The Batter Discomfort Index

Measuring Pitch Quality Through Batter Swing Disruption

Ben Gratz | Washington University in St. Louis | B.A. Economics & Computer Science, 2027

2025 MLB Season | Full Regular Season + Postseason

Data: Baseball Savant Statcast via pybaseball

726,142 pitches | 309,337 swings | 315,381 outside-zone pitches

The Batter Discomfort Index

In today's landscape of pitching analytics, there have been many attempts to quantify how "good" a certain pitch is, such as Stuff+, pitch-level xwOBA, and more. However, where many of these metrics come up short is that they focus largely on pitch physics, ignoring the effect a given pitch has on the batter. Therefore, I constructed the Batter Discomfort Index (BDI), a per-pitch-type metric that measures how uncomfortable a certain pitch makes a batter by tracking two things: disruption to the batter's swing mechanics relative to their own baseline, and disruption to their swing decision behavior outside the zone.

To build the BDI, I pulled all pitches registered in the 2025 MLB season using the pybaseball Python library, covering the full regular season and postseason (726,142 total pitches). The dataset includes two distinct subsets used for different components of the metric: swings (309,337 pitches where the batter swung, which have bat-tracking data) and outside-zone pitches (315,381 pitches, used to measure chase behavior).

Batter Baselines

A key design decision in the BDI is that all deviation metrics are computed relative to each batter's own pitch-type-specific baseline — not a season-wide average and not a league average. This is because curveballs inherently produce slower bat speeds and different swing physics than fastballs, regardless of the pitcher throwing them. If I used a batter's overall season average as the baseline, every breaking ball would look artificially "disruptive" just because it's a breaking ball.

By computing each batter's average swing metrics separately against fastballs, sliders, curveballs, and so on, I isolate what's unusual about a specific pitcher's pitch relative to what the batter normally does against that pitch type. In other words, Batter A is likely to swing differently against a fastball than a curveball. But Batter B is also likely to swing differently against a curveball than Batter A does on the same pitch. A per-batter, per-pitch-type baseline controls for both of these sources of variation simultaneously.

I confirmed this design decision empirically: an early version of the metric used season-wide baselines, and the resulting leaderboard was dominated almost entirely by curveballs and breaking balls — not because those pitches were actually the most disruptive, but because they structurally produce slower bat speeds than fastballs. Switching to pitch-type-specific baselines produced a more varied, credible leaderboard. Minimum sample requirements of 30 swings and 20 outside-zone pitches against a given pitch type were required to form a reliable baseline, producing 2,922 batter/pitch-type baseline rows across 541 unique batters.

Swing Mechanics Deviation (Physics Score)

The first component of the BDI measures how much a pitch disrupted a batter's swing mechanics. For each swing, I computed the deviation from the batter's pitch-type-specific baseline across four candidate variables: bat speed, horizontal bat path direction (attack direction), vertical swing plane (swing path tilt), and attack angle. Bat speed captures changes in the force and timing of the swing, while attack direction and swing path tilt describe how the hitter's bat travels through the hitting zone in the horizontal and vertical planes, respectively. Attack angle measures the upward or downward trajectory of the bat at contact. Together, these variables provide a comprehensive description of a batter's swing mechanics, allowing BDI to quantify how much a pitch alters the swing a hitter would typically produce against that pitch type.

Each deviation was z-scored within pitch type so that the same standard deviation of disruption on a curveball is directly comparable to the same disruption on a fastball. This matters because the natural variance in bat speed differs across pitch types — a 3 mph drop on a fastball swing might be well within normal variation, while the same 3 mph drop on a curveball might be extreme.

I then ran an OLS regression of launch speed on these four deviation metrics, using contact swings only (n = 210,312). The choice to train on launch speed rather than a binary outcome like whiff is deliberate: launch speed is the most direct physical measurement of what swing disruption actually produces — worse mechanics should translate to weaker contact. The regression was restricted to contact swings because launch speed only exists when the bat makes contact. The resulting weights are then applied to all swings, including whiffs, to produce a physics score for every swing in the dataset.

Variable Selection

Several candidate variables were tested and either retained or dropped based on their regression coefficients:

- **Bat speed deviation** dominated, with a coefficient of −5.501 (by far the largest), meaning a one standard deviation increase in bat speed deviation below baseline predicted a 5.5 mph drop in launch speed. This makes intuitive sense — a batter who is late or off-balance produces a slower, weaker swing.
- **Attack direction** required a key methodological discovery. When computed as an absolute value (unsigned deviation), the coefficient was positive — more horizontal deviation predicted harder contact. This is counterintuitive but logical: batters who deviate from their normal horizontal bat path and still make contact are often making aggressive, committed swings. The signed version correctly identified that deviation in one specific direction (being pulled off their normal path by the pitch) predicts weaker contact. This was one of the most important findings of the development process.
- **Swing path tilt** was kept as an unsigned (absolute value) deviation. Making it signed simultaneously destabilized the attack direction coefficient due to collinearity between the two geometric variables — both describe aspects of the same continuous swing motion. Keeping tilt unsigned produced stable, correctly-signed coefficients for both variables.
- **Attack angle** was dropped. Despite being negative in direction, its p-value was 0.483 when controlling for the other three variables — not statistically distinguishable from zero. Vertical angle deviation does not independently predict lower launch speed once bat speed, direction, and tilt are accounted for.

The final three-variable physics score:

$$\textit{Physics Score} = -(-5.501 \times \textit{dev\_bat\_speed\_z} + -0.221 \times \textit{dev\_attack\_direction\_z} + -0.254 \times \textit{dev\_swing\_path\_tilt\_z})$$

The sign is flipped so that higher score means more discomfort. This score is computed for all swings, including whiffs, using the coefficients derived from contact swings. Stage 1 achieved $R^2 = 0.091$, meaning the three swing mechanics deviation metrics explain about 9% of variance in launch speed at the individual pitch level — a meaningful result given the enormous noise inherent in single-pitch outcomes.

Chase Deviation

The second component captures pitch recognition failure: did this pitch cause batters to chase outside the zone more than they normally would? For each outside-zone pitch, I subtracted the batter's baseline

chase rate against that pitch type from the binary chase outcome (1 if chased, 0 if not). Positive values mean the batter chased when they normally wouldn't:

$$chase\_deviation = is\_chase - batter\_chase\_rate\_vs\_this\_pitch\_type$$

These per-pitch deviations are then averaged across all outside-zone pitches a pitcher threw of a given type to produce one aggregate chase deviation score per pitcher/pitch-type combination.

The aggregation to pitcher/pitch-type level is intentional and addresses a fundamental data sparsity problem. There are typically too few pitches between any individual pitcher and any individual batter (often 10–20 over a full season) to compute a meaningful matchup-level chase rate. But across all batters a pitcher faced with a given pitch type, the sample is large enough to be reliable (minimum 50 outside-zone pitches required). Critically, the batter-level baseline still controls for the fact that some hitters are naturally more aggressive chasers than others — a pitcher who faces a lineup full of free swingers isn't credited with high chase deviation just because those batters chase a lot generally.

Combining the Components (Stage 2)

The physics score covers all swings; chase deviation covers outside-zone pitches only. They operate on different pitch subsets and different scales. To combine them into a single BDI score, I ran a second OLS regression at the pitcher/pitch-type level, regressing aggregated whiff rate on z-scored versions of both components (n = 1,004 pitcher/pitch-type combos).

I chose whiff rate as the Stage 2 training target for two reasons. First, whiffs are one of the most consequential outcomes in baseball — unlike weak contact, a swinging strike produces zero offensive value and cannot be influenced by defense, park factors, or luck. Second, whiff rate is a deliberately different outcome from Stage 1's launch speed, ensuring the two training steps are independent of each other. The Stage 2 regression:

$$Whiff\ Rate \sim 0.0231 \times physics\_z + 0.0277 \times chase\_z$$

Both coefficients are positive and significant, meaning both components predict more whiffs. Chase deviation has a slightly larger coefficient (0.0277 vs. 0.0231), suggesting pitch recognition failure is a marginally stronger predictor of swing-and-miss than swing mechanics disruption alone at the aggregated level.

What makes this finding particularly notable is the relative magnitude. Using simple baseball intuition, one would expect chase rate — a well-established, directly observable outcome — to be a much stronger predictor of whiff rate than a constructed swing mechanics disruption score. The fact that the physics score coefficient is nearly as large as the chase coefficient is significant: it means the swing mechanics disruption signal, built entirely from bat-tracking deviation metrics, carries almost as much information about a pitcher's whiff-generating ability as their raw chase induction rate. Together, these two components make up the complete Batter Discomfort Index.

Validation

xwOBA allowed was never used in any training step — not in Stage 1 (which used launch speed) and not in Stage 2 (which used whiff rate). It serves as a fully held-out check on whether the BDI captures something real about pitcher effectiveness beyond what it was directly trained on.

Regressing average xwOBA allowed on the BDI across 1,004 pitcher/pitch-type combos:

Validation Outcome	n	R^2	Correlation	p-value
avg xwOBA allowed	1,004	0.152	−0.390	< 0.001

The held-out outcome moves in the expected direction: higher discomfort score predicts lower xwOBA allowed. The R^2 value reflect that this is a two-component metric being validated at the pitcher/pitch-type level rather than a full predictive model — the comparison point is not "can BDI alone predict xwOBA" (it can't fully, and no single metric should be expected to), but "does BDI contain real, independent signal about pitch effectiveness," which it does.

It is also worth noting why validation at the pitcher/pitch-type level (rather than pitch-by-pitch) is the right design. A single batted ball's xwOBA is enormously variable due to defensive positioning, exact contact geometry, park factors, and randomness. Validating at the aggregated level — asking whether pitchers whose pitches cause more discomfort also allow worse outcomes on average over a full season — is both more statistically stable and more relevant to how analysts and front offices actually use pitch-quality metrics.

Results

The top 15 pitcher/pitch-type combinations by BDI score for the 2025 season (minimum 100 swings, 50 outside-zone pitches):

Rank	Pitcher	Pitch	BDI	xwOBA	Whiff%	Physics	Chase Dev
1	Sonny Gray	Sweeper	0.144	0.198	41.5%	1.33	0.210
2	Danny Coulombe	Cutter	0.137	0.279	34.5%	1.35	0.194
3	Fernando Cruz	Splitter	0.137	0.180	52.9%	3.69	0.028
4	Josh Hader	Slider	0.130	0.168	56.2%	2.27	0.115
5	Tanner Banks	Fastball	0.127	0.212	17.1%	1.86	0.137
6	Reese Olson	Slider	0.121	0.241	46.9%	2.01	0.115
7	Mason Miller	Slider	0.120	0.169	53.6%	2.31	0.092
8	Caleb Ferguson	Sinker	0.115	0.272	13.9%	1.32	0.150
9	Graham Ashcraft	Slider	0.115	0.180	37.5%	1.29	0.152
10	Paul Skenes	Changeup	0.111	0.119	42.8%	2.07	0.089
11	Jameson Taillon	Changeup	0.109	0.171	36.1%	1.68	0.113
12	Garrett Whitlock	Changeup	0.108	0.261	31.4%	1.53	0.121
13	Logan Webb	Fastball	0.107	0.142	32.4%	2.68	0.038
14	Jeff Hoffman	Slider	0.103	0.212	47.7%	2.29	0.059
15	Andre Pallante	Sinker	0.100	0.241	12.5%	1.69	0.093

Sonny Gray's sweeper (#1) is the most interesting entry on the leaderboard. Despite a near-average physics score (1.33), it has the highest chase deviation on the entire board (0.210) — batters chase it outside the zone far more than their own baseline against sweepers would predict. This pitch works almost entirely through recognition failure, not mechanical disruption. When batters do make contact, the swing disruption is modest. This is a fundamentally different kind of "nasty" than most of the other top-ranked pitches.

Fernando Cruz's splitter (#3) shows the exact opposite profile: the highest physics score on the entire leaderboard (3.69) but near-zero chase deviation (0.028). Batters recognize it well enough outside the zone and largely don't chase it, but when they do swing, their mechanics are severely disrupted. A pure swing-disruption pitch. The Gray/Cruz contrast illustrates precisely why the two-component structure matters — a single-number metric would obscure these fundamentally different mechanisms.

Josh Hader's slider (#4) and **Mason Miller's slider (#7)** both show strong physics scores and meaningful chase deviation — elite on both dimensions simultaneously. These are the profiles you'd expect from pitches with genuine top-of-the-food-chain reputations.

Tanner Banks' fastball (#5) is one of the most surprising entries. A four-seamer in the top five is unusual — fastballs generally show lower BDI scores because they tend to produce more committed, full swings than breaking balls. His 0.137 chase deviation and 1.86 physics score suggest something unusual about the pitch's deception or movement profile, worth investigating further.

Paul Skenes' changeup (#10) has the lowest xwOBA on the entire leaderboard at 0.119 despite ranking 10th in BDI score. This gap is likely explained by tunneling — his changeup benefits from being paired with one of the best fastballs in baseball, making it harder to recognize than the BDI's pitch-by-pitch measurement captures. Modeling pitch sequencing and fastball-to-secondary tunneling is a natural direction for future work.

Chris Sale's slider (not shown, ranked 19th) appeared consistently in the top 10 across every methodological version of this metric throughout development — an earlier version, an intermediate version, and the final version. With 407 swings (the largest sample on the full leaderboard), its ranking is the most statistically durable finding in the project.

Limitations and Future Work

- **Bat-tracking coverage:** approximately 45–50% of pitches league-wide in 2025 have bat-tracking data, meaning the physics score cannot be computed for all swings. Coverage is improving as Statcast expands deployment.
- **Tunneling not modeled:** the BDI scores each pitch type independently. A changeup paired with an elite fastball is likely more disruptive than its standalone BDI score suggests. Future work could incorporate velocity differential and release-point similarity as proxies for tunneling.
- **Count and leverage context:** a disruptive swing on 0-2 is more consequential than one on 3-0. Weighting by count or leverage state is a natural extension.
- **Year-over-year stability:** this is a single-season metric. Whether leaderboard rankings are stable across seasons — which would indicate the metric is capturing genuine pitcher skill rather than variance — has not yet been tested.
- **Pitcher-side physics analysis:** a companion project (Physics of Swing Disruption) uses the BDI score as a dependent variable and regresses it against pitch physics (velocity, spin, movement, extension) separately for each pitch type, to ask which physical properties of a pitch actually drive batter discomfort.

Appendix A: Full BDI Leaderboard (Top 50)

Sorted by BDI score descending. Minimum 100 swings and 50 outside-zone pitches. All values are 2025 season aggregates. Physics = average per-swing physics score; Chase Dev = average chase deviation on outside-zone pitches.

Rank	Pitcher	Pitch	BDI	xwOBA	Whiff%	Physics	Chase Dev	Swings	OZ Pitches
1	Sonny Gray	ST	0.144	0.198	41.5%	1.33	0.210	277	306
2	Danny Coulombe	FC	0.137	0.279	34.5%	1.35	0.194	142	122
3	Fernando Cruz	FS	0.137	0.180	52.9%	3.69	0.028	140	177
4	Josh Hader	SL	0.130	0.168	56.2%	2.27	0.115	162	198
5	Tanner Banks	FF	0.127	0.212	17.1%	1.86	0.137	129	83
6	Reese Olson	SL	0.121	0.241	46.9%	2.01	0.115	113	140
7	Mason Miller	SL	0.120	0.169	53.6%	2.31	0.092	194	235
8	Caleb Ferguson	SI	0.115	0.272	13.9%	1.32	0.150	108	105
9	Graham Ashcraft	SL	0.115	0.180	37.5%	1.29	0.152	248	265
10	Paul Skenes	CH	0.111	0.119	42.8%	2.07	0.089	166	241
11	Jameson Taillon	CH	0.109	0.171	36.1%	1.68	0.113	119	118
12	Garrett Whitlock	CH	0.108	0.261	31.4%	1.53	0.121	140	145
13	Logan Webb	FF	0.107	0.142	32.4%	2.68	0.038	136	149
14	Jeff Hoffman	SL	0.103	0.212	47.7%	2.29	0.059	174	236
15	Andre Pallante	SI	0.100	0.241	12.5%	1.69	0.093	184	205
16	Andrew Kittredge	SL	0.107	0.207	36.4%	0.70	0.180	247	236
17	Blake Treinen	ST	0.103	0.228	33.6%	1.41	0.118	119	140
18	Garrett Crochet	ST	0.099	0.164	35.4%	1.17	0.163	246	249
19	Chris Sale	SL	0.099	0.154	38.3%	2.20	0.041	407	465
20	Joey Cantillo	CH	0.097	0.201	48.8%	3.61	-0.028	215	255
21	Brant Hurter	ST	0.097	0.188	43.4%	1.33	0.095	113	134
22	Robert Garcia	CH	0.096	0.225	29.2%	1.27	0.117	130	162
23	Tyler Kinley	CU	0.095	0.235	29.6%	1.51	0.085	108	97
24	Jhoan Duran	FS	0.094	0.264	19.2%	1.92	0.040	141	153
25	Logan Gilbert	FS	0.093	0.120	51.7%	2.34	0.016	145	178
26	Jonathan Loaisiga	SI	0.092	0.288	21.1%	0.52	0.187	133	138
27	Tyler Rogers	SL	0.092	0.252	25.7%	1.25	0.100	144	161
28	Shane Smith	CU	0.091	0.220	28.8%	1.62	0.054	118	108
29	Jose Ferrer	CH	0.090	0.221	47.1%	1.38	0.085	119	131
30	MacKenzie Gore	CH	0.089	0.232	49.1%	1.26	0.090	114	129
31	Abner Uribe	SL	0.088	0.235	37.8%	1.52	0.051	241	292
32	Sean Manaea	ST	0.087	0.187	35.6%	1.28	0.075	135	158
33	Chris Stratton	SL	0.086	0.253	33.3%	0.93	0.113	162	192
34	Tylor Megill	SL	0.086	0.215	35.4%	1.51	0.050	113	119
35	Spencer Strider	SL	0.085	0.218	34.1%	1.66	0.034	352	382

36	Reese Olson	CU	0.084	0.229	32.4%	1.44	0.055	102	89
37	Blake Snell	CU	0.083	0.218	40.5%	1.52	0.036	121	106
38	Phil Maton	FC	0.082	0.233	30.3%	1.28	0.065	145	152
39	Chris Bassitt	FC	0.081	0.270	29.2%	0.87	0.104	199	218
40	Ryan Helsley	SL	0.081	0.214	37.4%	1.35	0.044	235	267
41	Porter Hodge	ST	0.079	0.245	31.9%	0.84	0.103	116	129
42	Nick Pivetta	CU	0.078	0.243	31.5%	1.38	0.037	168	152
43	Caleb Thielbar	FF	0.077	0.265	17.2%	1.29	0.049	209	151
44	Garrett Hill	SL	0.077	0.269	28.1%	0.91	0.091	128	141
45	A.J. Minter	SL	0.076	0.256	29.4%	1.11	0.065	136	150
46	Matthew Liberatore	CH	0.076	0.261	27.6%	1.18	0.059	145	163
47	Brent Headrick	FF	0.075	0.259	19.3%	1.13	0.053	135	90
48	Gabe Speier	SL	0.075	0.269	30.2%	1.06	0.067	126	144
49	Tommy Nance	CU	0.074	0.241	27.3%	1.14	0.047	110	98
50	Edward Cabrera	FF	0.073	0.189	25.9%	1.82	0.011	112	103

Appendix B: Complete Pipeline Code

The BDI pipeline runs across five Python scripts in sequence. All scripts read from and write to the same working directory. Scripts are run in the order listed below.

B.1 batter_discomfort_index.py — Data Pull

Pulls all 2025 Statcast pitch data month by month to avoid server timeouts. Saves to disk so subsequent scripts never re-download.

```
from pybaseball import statcast, cache
import pandas as pd
import time

cache.enable()

months = [
    ('2025-03-27', '2025-04-30'), ('2025-05-01', '2025-05-31'),
    ('2025-06-01', '2025-06-30'), ('2025-07-01', '2025-07-31'),
    ('2025-08-01', '2025-08-31'), ('2025-09-01', '2025-09-28'),
    ('2025-10-01', '2025-10-31'), ('2025-11-01', '2025-11-02'),
]

frames = []
for start, end in months:
    print(f'Pulling {start} to {end}...')
    try:
        df_month = statcast(start_dt=start, end_dt=end)
        frames.append(df_month)
        print(f'  Got {len(df_month)} pitches')
        time.sleep(5)
    except Exception as e:
        print(f'  Failed: {e}, skipping...')

df = pd.concat(frames, ignore_index=True)
df.to_csv('statcast_2025_raw.csv', index=False)
print(f'Total pitches: {len(df)}')
```

B.2 build_tables.py — Pitcher Summary + Batter Baselines

Builds pitcher pitch summary and per-batter-per-pitch-type baselines. Chase rate is computed as chases / outside-zone pitches (not mean of is_chase, which would use all pitches as denominator).

```
import pandas as pd

df = pd.read_csv('statcast_2025_raw.csv', low_memory=False)
df = df.copy()
df['is_swing'] = df['description'].isin([
    'swinging_strike', 'swinging_strike_blocked',
    'foul', 'foul_tip', 'hit_into_play'])
df['is_whiff']        = df['description'].isin(['swinging_strike',
'swinging_strike_blocked'])
```

```
df['is_called_strike'] = df['description'] == 'called_strike'
df['is_outside_zone']  = df['zone'].isin([11, 12, 13, 14])
df['is_chase']         = df['is_outside_zone'] & df['is_swing']
df['is_hard_hit']      = df['launch_speed'] >= 95

# Pitcher pitch summary
pitch_summary = df.groupby(['player_name', 'pitcher', 'pitch_type']).agg(
    pitches           = ('pitch_type', 'count'),
    avg_velo          = ('release_speed', 'mean'),
    avg_spin          = ('release_spin_rate', 'mean'),
    avg_pfx_x         = ('pfx_x', 'mean'),
    avg_pfx_z         = ('pfx_z', 'mean'),
    avg_extension     = ('release_extension', 'mean'),
    whiff_rate        = ('is_whiff', 'mean'),
    chase_rate        = ('is_chase', 'mean'),
    xwoba_on_contact  = ('estimated_woba_using_speedangle', 'mean'),
).reset_index()
pitch_summary = pitch_summary[pitch_summary['pitches'] >= 100].round(3)
pitch_summary.to_csv('pitch_summary_2025.csv', index=False)

# Batter baseline per batter + pitch type
batter_baseline = df.groupby(['batter', 'pitch_type']).agg(
    swings               = ('is_swing', 'sum'),
    avg_bat_speed        = ('bat_speed', 'mean'),
    avg_attack_direction = ('attack_direction', 'mean'),
    avg_swing_path_tilt  = ('swing_path_tilt', 'mean'),
    avg_attack_angle     = ('attack_angle', 'mean'),
    avg_swing_length     = ('swing_length', 'mean'),
    outside_zone_pitches = ('is_outside_zone', 'sum'),
    chases               = ('is_chase', 'sum'),
    contact_swings       = ('launch_speed', 'count'),
    avg_launch_speed     = ('launch_speed', 'mean'),
).reset_index()

# Chase rate = chases / outside_zone_pitches (correct denominator)
batter_baseline['batter_chase_rate'] = (
    batter_baseline['chases'] /
    batter_baseline['outside_zone_pitches'].clip(lower=1))

batter_names = df.groupby('batter')['player_name'].agg(
    lambda x: x.mode().iloc[0]).reset_index()
batter_names = batter_names.rename(columns={'player_name': 'batter_name'})
batter_baseline = batter_baseline.merge(batter_names, on="batter", how="left")

# Minimum thresholds
batter_baseline = batter_baseline[
    (batter_baseline['swings'] >= 30) &
    (batter_baseline['outside_zone_pitches'] >= 20)
].reset_index(drop=True)
batter_baseline.round(3).to_csv('batter_baseline_2025.csv', index=False)
print(f"{len(batter_baseline)} batter/pitch-type baselines")
```

B.3 join_deviation.py — Compute Deviation Metrics

Computes per-swing deviation metrics (z-scored within pitch type) and per-outside-zone-pitch chase deviation separately. Attack direction is signed; swing path tilt is unsigned (absolute value).

```
import pandas as pd, numpy as np

df       = pd.read_csv('statcast_2025_raw.csv', low_memory=False)
baseline = pd.read_csv('batter_baseline_2025.csv')
df = df.copy()
df['is_swing']        = df['description'].isin(['swinging_strike',
    'swinging_strike_blocked','foul','foul_tip','hit_into_play'])
df['is_outside_zone'] = df['zone'].isin([11, 12, 13, 14])
df['is_chase']        = df['is_outside_zone'] & df['is_swing']

# ── Part 1: Swing-level deviation ──────────────────────
swings = df[df['is_swing']].copy()
baseline_cols = baseline[[
    'batter','batter_name','pitch_type','avg_bat_speed',
    'avg_attack_direction','avg_swing_path_tilt',
    'avg_attack_angle','avg_swing_length','avg_launch_speed']]
swings = swings.merge(baseline_cols, on=['batter','pitch_type'], how='inner')

# Bat speed: signed, clipped (only slower = discomfort)
swings['dev_bat_speed'] = (swings['avg_bat_speed'] -
swings['bat_speed']).clip(lower=0)
# Attack direction: SIGNED (direction matters, not just magnitude)
swings['dev_attack_direction'] = swings['attack_direction'] -
swings['avg_attack_direction']
# Swing path tilt: UNSIGNED (making it signed caused collinearity with attack
direction)
swings['dev_swing_path_tilt'] = (swings['swing_path_tilt'] -
swings['avg_swing_path_tilt']).abs()
# Attack angle: unsigned, kept for reference (dropped from score -- not significant)
swings['dev_attack_angle'] = (swings['attack_angle'] -
swings['avg_attack_angle']).abs()

# Z-score within pitch type
for col in
['dev_bat_speed','dev_attack_direction','dev_swing_path_tilt','dev_attack_angle']:
    means = swings.groupby('pitch_type')[col].transform('mean')
    stds  = swings.groupby('pitch_type')[col].transform('std')
    swings[f'{col}_z'] = (swings[col] - means) / stds

swings[['game_date','pitcher','player_name','pitch_type','pitch_name',
        'batter','batter_name','p_throws','stand',
        'release_speed','release_spin_rate','spin_axis',
        'pfx_x','pfx_z','release_extension',
        'bat_speed','avg_bat_speed','dev_bat_speed','dev_bat_speed_z',

'attack_direction','avg_attack_direction','dev_attack_direction','dev_attack_direction
_z',

'swing_path_tilt','avg_swing_path_tilt','dev_swing_path_tilt','dev_swing_path_tilt_z',
        'attack_angle','avg_attack_angle','dev_attack_angle','dev_attack_angle_z',
        'launch_speed','avg_launch_speed','description','launch_angle',
        'estimated_woba_using_speedangle']].to_csv('pitch_level_deviation_2025.csv',
index=False)
```

```
# ── Part 2: Chase deviation (outside-zone only) ──────
outside_zone = df[df['is_outside_zone']].copy()
outside_zone = outside_zone.merge(
    baseline[['batter','pitch_type','batter_chase_rate']],
    on=['batter','pitch_type'], how='inner')
outside_zone['chase_deviation'] = (
    outside_zone['is_chase'].astype(float) - outside_zone['batter_chase_rate'])
outside_zone[['game_date','pitcher','player_name','pitch_type',
              'batter','p_throws','stand','is_chase',
              'batter_chase_rate','chase_deviation',
              'zone','plate_x','plate_z']].to_csv(
    'outside_zone_chase_deviation_2025.csv', index=False)
print(f"Saved {len(swings)} swing rows, {len(outside_zone)} outside-zone rows")
```

B.4 build_discomfort_score.py — Two-Stage Model

Stage 1: OLS on launch speed (contact only) to derive physics weights. Stage 2: OLS on whiff rate to combine physics and chase. xwOBA validated as a held-out outcome.

```
import pandas as pd, statsmodels.api as sm, numpy as np

swings = pd.read_csv('pitch_level_deviation_2025.csv', low_memory=False)
chase  = pd.read_csv('outside_zone_chase_deviation_2025.csv', low_memory=False)
swings['is_whiff'] = swings['description'].isin(
    ['swinging_strike','swinging_strike_blocked']).astype(int)

# ── Stage 1: OLS on launch speed (contact swings only) ─
dev_cols = ['dev_bat_speed_z','dev_attack_direction_z','dev_swing_path_tilt_z']
contact  = swings[swings['launch_speed'].notna()].dropna(
    subset=dev_cols + ['launch_speed'])
stage1   = sm.OLS(contact['launch_speed'], sm.add_constant(contact[dev_cols])).fit()
coefs    = stage1.params.drop("const")
neg_coefs = coefs[coefs < 0]   # keep only negative (more deviation = lower launch
speed)
surviving = list(neg_coefs.index)

# Apply to ALL swings (including whiffs), sign flip: higher = more discomfort
swings_scored = swings.dropna(subset=surviving).copy()
swings_scored['is_whiff'] = swings_scored['description'].isin(
    ['swinging_strike','swinging_strike_blocked']).astype(int)
swings_scored['physics_score'] = -(sum(
    neg_coefs[col] * swings_scored[col] for col in surviving))
swings_scored.to_csv('pitch_level_with_discomfort_score_2025.csv', index=False)

# Aggregate physics to pitcher/pitch-type
pitcher_physics = swings_scored.groupby(['player_name','pitch_type']).agg(
    swings=('physics_score','count'), avg_physics=('physics_score','mean'),
    avg_xwoba=('estimated_woba_using_speedangle','mean'),
    whiff_rate=('is_whiff','mean'), avg_launch_speed=('launch_speed','mean'),
).reset_index()
pitcher_physics = pitcher_physics[pitcher_physics['swings'] >=
100].reset_index(drop=True)

# Aggregate chase deviation to pitcher/pitch-type
pitcher_chase = chase.groupby(['player_name','pitch_type']).agg(
```

```
    outside_zone_pitches=('is_chase','count'),
    avg_chase_deviation=('chase_deviation','mean'),
    raw_chase_rate=('is_chase','mean'),
).reset_index()
pitcher_chase = pitcher_chase[pitcher_chase['outside_zone_pitches'] >=
50].reset_index(drop=True)

# Merge and z-score for Stage 2
combined = pitcher_physics.merge(
    pitcher_chase[['player_name','pitch_type','avg_chase_deviation',
                   'outside_zone_pitches','raw_chase_rate']],
    on=['player_name','pitch_type'], how='inner'
).dropna(subset=['avg_physics','avg_chase_deviation','avg_xwoba'])

combined['physics_z'] = ((combined['avg_physics'] - combined['avg_physics'].mean())
                         / combined['avg_physics'].std())
combined['chase_z']   = ((combined['avg_chase_deviation'] -
combined['avg_chase_deviation'].mean())
                         / combined['avg_chase_deviation'].std())

# ── Stage 2: OLS on whiff rate (pitcher/pitch-type) ──
stage2 = sm.OLS(combined['whiff_rate'],
                sm.add_constant(combined[['physics_z','chase_z']])).fit()
physics_coef = stage2.params['physics_z']
chase_coef   = stage2.params['chase_z']

combined['discomfort_score'] = (
    physics_coef * combined['physics_z'] + chase_coef * combined['chase_z'])

# ── Validation: held-out xwOBA ──────────────────────
for outcome in ['avg_xwoba','avg_launch_speed']:
    sub   = combined.dropna(subset=['discomfort_score',outcome])
    model = sm.OLS(sub[outcome], sm.add_constant(sub['discomfort_score'])).fit()
    corr  = sub['discomfort_score'].corr(sub[outcome])
    print(f'{outcome}: R2={model.rsquared:.4f} corr={corr:.4f}')

combined.round(4).to_csv('discomfort_leaderboard_2025.csv', index=False)
```

B.5 regress_by_pitch_type_v2.py — Physics Regression (Project 2)

Companion project. Regresses BDI score on z-scored, handedness-adjusted pitch physics separately for each pitch type. pfx_x is sign-flipped for LHP so positive always = arm-side break.

```
import pandas as pd, statsmodels.api as sm, numpy as np

swings = pd.read_csv('pitch_level_with_discomfort_score_2025.csv', low_memory=False)
swings = swings.copy()
# Handedness adjustment: Statcast records pfx_x from catcher view.
# LHP slider breaks opposite direction of RHP slider -- flip sign for LHP.
swings['pfx_x_adj'] = np.where(swings['p_throws']=='R', swings['pfx_x'], -
swings['pfx_x'])

predictors  = ['release_speed','release_spin_rate','pfx_x_adj',
               'pfx_z','release_extension','spin_axis']
```

```
valid_types = swings['pitch_type'].value_counts()
valid_types = valid_types[valid_types >= 1000].index.tolist()

results_rows = []
for ptype in valid_types:
    subset = swings[swings['pitch_type']==ptype].dropna(
        subset=predictors+['discomfort_score']).copy()
    if len(subset) < 1000: continue
    X_raw = subset[predictors]
    X_z   = (X_raw - X_raw.mean()) / X_raw.std()   # z-score predictors
    model = sm.OLS(subset["discomfort_score"], sm.add_constant(X_z)).fit()
    for var in predictors:
        results_rows.append({
            'pitch_type': ptype, 'n': len(subset), 'variable': var,
            'std_coef': model.params[var], 'p_value': model.pvalues[var],
            'significant': model.pvalues[var] < 0.05,
            'r_squared': model.rsquared,
            'condition_number': model.condition_number})

results_df = pd.DataFrame(results_rows).round(5)
results_df.to_csv('physics_regression_standardized_2025.csv', index=False)

# Pivot: significant coefficients only
sig = results_df.copy()
sig['coef_if_sig'] = sig.apply(
    lambda r: r['std_coef'] if r['significant'] else None, axis=1)
pivot = sig.pivot(index='variable', columns='pitch_type', values='coef_if_sig')
print(pivot.round(4).to_string())
```

Appendix C: Full Regression Tables

C.1 Stage 1 — OLS on Launch Speed

Contact swings only. n = 210,312. R^2 = 0.091. All deviation metrics z-scored within pitch type before regression.

Variable	Coefficient	Std Error	t-stat	p-value	Decision
Intercept	82.583	0.032	2615.9	< 0.001	—
dev_bat_speed_z	−5.501	0.040	−135.9	< 0.001	Kept — dominant
dev_attack_direction_z	−0.221	0.035	−6.3	< 0.001	Kept — signed version
dev_swing_path_tilt_z	−0.254	0.032	−8.0	< 0.001	Kept — unsigned
dev_attack_angle_z	−0.024	0.034	−0.7	0.483	Dropped — not significant

Durbin-Watson: 1.980. Condition number: 1.47 (no multicollinearity). Residual non-normality expected at this scale for individual pitch data.

C.2 Stage 2 — OLS on Whiff Rate

Pitcher/pitch-type level. n = 1,004. R^2 = 0.158. Inputs are z-scored pitcher-level averages.

Variable	Coefficient	Std Error	t-stat	p-value	Interpretation
Intercept	0.2317	0.003	80.0	< 0.001	League avg whiff rate
physics_z	0.0231	0.003	7.8	< 0.001	1 SD more physics → +2.3pp whiff rate
chase_z	0.0277	0.003	9.3	< 0.001	1 SD more chase dev → +2.8pp whiff rate

Durbin-Watson: 2.244. Near-normal residuals (skew 0.247, kurtosis 2.556) — aggregation naturally smooths the distribution. Condition number: 1.24. This is the cleanest regression in the pipeline.

C.3 Held-Out Validation

xwOBA and launch speed were never used in any training step.

Outcome	n	R^2	Coef	Correlation	p-value
avg xwOBA allowed	1,004	0.152	−0.604	−0.390	< 0.001
avg launch speed allowed	1,004	0.131	−23.13	−0.362	< 0.001

C.4 Variable Selection History

Every inclusion/exclusion decision was made empirically. The table below documents each variable tested and the reason for its fate.

Variable	How Tested	Coefficient	Decision	Reason

dev_swing_length	Logistic regression on whiff	Negative	Dropped	Shorter swings predict fewer whiffs — shortening up is successful adaptation, not discomfort
dev_attack_angle (unsigned)	OLS Stage 1	−0.024 (p=0.483)	Dropped	Not significant when controlling for the other three variables
dev_attack_direction (unsigned)	OLS Stage 1	+1.046	Rejected	Positive coefficient: absolute deviation predicts harder contact (aggressive committed swings)
dev_attack_direction (signed)	OLS Stage 1	−0.221	Kept	Signed version correctly predicts lower launch speed (directional signal is real)
dev_swing_path_tilt (signed)	OLS Stage 1	Destabilized others	Reverted to unsigned	Collinear with signed attack direction; condition number rose from 1.47 to 2.42
dev_launch_speed_z	Logistic regression on whiff	−5.13 (quasi-sep)	Dropped	Quasi-separation: NaN launch speed on whiffs created near-perfect artificial predictor
Whiff bonus (+1 SD)	Added to composite score	—	Removed	Introduced circularity with whiff rate validation; incorporated via Stage 2 training instead

---

Full raw CSVs and per-swing data available upon request.