Fantasy Baseball Draft Tool

Technical Documentation for AI Handoff

Claude + Kevin

February 2026

Outline

Repository Structure

```
/home/user/FBB/
|-- draft_tool.html        # Main app (HTML + JS, ~1200
   lines)
|-- create_league_stats.py    # Generates hitter CSVs
|-- create_pitching_stats.py  # Generates pitcher CSVs
|-- normalize_pa.py           # Aligns PA across
   systems
|
|-- DC_Raw_Jan_25.csv         # Depth Charts raw
   projections
|-- The_Bat_Raw_Jan_25.csv    # The Bat raw
   projections
|-- The_BatX_Raw_Jan_25.csv   # The BatX raw
   projections
|
|-- The_Bat_Normalized_PA.csv  # The Bat with DC
   playing time
|-- The_BatX_Normalized_PA.csv # BatX with DC playing
   time
```

Data Flow Overview

Pipeline:

1. Raw projections (DC, The Bat, BatX) downloaded from FanGraphs
2. `normalize_pa.py`: Scale The Bat/BatX to use DC playing time
3. `create_league_stats.py`: Add z-scores, supplement to 600 PA
4. Output CSVs embedded as JS arrays in `draft_tool.html`

WARNING: The player data arrays in draft_tool.html are generated separately. If you regenerate CSVs, you must manually update the JS arrays.

Location: Lines 482-510

```
// Weekly standard deviations from 2024 league data
const HITTING_SD = {
  R: 6.03, HR: 2.93, RBI: 6.72, SB: 2.57,
  SO: 7.45, TB: 15.94, OBP: 0.04, AB: 16.89
};
const PITCHING_SD = {
  L: 1.8346, SV: 1.5362, K: 11.7861, HLD: 1.6383,
  ERA: 1.3141, WHIP: 0.2057, QS: 1.4012, IP: 10.82
};

// League averages (used as opponent baseline)
const HITTING_AVG = {
  R: 28.96, HR: 8.02, RBI: 27.86, SB: 4.74,
  SO: 50.11, TB: 88.87, OBP: 0.32, AB: 208.3
};
```

NOTE: These SDs come from 2024 league weekly results. They determine category leverage.

Replacement Level Constants

```
// Hitter replacement (600 PA season totals)
const HITTER_REP = {
  name: 'Replacement', type: 'H', pa: 600,
  r: 73, hr: 20, rbi: 72, so: 134,
  tb: 224, sb: 9, obp: 0.32
};

// RP replacement (per roster slot, weekly)
const RP_REP_PER_SLOT = {
  ip_wk: 2.480, l_wk: 0.1180, sv_wk: 0.1208,
  hld_wk: 0.8484, k_wk: 2.693,
  er_wk: 0.963, wh_wk: 2.852
};

// SP replacement (per start)
const SP_REP_PER_START = {
  ip: 5.5, l: 0.11, qs: 0.45,
  k: 5.0, er: 2.5, wh: 7.0
};
```

Where Replacement Rates Come From

Hitter replacement (defined in `create_league_stats.py`):

- Rank all hitters by zTotal using Depth Charts
- Take players ranked 155–175 (just beyond $16 \times 9 = 144$ drafted)
- Average their per-PA rates
- Multiply by 600 PA to get season totals

WARNING: OBP is hardcoded to 0.320 (league avg) because the cohort's actual OBP (0.324) exceeded it. Replacement should be neutral, not positive.

RP replacement: Middle relievers who get holds, not saves.
SP replacement: Backend starter (5.5 IP, 4.1 ERA per start).

normalCDF and winProbability (Lines 580-599)

```
function normalCDF(x) {
  // Abramowitz-Stegun approximation
  const a1=0.254829592, a2=-0.284496736, ...
  const sign = x < 0 ? -1 : 1;
  x = Math.abs(x) / Math.sqrt(2);
  const t = 1.0 / (1.0 + 0.3275911 * x);
  const y = 1 - (((a5*t+a4)*t+a3)*t+a2)*t+a1)*t
            * Math.exp(-x*x);
  return 0.5 * (1.0 + sign * y);
}

function winProbability(myMean, oppMean, sd,
                        lowerIsBetter = false) {
  const z = lowerIsBetter
    ? (oppMean - myMean) / (sd * Math.sqrt(2))
    : (myMean - oppMean) / (sd * Math.sqrt(2));
  return normalCDF(z);
}
```

The Win Probability Formula

$$P(\text{win}) = \Phi\left(\frac{\mu_{me} - \mu_{opp}}{\sigma\sqrt{2}}\right)$$

Where:

- μ_{me} = my team's expected weekly total
- μ_{opp} = opponent's expected (uses `HITTING_AVG`)
- σ = weekly SD for that category
- $\sqrt{2}$ comes from $\text{Var}(X - Y) = 2\sigma^2$

NOTE: The $\sqrt{2}$ assumes both teams have same variance. This is a simplification—opponent quality varies—but reasonable.

getHittingProjections (Lines 604-615)

```
function getHittingProjections(teamName) {
  const roster = allTeams[teamName].hitters;
  // Empty slots use HITTER_REP
  const players = roster.map(slot =>
    slot.player || HITTER_REP);

  const proj = {};
  ['r','hr','rbi','sb','so','tb'].forEach(cat => {
    proj[cat.toUpperCase()] = players.reduce(
      (sum, p) => sum + p[cat] / NUM_WEEKS, 0);
  });
  // OBP is average across 9 hitters
  proj['OBP'] = players.reduce(
    (sum, p) => sum + p.obp, 0) / players.length;
  return proj;
}
```

WARNING: Empty slots auto-fill with HITTER_REP. This is how marginal value "vs replacement" works.

calculateMarginalValue (Lines 724-753)

```
function calculateMarginalValue(player) {
  const team = allTeams['My Team'];
  const currentWins = getExpectedWins('My Team')
                      .wins.TOTAL;

  // Temporarily add player to first empty slot
  if (player.type === 'H') {
    const idx = team.hitters.findIndex(
      s => s.player === null);
    if (idx === -1) return -999; // No empty slot
    team.hitters[idx].player = player;
    newWins = getExpectedWins('My Team').wins.TOTAL;
    team.hitters[idx].player = null; // Restore
  }
  // Similar for SP, RP...

  return newWins - currentWins;
}
```

Marginal Value Calculation: Step by Step

When roster is empty:

1. `currentWins` = wins with 9 replacement hitters
2. Add player to slot 0, recalculate `newWins`
3. `marginalValue = newWins - currentWins`

Example: José Ramírez on empty roster

- 9 replacement hitters: expected wins = 2.99/7
- 8 replacement + Ramírez: expected wins = 3.36/7
- Marginal value = 0.378

NOTE: Marginal value depends on current roster. As you draft more players, MV changes based on team composition.

getPitchingProjections (Lines 617-690)

Key logic for SP slots:

```
const SP_SLOTS = 7;
const SP_STARTS_PER_WEEK = 1.1; // per slot

// Drafted SPs contribute their projected stats
sps.forEach(sp => {
  sp_ip += sp.ip_wk;  // Already per-week
  sp_k += sp.k_wk;
  ...
});

// Empty slots filled with replacement
const repSpSlots = SP_SLOTS - sps.length;
sp_ip += repSpSlots * SP_STARTS_PER_WEEK
         * SP_REP_PER_START.ip;
```

WARNING: SP stats in the PITCHERS array are already scaled to 1.1 starts/week. Don't double-scale.

ERA and WHIP Calculation

```
// Total pitching
const total_ip = sp_ip + rp_ip;
const total_er = sp_er + rp_er;
const total_wh = sp_wh + rp_wh;

return {
  ...
  ERA: total_ip > 0
       ? (total_er * 9 / total_ip) : 4.5,
  WHIP: total_ip > 0
       ? (total_wh / total_ip) : 1.3,
  ...
};
```

NOTE: ERA/WHIP are innings-weighted. A reliever pitching 2.5 IP/week has 6% weight on team ERA. This is why RP ERA/WHIP contributions are tiny.

Z-Score Formulas

For counting stats (R, HR, RBI, TB, SB):

```
z_r = (runs / NUM_WEEKS) / SD_R
```

For strikeouts (lower is better):

```
z_so = -(strikeouts / NUM_WEEKS) / SD_SO
```

For OBP (rate stat, 9 hitters share team OBP):

```
z_obp = (obp - AVG_OBP) / 9 / SD_OBP
```

NOTE: The /9 on OBP accounts for one player contributing 1/9 of team OBP. Without it, OBP would be overweighted.

PA Supplementation

```
TARGET_PA = 600

if pa < TARGET_PA:
    gap_pa = TARGET_PA - pa
    runs = runs + gap_pa * REP_R_PER_PA
    hr = hr + gap_pa * REP_HR_PER_PA
    ...
    # OBP is weighted average
    obp = (pa * obp + gap_pa * REP_OBP) / TARGET_PA
    pa = TARGET_PA
```

Purpose: A player with 400 PA shouldn't look worse than 600 PA just due to fewer counting stats. We fill the gap with replacement-level production.

PA Normalization (normalize_pa.py)

```
# Scale The Bat stats to match DC playing time
scale = dc_pa[name] / thebat_pa

counting_stats = ['AB','H','1B','2B','3B','HR',
                  'R','RBI','BB','SO','SB','CS',...]

for stat in counting_stats:
    new_row[stat] = original * scale

# Rate stats (OBP, K%) unchanged
```

Purpose: Compare projection *skill*, not playing time estimates. All systems use DC's PA projections.

Player Data Arrays (Lines 438-480)

```
const HITTERS_THEBAT = [
  {"name":"Shohei Ohtani","type":"H","pa":679,
   "r":126,"hr":48,"rbi":119,"so":160,"tb":347,
   "sb":24,"obp":0.385},
  ...
];

const HITTERS_DC = [...];
const HITTERS_BATX = [...];

const PITCHERS = [
  {"name":"Tarik Skubal","type":"SP","gs":29.6,
   "ip_wk":7.03,"l_wk":0.259,"sv_wk":0.0,
   "hld_wk":0.0,"k_wk":8.48,"qs_wk":0.673,
   "er_wk":2.17,"wh_wk":6.854,...},
  ...
];
```

WARNING: Hitter stats are season totals. Pitcher stats are already

Projection System Toggle

```
let currentProjection = 'thebat'; // default

function getHitters() {
  if (currentProjection === 'thebat')
    return HITTERS_THEBAT;
  if (currentProjection === 'batx')
    return HITTERS_BATX;
  return HITTERS_DC;
}

function switchProjection(value) {
  currentProjection = value;
  renderAll();
}
```

NOTE: PITCHERS array is shared across all projection systems (not toggled).

Setup: Empty Roster

`allTeams['My Team'].hitters` = 9 slots, all null.

`getHittingProjections` fills nulls with `HITTER_REP`:

Stat	Team Weekly Total
R	$9 \times 73/25 = 26.28$
HR	$9 \times 20/25 = 7.20$
SB	$9 \times 9/25 = 3.24$
OBP	0.320

These are compared to `HITTING_AVG` to get win probabilities.

Win Probabilities: Replacement Team

Cat	My Team	Opp (AVG)	SD	P(win)
R	26.28	28.96	6.03	37.7%
HR	7.20	8.02	2.93	42.2%
RBI	25.92	27.86	6.72	41.9%
SO	48.24	50.11	7.45	57.0%
TB	80.64	88.87	15.94	35.8%
SB	3.24	4.74	2.57	34.0%
OBP	0.320	0.320	0.04	50.0%

$\texttt{currentWins} = \sum P = 2.99$

Add Ramírez to Slot 0

Ramírez's season line: 679 PA, 98 R, 30 HR, 94 RBI, 78 SO, 300 TB, 34 SB, .348 OBP

Team becomes 8 replacement + Ramírez:

Cat	Before	After	Δ
R	26.28	27.28	+1.00
HR	7.20	7.60	+0.40
SO	48.24	46.00	−2.24
SB	3.24	4.24	+1.00
OBP	0.320	0.323	+0.003

NOTE: OBP shifts by $(0.348 - 0.320)/9 = 0.003$ because it's averaged across 9 hitters.

New Win Probabilities

Cat	Old P	New P	ΔP
R	37.7%	42.2%	+4.5%
HR	42.2%	46.0%	+3.8%
RBI	41.9%	45.6%	+3.6%
SO	57.0%	65.2%	+8.1%
TB	35.8%	40.9%	+5.1%
SB	34.0%	44.5%	+10.5%
OBP	50.0%	52.2%	+2.2%
Total	2.99	3.36	**+0.378**

`marginalValue = 3.36 - 2.99 = 0.378`

This is what the UI shows as "+MV" for Ramírez.

Why SB and SO Dominate

Cat	Weekly Δ	SD	ΔP
SB	$+1.00$	2.57	$+10.5\%$
SO	-2.24	7.45	$+8.1\%$
TB	$+3.04$	15.94	$+5.1\%$

Key insight: Marginal value $\propto \frac{\Delta}{\text{SD}}$

- SB: $1.00/2.57 = 0.39$ (high leverage)
- TB: $3.04/15.94 = 0.19$ (low leverage)

Categories with small SD have outsized impact per unit.

Things That Will Trip You Up

1. **Hitter stats are season totals**, pitcher stats are weekly
2. **Empty slots auto-fill with replacement**—this is how "vs replacement" works
3. **OBP divides by 9** in z-score calculation (one player = 1/9 of team OBP)
4. **Replacement OBP is hardcoded** to 0.320 (not computed from cohort)
5. **SP ip_wk is already scaled** to 1.1 starts/week in PITCHERS array
6. **ERA/WHIP are ratio stats**—RPs have tiny impact because they pitch few innings
7. **Player arrays are embedded in HTML**—regenerating CSVs requires manual copy

SD Interpretation

Two related but different metrics:

1/SD = marginal leverage per unit

- Used for player valuation
- SB: $1/2.57 = 0.39$ (high leverage)
- K: $1/11.79 = 0.08$ (low leverage)

CV = SD/Mean = category noise

- SB: $2.57/4.74 = 54\%$ (very noisy)
- SO: $7.45/50.11 = 15\%$ (stable)

NOTE: For marginal value, only absolute SD matters. CV tells you how luck-dependent a category is, but doesn't change the valuation formula.

Changes Made (Chronological)

1. Base draft tool with UI and player data
2. Marginal win probability calculation
3. PA supplementation to 600 PA floor
4. Projection toggle (The Bat vs Depth Charts)
5. PA normalization across projection systems
6. 300 PA minimum filter
7. Replacement recalibration (ranks 155-175 cohort)
8. OBP cap at league average (0.320)
9. The BatX projection system added

NOTE: Each feature should be documented here when added. Include: what changed, which files, any gotchas.

Quick Reference: Key Lines in draft_tool.html

What	Lines
HITTERS arrays	438-480
PITCHERS array	480
SD constants	482-483
AVG constants	486-487
Replacement constants	489-510
normalCDF / winProbability	580-599
getHittingProjections	604-615
getPitchingProjections	617-690
getExpectedWins	692-720
calculateMarginalValue	724-753
renderAvailablePlayers	877-980

WARNING: Line numbers may drift as code is edited. Use function names to search.