ball rating. That chart was sitting fully built in the codebase, validated cell-by-cell against a photographed copy of the physical Control chart with zero discrepancies, but nothing downstream ever converted its probabilities into runs. Wild Pitch and Passed Ball, as far as the model was concerned, didn't exist until v24.

They resolve identically on the field – a Control chart hit, whether charged to the pitcher or the catcher, means every baserunner advances one base and the batter stays at the plate – so the model prices them as one event: a battery's combined probability of either outcome drives a single run-expectancy delta on the base-out state, using the identical state-visitation-weighted machinery already built for steal mechanics. A league-average battery (both sides graded A, which is genuinely the modal grade at both positions across this league's card pool) sits at zero by construction; the full swing from an A-graded to an F-graded Wild Pitch rating, holding the catcher at league average, costs a pitcher about 0.04 runs per nine innings – small, appropriately, since the Control chart is only reached on a fraction of an already-small 2.7% band. The equivalent swing on the catcher's side, from an A-graded to an F-graded Passed Ball rating, costs about 0.58 runs across a 650-plate-appearance season.

A companion block in the same Umpire band – bases-empty rolls 25 through 32, labeled “catcher Passed Ball rating / Temperature” on the original chart – turned out not to be worth pricing at all. Modeling in-game temperature was ruled out as more trouble than the payoff justified, and the practical resolution settled on was a flat three-way split – a single to center, a return to the batter's normal card sequence, or a soft groundout to the pitcher – applied identically regardless of catcher grade. Because it doesn't vary by rating, it can't move any player's score under a model built entirely on deltas from league average – an elite and a replacement-level catcher get the exact same addition to the league's run environment from this block, so leaving it unwired costs nothing.

The catcher error-on-steal mechanic followed the same pattern. Row six of the Steal chart, labeled "SB + Error/Catcher?," was already transcribed and column-validated in the baserunning module, so the model already knew this row exists and already folded it into the overall probability that a steal attempt succeeds. What it never did was distinguish a clean stolen base from one that comes with a bonus: landing in that row still means a successful steal either way, but triggers a second, independent roll against the catcher's own existing fielding-error rating (the same ERR-1 rating already pricing his errors at his primary position, reused here exactly the way Arm rating is already reused across position fielding and basestealing prevention) – clear that roll and every runner on base, including the one who just stole, picks up one more base.

Wiring that in stayed deliberately separate from the module that already prices steal attempts. That module carries an explicit history of a long, three-round debugging chase – a –18.50 Run Game Control anomaly (Hunter Brown, the outlier) traced first to a bad reference baseline, then to a missing send-decision gate, then to a mathematical sign-instability bug in how deterrence gets valued – and its own documentation is blunt about how much validated care sits behind it. Rather than open that logic back up for what amounted to one additional roll-and-branch, the error-on-steal component was built as a fully separate, additive layer: its own run-expectancy delta for "steal plus bonus base," integrated over the same real, plate-appearance-weighted runner distribution the rest of the module already uses, added into Field Score after the fact rather than woven into the validated core. The full swing, from a rating-100 to a rating-5 catcher, comes to about 0.35 runs across a 650-plate-appearance season – smaller than Passed Ball, appropriately, since it depends on first landing in an already-narrow chart row before the error roll even happens.

The smell test held here too. The best-rated arm-and-error catchers in the pool – Salvador Pérez, Carson Kelly, Gabriel Moreno – all sit above league average on both new components; the single worst-rated catcher in the error dimension is Gary Sánchez, whose real-world reputation for throwing and blocking trouble is about as well established as any defensive knock in the sport. Nobody built that name into the model on purpose.

Diagnostic context: wRC+ and xFIP

Not every number in this model feeds Total Expected Contribution, and two new ones added this version are deliberately built to sit outside it: wRC+ for hitters and xFIP for pitchers, both familiar names from FanGraphs and both rebuilt from scratch against this league's own chart and player data rather than imported based on actual MLB performance. After all, the model cares only about a player's card value in the context of the game, not what their measured value it in MLB, though the two should closely align.

wRC+ came together cleanly, because the hard part was already done. Hit Score is already a runs-above-average figure per plate appearance, computed against weights that are, by the RE24 solve's own construction, exactly zero at league average – so converting it into an index number where 100 means league-average required only anchoring it to this league's own runs-per-plate-appearance rate, not a real-MLB constant, and rescaling. Bobby Witt Jr.'s wRC+ against right-handed pitching came out to 129, a clean match for a player already known, from Run Score alone, as one of the most dangerous all-around hitters in the pool.

xFIP took a wrong turn first, and the wrong turn is worth documenting because it's a direct echo of the deep-drive and steal-mechanics stories elsewhere in this paper: a rate built for one context doesn't automatically transfer to another. The first pass reused the classic FIP formula's coefficients – 13 for a home run, 3 for a walk or hit-by-pitch, –2 for a strikeout, plus a constant anchoring the whole thing to league ERA – the same numbers FanGraphs and every other sabermetric site uses. The result came out two to three times higher than it should have. Those coefficients aren't universal; they're values empirically fit to real MLB's own linear weights, and this model has its own, independently solved from its own chart data, with no guarantee the ratios match. xFIP was rebuilt from this league's own weights instead – the same Home Run, Walk, Hit-By-Pitch, and Strikeout values already anchoring every other run-value figure in this paper – recalibrated so that a league-average pitcher's xFIP lands exactly on the league's own runs-per-nine-innings figure, the same anchoring principle real FIP uses, just pointed at the right target. The result sits where it should: a league median of 4.26 runs, matching the model's own 4.23-to-4.36 run environment from earlier in this paper, with Tarik Skubal coming in at 4.09 – modestly better than average, not the outlier his real-world reputation would suggest, which is itself a useful, honest finding: the three true outcomes explain a smaller share of this model's total run prevention than they do in actual MLB, a property of this league's own linear weights rather than a flaw in the formula.

One structural note worth keeping, because it explains why xFIP and ordinary FIP are the same number in this model and always will be, absent a future change to how the model handles power: whether a deep drive becomes a home run is governed entirely by the batter's own Power rating on the Distance Hit chart – pitchers carry no equivalent rating anywhere in the source data, in the physical card game, or in this model – and every pitcher in this pipeline is evaluated against a context-neutral, league-average-power opponent by design. A pitcher's own home-run rate is therefore already exactly his own deep-drive frequency times a fixed league conversion rate; there's no lucky or unlucky home-run component left for xFIP to normalize away. An important note: the ability of a low-homer-ratio pitcher to take away a homerun determined from the hitter’s card Homerun band is not factored into this model.

Does it pass the smell test?

A model like this lives or dies on whether the players it says are good match the players who are actually good, and the validation here leaned hard on real-world reputation as a check the math alone can't provide. The top of the Run Score leaderboard: Trea Turner, Bobby Witt Jr., José Ramírez, Julio Rodríguez, Elly De La Cruz – five names that would show up on almost anyone's list of the game's most dangerous baserunners, arrived at without ever telling the model who they were. Shohei Ohtani's Run Score contribution is solidly positive, consistent with a player who's had a real 50-homer, 50-steal season. Chandler Simpson, the model's single highest baserunning value, is a legitimately elite modern burner. On the defensive side, the same discipline held: Pete Crow-Armstrong and Corbin Carroll at the top of Field Score, Kyle Schwarber at the bottom, none of it hand-tuned – all model-derived.

Beyond spot-checking famous names, the later stages of the build added a systematic layer: additivity verified across the full player pool (all 1,143 hitters and pitchers, not a sample), not a single missing value anywhere, and every rating-dependent function checked for monotonicity

across its entire realistic range rather than a couple of convenient test points. That discipline caught a genuinely separate bug hiding in plain sight during a quality-control pass: pickoff risk had been coded as a flat, unconditional frequency on every opportunity, when the rulebook places it as a sub-branch of the same Attempt-chart roll the steal-attempt gate already covers – it only exists once a steal attempt is already being considered. Fixing it meaningfully tightened Run Game Control's range across the league.

A separate, smaller find came from checking whether Run Score was double-counting a hitter's speed inside both Run Score itself and the Hit Score weights that feed Expected Offensive Contribution. It wasn't. Hit Score uses one fixed baserunning assumption for every hitter regardless of his own speed, so there was nothing for Run Score to duplicate, but the check surfaced a real, if minor, internal inconsistency that turned out to be worth chasing: the baseline extra-base rate baked into Hit Score's weights was a stale snapshot of an earlier, since-superseded calculation. Fixing that small gap is what led directly to discovering that the entire extra-base mechanic had the same “always attempt” flaw the steal mechanics needed fixed earlier – a reminder that the smaller consistency checks are often what surface the bigger structural ones.

Under the hood: the modeling platform and the math behind it

This section is for anyone curious about how the model is actually built, not just what it says – skippable if you're only here for the baseball.

The platform. Everything in this paper is built in Python, using NumPy for the linear algebra that solves the run-expectancy system. There's no borrowed MLB data anywhere in the pipeline and no black-box statistics package doing the heavy lifting. Every chart (error rates, range results, deep drive tables, baserunner advancement) is ingested directly from DLB's own rulebook and card and chart data, and every number downstream of that is derived, not assumed.

A few mathematical ideas do most of the work, and they're worth naming since they recur throughout this paper:

Markov chains and run expectancy. The 24-state RE24 table described earlier is a Markov chain – a system where the next state depends only on the current state, not on how you got there. Solving it means finding the long-run expected value of every state simultaneously, which reduces to a linear algebra problem (the kind of matrix solve NumPy handles natively). This is the same structure Lindsey worked out by hand from 6,000-plus half-innings of charted games in 1963; the model here just solves it exactly, from the chart's own probabilities, instead of estimating it from a sample.

Linear weights. Once every state has a value, converting that into “how many runs is a double worth” is just arithmetic: the change in expected runs the event causes, averaged across every situation it can occur in. Palmer's Linear Weights, described in the history section above, is the direct ancestor of every weight in the table in “The engine” section.

Logistic (sigmoid) functions. Several mechanics in this model – whether a runner attempts a steal, whether he tries to stretch a single into a double – aren't simple yes/no thresholds. A player

whose odds sit right at the breakeven point should be genuinely uncertain to attempt; one whose odds are far above or below it should be almost certain either way. A logistic curve, the same S-shaped function used in logistic regression, produces exactly that behavior: it smoothly transitions from “essentially never” to “essentially always” as a player's underlying success rate crosses the breakeven, rather than flipping on and off at an arbitrary line.

Weighted distributions instead of single reference points. The most consequential fix in this build – the discovery that a catcher's arm value depends on the *actual mix* of runners he faces, not one “average” runner – is really just the standard definition of expected value: sum every possible outcome times its probability of happening, rather than plugging in one typical case and hoping it's representative. The same idea underlies the error-severity mix described above: a position's error cost isn't one number, it's a probability-weighted blend across every way an error can actually happen at that spot.

Could this run in R or SQL?

R, yes, comfortably – arguably more naturally than Python in some respects, since R's *solve()* function handles exactly the kind of linear-system solve this model's RE24 engine needs, built in, no additional library required. R's data-frame-centric design maps directly onto the player-by-player computations throughout this build, and a meaningful share of the existing sabermetric community already works in R (the *baseballr* package and various Retrosheet-parsing tools are R-native). Excel output would move from *openpyxl* to *openxlsx* or *writexl*; matrix work would move from NumPy to base R or the *Matrix* package. Nothing in this model's actual logic would need to change, just its syntax.

SQL is a different, more honest answer: no, not for the engine itself, and it's worth being specific about why rather than hedging. SQL is a query language built for retrieving and aggregating relational data and it has no native way to solve a linear system, which is the entire mathematical core of the RE24 solve, and the nested conditional logic that resolves a card's roll ranges into specific outcomes doesn't map cleanly onto SQL's declarative, set-based query model. You could technically force some of it through recursive queries and procedural extensions, but you'd be fighting the language the whole way, reimplementing a numerical solver in a tool that was never meant to be one. Where SQL *would* fit naturally, and this is a real, useful role: as the storage and query layer underneath the model, once the numbers exist. The chart data itself, and the final computed value for every player, could live in relational tables – easy to filter, sort, and join against a league's roster data – while the actual computation that produces those numbers happens somewhere built for computation. That's a common, sensible split in real analytics work, and it would work well here too.

What this model doesn't do

Some omissions are permanent design choices, not gaps waiting to be filled:

- **No in-game situational overlays.** Clutch, Jam, hit-and-run, infielder-in, and similar context-dependent modifiers are deliberately excluded – Hit Score and Pitch Score are context-neutral by design, and folding in situational effects would defeat the purpose.

- **Primary position only.** Field Score evaluates a player at whichever position he's played the most innings, with no blending for true multi-position utility.
- **No Overworked-zone pitching.** Expected innings are capped at 1.5x a pitcher's card IP (the rulebook's safe ceiling), so the model never has to guess what happens past it, on the theory that no rational manager pushes a pitcher into that territory on purpose.

Others are real, current gaps:

- **Field Condition, Lighting, and Wind sub-charts** aren't modeled. The roll ranges that route into them are accounted for, but their own internal outcome breakdowns are not yet priced.
- **Double-play pivot, bunt-for-hit, and outfield home-run robberies** are real mechanics on the cards that haven't been modeled – flagged, not forgotten.
- **Outfield arm is not position-specific** – see the Fielding section above.
- **Injuries** aren't modeled. And they won't be in future versions. There is simply no way to credibly evaluate whether or not a player's injury rating will be called on during a DLB season.

Where this leaves things

The Version 24 model prices five skills, adds them into one number, and carries two further diagnostic-only stats alongside it, and every piece of all of it – the run-expectancy solve, the power curve, the park, the baserunning mechanics, the fielding components, the Umpire Control chart, the diagnostic stats – traces back to real chart data rather than an assumed shortcut. That wasn't true at every point in the build; the flat 40% home-run rate, the automatic-attempt steal mechanic, the automatic-attempt extra-base mechanic, the flat single-equivalent error value, and an unpriced Control chart and an unpriced error-on-steal branch were all shortcuts or outright gaps that got caught and closed over the course of a build that kept finding new ways to check its own work rather than declaring victory early.

The pattern connecting the steal-mechanics story, the extra-base story, and the error-severity story is worth stating as a general lesson, independent of this specific league: when the same underlying number drives both "how often does this happen" and "how does it turn out," treating either as uniform when it isn't will eventually produce a wrong answer that looks locally reasonable.

It happened three times over the course of this build, in three different parts of the model, at three different points in the schedule. The third time came bundled with an unrelated, purely mechanical bug (a case-sensitivity mismatch) that had nothing to do with severity at all and would never have surfaced without opening the same function to fix it. That's not a baseball-specific insight, but baseball, with its charts and its dice and its rulebook clauses about who's allowed to decide when a runner goes, happened to be where it showed up.

How much does the manager matter?

Reading this paper for many DLB managers might be daunting and even demoralizing. If the player ratings suggest my team is average, perhaps I should put the team on Computer Manager, save myself the misery, and hope my talent and probabilities improve next year, right?

Wrong. The model assesses player value based on probabilistic outcomes across normalized usage patterns. The total contribution calculations are ratioed based on actual MLB player and pitcher usage. If a player card has an at-bat limit, then their usage is capped to that limit, but a really good player with 250 MLB plate appearances and no at-bat limits could start and play a full season in DLB, thus playing more than the usage factored into their expected contribution. A good player with a bad injury rating could be hidden in the DH slot, dramatically reducing his injury likelihood. A player who is really good against one side of the platoon may never be used against their "bad" side by their DLB manager, thus producing a higher contribution than the model expects.

Then there are situation usage patterns and strategic ones, too. Bad outfielders might be used when backing pitchers who have narrow Outfield Range bands – the same could be true for bad infielders matching up with pitchers who have small Infield Range bands. A +3 Arm catcher could be paired with a high Hold-rating pitcher, negating the likelihood that the bad arm rating ever comes into play; or a bad run control battery could be matched against a team of slugs. A .227 hitter who has high baserunning and good bunt ratings could be asked to bunt for a hit every time – with no one on base, a 10 baserunner with an A bunt rating is safe 30% of the time. If a team is fleet of foot and solid with bunting and hit-and-run ratings, they will be more adept at moving runners and avoiding the most negative play of all – the double play. In order to score runs, you have to aggregate bases; and while mashing is the best and fastest way to aggregate bases in bulk, speed and peripheral skills can reduce highly negative, rally-killing outcomes.

Pitcher deficiencies are bit harder to hide, but every so often the pitcher who has uneven splits may have an opportunity to face a team that is simply unable to exploit their weakness; and since DLB doesn't use rotations, per se, managers can plan ahead to spot that one starter against the one team they're best able to throttle. The Opener strategy, used in the MMDBL chronically by the Phillies – including masterfully in their World Series championship season – can help a team cover a number of deficiencies by optimizing matchups to flip an opponent's lineups, deplete their late-game options, and maximizing run suppression. When paired with a speed and defense profile, this can be an effective way for a team seemingly bereft of top aces and hitting stars to compete – and we see it happen every year in the MMDBL.

Then there' the false positives that arise in a 20-team league where many players below baseline are never rostered. For example, an A-power guy is rendered far less effective if an opposing team is able to roll through a bullpen of arms with no Deep Drive results. Conversely, if a closer has a ton of Deep Drives but is facing 3 guys with C or worse power, he plays up. However, if the offensive manager has 3 A or B power guys waiting on the bench for just this late-inning opportunity, the closer plays down. In leagues with only 20-24 teams, managers need to construct their team with the expectation that a pitcher's deficiencies will more likely than not be exploited. This is less so for hitters, however, the Opener strategy makes is more likely if the lineup flip-flop leaves the wrong guy for a late-inning at-bat.

So what percentage of a team's ability to outperform their modeled aggregate expected contribution value falls squarely on how a GM constructs their team and the manager manages it? This is not a modeled outcome but we're estimating a swing of +/- 25% in wins beyond the game-based, model-driven probabilities. The range also accounts for good and bad luck, such as a team unperforming or overperforming on roll outcomes; or a team suffering a number of key injuries versus one that slides through with minimal injury impact despite having a number of players with bad injury ratings. Luck – or bad luck – is a real factor that should even out over a large sample size but sometimes doesn't. The best approach is to build your team the way you want to play it, maximizing the positive outcome probabilities within that context, and then have fun playing the season.

Appendix: Baseline reference table

Every "zero" in this model is a real, computed number, not an assumption – this table is the full list of what "league average" actually means in each dimension the model prices, current as of the fixes described above.

A player at exactly zero Total Expected Contribution performs, in every dimension this model prices, exactly at the league-average rates below – not replacement level in the wins above replacement (WAR) context, not a bad player, precisely average. Real-world replacement level in MLB sits roughly 15-to-20 runs below this zero point per 650 PA; this model does not yet define a replacement baseline of its own.

Run environment

Item	Value	Note
Runs/9 innings, vs. left-handed pitching	4.36	RE24-solved (see main text)
Runs/9 innings, vs. right-handed pitching	4.23	
Mean event value	0.000 (by construction)	Every linear weight is expressed relative to this

Power / Deep Drive

Item	Value
P(Deep Drive → Home Run), Grade A	39.0%
P(Deep Drive → Home Run), Grade B	22.3%
P(Deep Drive → Home Run), Grade C	10.6%
P(Deep Drive → Home Run), Grade D	3.7%
P(Deep Drive → Home Run), Grade F	0.4%

Park

The current benchmark is the **Dynasty League Average Ballpark** — wall distances averaged, unweighted, across all 30 real MLB parks, replacing Comerica Park as the universal benchmark used in every Deep Drive resolution.

Sector	Distance
Left Field Line	346 ft
Left Field	374 ft
Left-Center	397 ft
Center Field	416 ft
Right-Center	397 ft

Sector	Distance
Right Field	373 ft
Right Field Line	344 ft

Fielding

Item	Value
League Range grade by position	Mostly grade C (Catcher and First Base average grade D)
League Error rating, Shortstop	64.2 (of 100)
League Arm rating, Right Field	−0.29 (of a −4-to-+4 scale, negative is better)
League Catcher Handling grade	C
P(Infield Range Check fires)	4.87% of plate appearances
P(Outfield Range Check fires)	1.94% of plate appearances

Error severity mix by position

Fraction of that position's errors that are 1-base, 2-base, or 3-base equivalent, at the best (100) and worst (5) end of the error-rating scale, transcribed from the physical ERROR chart and based solely on a player's primary defensive position:

Position	1B/2B/3B at rating 100	1B/2B/3B at rating 5
Pitcher	0% / 0% / 0% (doesn't fire)	69% / 31% / 0%
Catcher	0% / 0% / 0% (doesn't fire)	56% / 44% / 0%
First Base	100% / 0% / 0%	71% / 26% / 3%
Second Base	70% / 30% / 0%	66% / 34% / 0%
Shortstop	76% / 24% / 0%	78% / 22% / 0%
Third Base	81% / 19% / 0%	83% / 17% / 0%
Outfield (LF/CF/RF)	0% / 0% / 0% (doesn't fire)	26% / 23% / 52%

Baserunning ratings

Item	Value
League Baserunning ("Run") rating	6.19
League Lead-2nd / Steal-2nd rating	1.87 / 3.55
League Lead-3rd / Steal-3rd rating	1.31 / 2.02
League pitcher Hold grade (most common)	B+
League pitcher Hold plus/minus	+0.88
League pitcher Pickoff rating	2.57 (of 10)
League catcher Arm rating	−0.17

Steal mechanics

A steal attempt requires two independent things to both happen: the offensive manager deciding to send the runner (based on the runner's own perceived success rate against the catcher and pitcher on the field), and — separately — the runner getting a "good jump" once he goes, governed by the pitcher's Hold *letter grade* and the runner's Lead rating. These are gated by different numbers on the card, confirmed directly from the rulebook.

Item	Value
Steal-of-second breakeven success rate	74.6%
Steal-of-third breakeven success rate	84.2%
Send-gate steepness	Logistic curve, centered on breakeven
Share of PA at the lowest Steal-2nd rating (non-threats)	25.9%

Extra-base mechanics

A separate decision with its own two breakevens — extending to a base without scoring is a much harder sell than attempting to score, because an immediate run is worth far more than an incremental base.

Item	Value
Breakeven, extending to an extra base (no run scores)	89.8%
Breakeven, attempting to score	58.5%
League-average success rate (both breakevens compared against this)	63.5%

Usage and workload

Item	Value
Hitter reference season	650 plate appearances
Pitcher usage ceiling	1.5x card innings pitched
Team plate-appearance normalization target	6,200
Team innings-pitched normalization target	1,458

Umpire Control chart (Wild Pitch, Passed Ball, catcher error-on-steal)

Item	Value
League Wild Pitch grade (pitcher, modal)	A
League Passed Ball grade (catcher, modal)	A
League catcher ERR-1 rating (PA-weighted mean)	73.4 (of 100)
RGC swing, Wild Pitch A to F (per 9 IP, catcher held at league average)	−0.04 runs

Item	Value
Field Score swing, Passed Ball A to F (per 650 PA, pitcher held at league average)	−0.58 runs
Field Score swing, error-on-steal rating 100 to 5 (per 650 PA)	−0.35 runs

Diagnostic stats: wRC+ and xFIP

Not summed into Total Expected Contribution — context only.

Item	Value	Note
wRC+ index point	100	League average, by construction
xFIP anchor	League runs/9 (4.23–4.36)	Same anchoring principle as real FIP, this league's own weights
Example: Bobby Witt Jr. wRC+ (vs RHP)	129	
Example: league median xFIP	4.26	
Example: Tarik Skubal xFIP (overall)	4.09	

Appendix: Top 20 Qualified Hitters

Pulled directly from the current model output (dlb_results_v7.xlsx), qualified at 300+ projected plate appearances for a 162-game MLB season.

Ranked by Expected Offensive Contribution

Rk	Player	MLB	MMDBL	PA	Off. Exp. Contribution
1	Aaron Judge	NYY	BAL	665	+74.50
2	Shohei Ohtani	LAD	DET	720	+64.47
3	Juan Soto	NYM	MIN	704	+45.57
4	Kyle Schwarber	PHI	MIL	712	+45.35
5	Cal Raleigh	SEA	ATL	693	+44.80
6	Nick Kurtz	OAK	ATL	483	+40.34
7	Geraldo Perdomo	ARZ	SFG	691	+32.06
8	George Springer	TOR	NYY	567	+31.66
9	Corbin Carroll	ARZ	STL	631	+27.66
10	Rafael Devers	SFG	CIN	719	+27.29
11	Byron Buxton	MIN	WAS	529	+26.39
12	Will Smith	LAD	MIN	426	+26.12
13	Pete Alonso	NYM	ATL	685	+26.01
14	Ronald Acuna Jr.	ATL	BAL	409	+25.34
15	Kyle Stowers	MIA	BOS	447	+24.75
16	Ketel Marte	ARZ	CIN	544	+24.20
17	Bobby Witt Jr.	KCR	KCR	672	+23.46
18	Matt Olson	ATL	CHC	715	+22.95
19	Eugenio Suarez	SEA	SEA	634	+22.75
20	Shea Langeliers	OAK	MIL	517	+22.53

Ranked by Hit Score vs Right-Handed Pitching

Rk	Player	MLB	MMDBL	PA	Hit Score vRHP
1	Nick Kurtz	OAK	ATL	483	+78.70
2	Shohei Ohtani	LAD	DET	720	+66.17
3	Aaron Judge	NYY	BAL	665	+64.87
4	Juan Soto	NYM	MIN	704	+53.01
5	Kyle Stowers	MIA	BOS	447	+50.81
6	Max Muncy	LAD	MIN	377	+44.72
7	Ronald Acuna Jr.	ATL	BAL	409	+44.15
8	Will Smith	LAD	MIN	426	+42.95

Rk	Player	MLB	MMDBL	PA	Hit Score vRHP
9	George Springer	TOR	NYY	567	+41.77
10	Kyle Schwarber	PHI	MIL	712	+37.43
11	Corey Seager	TEX	TEX	438	+36.60
12	Rafael Devers	SFG	CIN	719	+34.27
13	Cal Raleigh	SEA	ATL	693	+34.25
14	Pete Alonso	NYM	ATL	685	+34.09
15	Corbin Carroll	ARZ	STL	631	+32.71
16	Michael Busch	CHC	HOU	580	+32.29
17	Daylen Lile	WAS	STL	342	+31.73
18	Trent Grisham	NYY	NYY	576	+31.09
19	Jonathan Aranda	TBR	MIN	411	+29.53
20	Ketel Marte	ARZ	CIN	544	+28.44

Ranked by Hit Score vs Left-Handed Pitching

Rk	Player	MLB	MMDBL	PA	Hit Score vLHP
1	Aaron Judge	NYY	BAL	665	+93.57
2	Ivan Herrera	STL	NYY	431	+82.42
3	Shea Langeliers	OAK	MIL	517	+72.72
4	Byron Buxton	MIN	WAS	529	+65.95
5	Cal Raleigh	SEA	ATL	693	+62.29
6	Miguel Andujar	CIN	MIN	338	+56.95
7	Cody Bellinger	NYY	TEX	645	+53.98
8	Kyle Schwarber	PHI	MIL	712	+51.75
9	Jackson Chourio	MIL	KCR	579	+47.70
10	Ryan Jeffers	MIN	DET	456	+46.82
11	Dillon Dingler	DET	TEX	458	+46.30
12	Joey Bart	PIT	LAD	325	+45.68
13	Gleyber Torres	DET	WAS	617	+45.57
14	Geraldo Perdomo	ARZ	SFG	691	+45.27
15	Paul Goldschmidt	NYY	TEX	525	+44.72
16	Nico Hoerner	CHC	TOR	638	+44.40
17	Romy Gonzalez	BOS	MIN	333	+42.35
18	Maikel Garcia	KCR	SFG	657	+42.21
19	Austin Hays	CIN	BAL	409	+40.73
20	Jo Adell	LAA	STL	559	+38.74

Ranked by Field Score

Rk	Player	MLB	MMDBL	PA	Field Score
1	Pete Crow-Armstrong	CHC	ATL	620	+4.68
2	Ceddanne Rafaela	BOS	BOS	574	+4.17
3	Corbin Carroll	ARZ	STL	631	+4.08
4	Cody Bellinger	NYY	TEX	645	+3.83
5	Steven Kwan	CLE	PHI	680	+3.81
6	Matt Olson	ATL	CHC	715	+3.31
7	Bobby Witt Jr.	KCR	KCR	672	+3.31
8	Sal Frelick	MIL	SFG	575	+3.19
9	Ian Happ	CHC	PIT	656	+3.14
10	Fernando Tatis Jr.	SDP	ATL	683	+2.90
11	Wyatt Langford	TEX	ATL	563	+2.83
12	Trea Turner	PHI	NYY	632	+2.73
13	Nico Hoerner	CHC	TOR	638	+2.72
14	Masyn Winn	STL	HOU	525	+2.69
15	Maikel Garcia	KCR	SFG	657	+2.65
16	Ke'Bryan Hayes	CIN	STL	563	+2.59
17	Aaron Judge	NYY	BAL	665	+2.55
18	Spencer Steer	CIN	MIL	560	+2.45
19	Andy Pages	LAD	BAL	610	+2.44
20	Tyler Soderstrom	OAK	NYY	616	+2.40

Ranked by Total Expected Contribution

Rk	Player	MLB	MMDBL	PA	Total Exp. Contribution
1	Aaron Judge	NYY	BAL	665	+76.98
2	Shohei Ohtani	LAD	DET	720	+65.34
3	Cal Raleigh	SEA	ATL	693	+45.39
4	Juan Soto	NYM	MIN	704	+44.91
5	Kyle Schwarber	PHI	MIL	712	+40.33
6	Nick Kurtz	OAK	ATL	483	+39.68
7	Geraldo Perdomo	ARZ	SFG	691	+33.81
8	Corbin Carroll	ARZ	STL	631	+33.06
9	George Springer	TOR	NYY	567	+30.81
10	Bobby Witt Jr.	KCR	KCR	672	+28.34
11	Byron Buxton	MIN	WAS	529	+28.15
12	Rafael Devers	SFG	CIN	719	+27.15
13	Will Smith	LAD	MIN	426	+26.11

Rk	Player	MLB	MMDBL	PA	Total Exp. Contribution
14	Kyle Stowers	MIA	BOS	447	+26.10
15	Matt Olson	ATL	CHC	715	+25.43
16	Ronald Acuna Jr.	ATL	BAL	409	+25.35
17	Pete Alonso	NYM	ATL	685	+25.28
18	Ketel Marte	ARZ	CIN	544	+24.54
19	Francisco Lindor	NYM	WAS	709	+24.45
20	Shea Langeliers	OAK	MIL	517	+23.05

Appendix: Top 20 Qualified Pitchers

Same source and convention as the hitters above, qualified at 90+ projected innings pitched.

Ranked by Expected Stuff & Command Contribution

Rk	Player	MLB	MMDBL	xFIP (Overall)	Stuff & Cmd Contribution
1	Garrett Crochet	BOS	BOS	4.11	+50.78
2	Hunter Brown	HOU	DET	4.15	+48.46
3	Nick Pivetta	SDP	NYY	4.20	+47.69
4	Tarik Skubal	DET	TEX	4.09	+47.02
5	Paul Skenes	PIT	MIL	4.09	+46.76
6	Bryan Woo	SEA	CHC	4.22	+43.61
7	Nathan Eovaldi	TEX	TOR	4.13	+43.41
8	Yoshinobu Yamamoto	LAD	KCR	4.17	+41.76
9	Kevin Gausman	TOR	MIN	4.19	+40.57
10	Zack Wheeler	PHI	TOR	4.14	+38.91
11	Trevor Rogers	BAL	SFG	4.14	+35.88
12	Jacob deGrom	TEX	SEA	4.29	+35.02
13	Carlos Rodon	NYY	SFG	4.21	+34.81
14	Drew Rasmussen	TBR	MIN	4.25	+34.72
15	Cristopher Sanchez	PHI	TOR	4.11	+34.65
16	Ryne Nelson	ARZ	STL	4.26	+29.50
17	Aroldis Chapman	BOS	SEA	4.00	+28.14
18	Freddy Peralta	MIL	MIL	4.25	+26.96
19	Chris Sale	ATL	ATL	4.09	+25.22
20	Max Fried	NYY	PIT	4.21	+25.02

Ranked by Pitch Score vs Right-Handed Hitters

Rk	Player	MLB	MMDBL	xFIP vRHH	Pitch Score vRHH
1	Mason Miller	SDP	BOS	3.85	+3.05
2	Robert Suarez	SDP	WAS	4.06	+2.79
3	Shawn Armstrong	TEX	WAS	4.09	+2.57
4	Aroldis Chapman	BOS	SEA	4.01	+2.54
5	Matt Svanson	STL	SFG	4.07	+2.25
6	Yariel Rodriguez	TOR	PHI	4.11	+2.19
7	Nathan Eovaldi	TEX	TOR	4.06	+2.19
8	Eduard Bazardo	SEA	WAS	4.10	+2.09

Rk	Player	MLB	MMDBL	xFIP vRHH	Pitch Score vRHH
9	Clarke Schmidt	NYY	NYY	4.14	+2.00
10	Dennis Santana	PIT	LAD	4.15	+1.93
11	Trevor Rogers	BAL	SFG	4.12	+1.89
12	David Bednar	NYY	TEX	3.91	+1.87
13	Steven Okert	HOU	WAS	4.06	+1.87
14	Brad Keller	CHC	SEA	4.12	+1.82
15	Zack Wheeler	PHI	TOR	4.05	+1.78
16	Paul Skenes	PIT	MIL	4.05	+1.75
17	Emmet Sheehan	LAD	SEA	4.04	+1.74
18	Carlos Estevez	KCR	NYY	4.14	+1.71
19	Drew Rasmussen	TBR	MIN	4.19	+1.69
20	Tarik Skubal	DET	TEX	4.07	+1.65

Ranked by Pitch Score vs Left-Handed Hitters

Rk	Player	MLB	MMDBL	xFIP vLHH	Pitch Score vLHH
1	Aroldis Chapman	BOS	SEA	3.99	+3.16
2	Garrett Crochet	BOS	BOS	4.12	+2.95
3	Adrian Morejon	SDP	BAL	4.13	+2.63
4	Tanner Banks	PHI	PHI	4.17	+2.44
5	Steven Okert	HOU	WAS	4.16	+2.39
6	Dennis Santana	PIT	LAD	4.30	+2.25
7	Emilio Pagan	CIN	MIN	4.16	+2.20
8	Jacob Latz	TEX	PIT	4.33	+2.18
9	Chris Sale	ATL	ATL	4.16	+2.16
10	Tyler Mahle	TEX	MIN	4.23	+2.16
11	Brandon Woodruff	MIL	NYY	4.17	+2.15
12	Steven Matz	BOS	PHI	4.28	+2.14
13	Ryne Nelson	ARZ	STL	4.29	+2.13
14	Yoshinobu Yamamoto	LAD	KCR	4.17	+2.13
15	Carlos Vargas	SEA	STL	4.31	+2.13
16	Edwin Diaz	NYM	SFG	4.09	+2.12
17	Trevor Rogers	BAL	SFG	4.17	+2.12
18	Tyler Holton	DET	MIL	4.25	+2.10
19	Jhoan Duran	PHI	ATL	4.16	+2.06
20	Gabe Speier	SEA	SFG	4.10	+2.00

Ranked by Run Game Control

Rk	Player	MLB	MMDBL	Run Game Control
1	Shota Imanaga	CHC	NYY	+1.51
2	Tarik Skubal	DET	TEX	+1.35
3	Brad Lord	WAS	ATL	+1.34
4	Matthew Boyd	CHC	SFG	+1.31
5	Kris Bubic	KCR	TEX	+1.20
6	Valente Bellozo	MIA	ZXFA	+1.16
7	Ryne Nelson	ARZ	STL	+1.09
8	Aaron Civale	CHC	TOR	+1.05
9	Zack Wheeler	PHI	TOR	+1.05
10	Hoby Milner	TEX	PHI	+1.00
11	Bailey Ober	MIN	ZXFA	+1.00
12	Daniel Lynch IV	KCR	KCR	+0.96
13	Jared Koenig	MIL	NYY	+0.94
14	JP Sears	SDP	ZXFA	+0.92
15	Colin Rea	CHC	TEX	+0.90
16	Logan Webb	SFG	MIL	+0.90
17	Clayton Kershaw	LAD	PHI	+0.87
18	Ryan Walker	SFG	SFG	+0.87
19	Chad Patrick	MIL	ATL	+0.84
20	Sonny Gray	STL	BAL	+0.82

Ranked by Field Score

Rk	Player	MLB	MMDBL	Field Score
1	Jacob deGrom	TEX	SEA	+2.09
2	David Peterson	NYM	BAL	+2.04
3	Luis Severino	OAK	NYY	+1.97
4	Logan Webb	SFG	MIL	+1.79
5	Tomoyuki Sugano	BAL	MIL	+1.60
6	Dean Kremer	BAL	LAD	+1.54
7	Jose Soriano	LAA	PHI	+1.52
8	Seth Lugo	KCR	SFG	+1.48
9	Tanner Bibee	CLE	ATL	+1.41
10	Ryne Nelson	ARZ	STL	+1.38
11	Michael Wacha	KCR	SEA	+1.33
12	Brady Singer	CIN	BOS	+1.31
13	Ryan Pepiot	TBR	DET	+1.29
14	Gavin Williams	CLE	KCR	+1.29

Rk	Player	MLB	MMDBL	Field Score
15	Jose Berrios	TOR	PIT	+1.29
16	Bryce Elder	ATL	PHI	+1.21
17	Brad Lord	WAS	ATL	+1.17
18	Matthew Liberatore	STL	CIN	+1.17
19	Kevin Gausman	TOR	MIN	+1.17
20	Antonio Senzatela	COL	PIT	+1.17

Ranked by Total Expected Contribution

Rk	Player	MLB	MMDBL	Stuff & Cmd	Run Game Control	Field Score	Total Exp. Contribution
1	Garrett Crochet	BOS	BOS	+50.78	-1.01	+0.83	+50.61
2	Hunter Brown	HOU	DET	+48.46	+0.12	+1.12	+49.70
3	Tarik Skubal	DET	TEX	+47.02	+1.35	+0.33	+48.70
4	Nick Pivetta	SDP	NYY	+47.69	+0.10	+0.00	+47.79
5	Paul Skenes	PIT	MIL	+46.76	+0.36	-0.87	+46.25
6	Bryan Woo	SEA	CHC	+43.61	-0.21	+0.51	+43.91
7	Nathan Eovaldi	TEX	TOR	+43.41	-0.99	+0.71	+43.14
8	Yoshinobu Yamamoto	LAD	KCR	+41.76	-0.31	+0.95	+42.40
9	Kevin Gausman	TOR	MIN	+40.57	-1.31	+1.17	+40.42
10	Zack Wheeler	PHI	TOR	+38.91	+1.05	+0.44	+40.40
11	Jacob deGrom	TEX	SEA	+35.02	-0.47	+2.09	+36.64
12	Trevor Rogers	BAL	SFG	+35.88	-0.12	+0.44	+36.19
13	Drew Rasmussen	TBR	MIN	+34.72	-0.50	+0.91	+35.13
14	Cristopher Sanchez	PHI	TOR	+34.65	-0.93	+1.11	+34.83
15	Carlos Rodon	NYY	SFG	+34.81	-1.97	+0.79	+33.63
16	Ryne Nelson	ARZ	STL	+29.50	+1.09	+1.38	+31.97
17	Aroldis Chapman	BOS	SEA	+28.14	-1.60	+0.25	+26.78
18	Max Fried	NYY	PIT	+25.02	+0.10	+0.49	+25.61
19	Freddy Peralta	MIL	MIL	+26.96	-2.73	+1.07	+25.29
20	Chris Sale	ATL	ATL	+25.22	-0.28	+0.21	+25.15

Appendix: MLB Team Leaderboard

Every MLB team's roster, summed: Total Run Production Score is the sum of every hitter's Total Expected Contribution on that MLB team's roster; Total Run Prevention Score is the same sum for pitchers; Team Talent Score is the two added together. This is a real MLB team's aggregate player pool.

MLB Team	Total Run Production Score	Total Run Prevention Score	Team Talent Score
SEA	101.08	69.67	170.75
NYY	106.23	55.93	162.15
CHC	48.51	107.07	155.59
PHI	73.00	45.21	118.21
MIL	67.69	41.18	108.87
LAD	109.51	-4.23	105.28
SDP	4.02	99.21	103.23
TEX	-18.58	104.21	85.63
TBR	-26.92	92.66	65.75
NYM	79.42	-15.38	64.04
BOS	19.03	40.76	59.78
TOR	13.68	33.00	46.68
KCR	-22.39	57.45	35.06
DET	42.49	-12.11	30.38
HOU	-35.86	48.66	12.80
PIT	-54.97	63.88	8.91
CIN	-41.13	38.11	-3.02
ARZ	55.79	-85.03	-29.24
OAK	58.46	-89.32	-30.85
SFG	5.55	-38.30	-32.76
MIA	11.99	-67.56	-55.57
ATL	-1.72	-61.52	-63.24
CLE	-84.88	16.68	-68.20
MIN	-16.86	-76.42	-93.28
STL	-20.01	-101.54	-121.55
BAL	-79.90	-59.25	-139.16
CHW	-72.40	-79.24	-151.64
LAA	-68.68	-117.50	-186.18
WAS	-36.53	-193.93	-230.47
COL	-83.77	-287.71	-371.48