Fantasy Baseball Draft: Outsmart Opponents with Reinforcement Learning

Bosan Hsu,
Washington University in St. Louis

As a long-time fantasy player competing with friends across the U.S. and Taiwan, I've run into a very real issue: time zone chaos. Our drafts often happen at midnight or 6 A.M. for me. I'm half-asleep, meticulously juggling positions, projections, and rivals' picks, and every second counts. I kept thinking, "Why isn't there a tool that can think this through with me?" That frustration became my motivation. I set out to build an AI assistant that doesn't just rank players, but actually thinks like a manager: reacting to draft changes, roster needs, and rival moves in real time to help me win in every league.

1. Business Problem Framing

Over 10 million fantasy baseball players draft yearly, BUT 90% miss the championship due to outdated, static rankings. Why? One potential reason is that they rely on outdated, static rankings that don't adapt to live draft dynamics or rival strategies. So, how can we adapt to live drafts?

The fantasy sports market in North America is on track to hit $13 billion in 2024 (Mordor Intelligence, 2023). Folks want a smarter game plan, and the reinforcement learning model can learn and provide optimal strategies. This isn't just about picking players, but it's a new way to win. My AI focuses on outdoing rivals for players in 10-team, 23-round H2H leagues.

- **H2H**: Each stat is considered 1 game per Game Week, and Weekly wins, losses, and ties accumulate each week.
- **H2H One Win**: Identical to Head-to-Head Categories, except only 1 point is added to the W-L-T record per week based on whether the manager outperforms more categories than the opponent.

2. Analytics Problem Framing

Objective: Forge an RL beast to spike H2H win rates, crushing rivals pick-by-pick.

How It Works:

- **PCA (Principal Component Analysis) Player Power**: Turns complex stats into simple "impact scores" to show which players matter most.
- **Strategies/Actions**: Different strategies for every round to build the team.
  - e.g., Stack top-tier (max talent) players, Balance across positions, Chase specific stats, Target elite-level gaps other teams miss
- **Opponent Radar**: Tracks rival picks to keep the team one step ahead.
- **Regression**: Evaluates strategy effectiveness and removes underperforming logic to streamline RL decision-making.
- **Reinforcement Learning (RL)**: Learns and improves as the draft progresses, and adjusts picks based on what works. → Regression cuts the weak, RL sharpens the edge.

No more random guesses. By learning to use the best strategy at each turn and refining its approach step by step, the RL model should build the best team in the league.

3. Data Collection & Preprocessing

Data Sources: 2024 data from **Fangraphs** *(Accurate and high-quality content provider)*

- Batting: H, HR, RBI, SB, BB, K, AVG, OBP, SLG
- Pitching: W, OUT, ER, BB, K, ERA, WHIP, QS

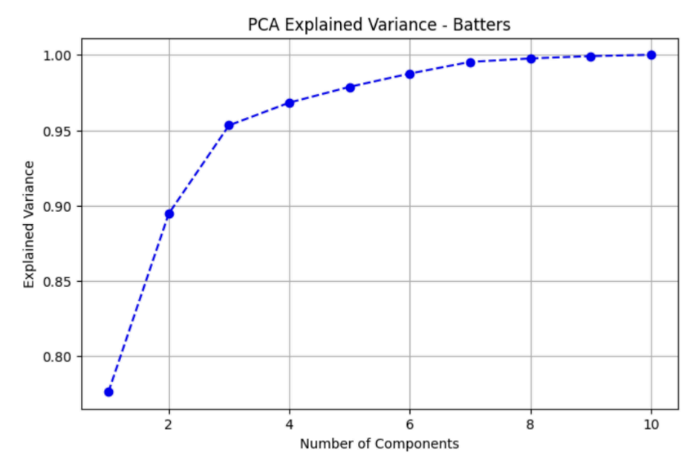
$\}$ Pick the categories based on the league setting.

- Positional Data: C, 1B, 2B, 3B, SS, LF, CF, RF, P

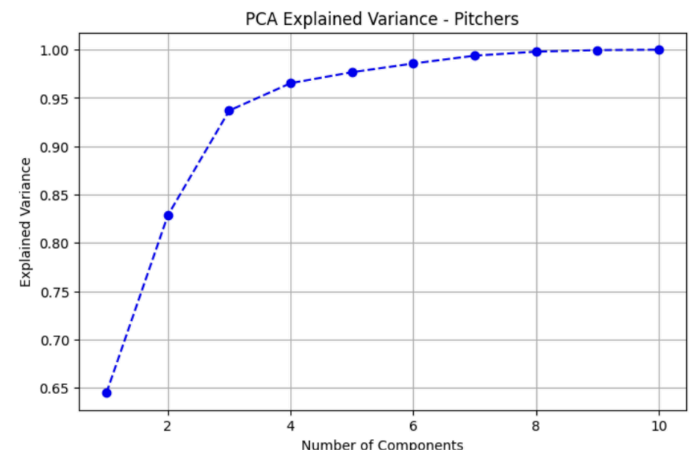
Prep Work:

- Merged batting, pitching, and positional data into one table and filled in missing values.
- Combined LF, CF, RF into OF, split P into SP/RP by starts, and tagged two-way players (e.g., Ohtani).
- Applied PCA to reduce stat noise and capture player value in 1-2 dimensions. (2-3 components grab >90% variance).
- Fuels H2H wins, tracks stats live for 23 slots (1 C, 2 SP, 3 OF, etc.).

The first 2–3 components capture 90%+ of the variance for both batter and pitcher PCAs



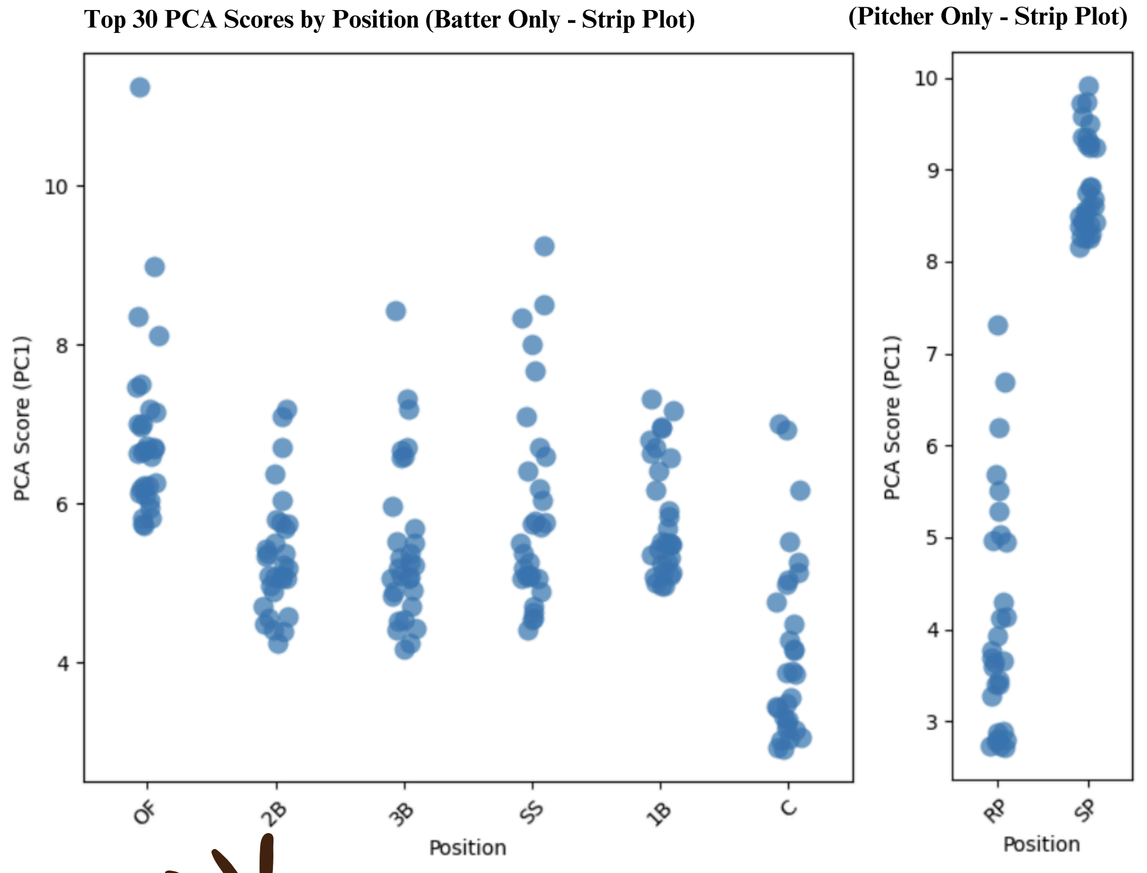
	PC1	PC2	PC3
H_batter	0.337726	-0.228249	-0.041411
HR	0.315297	-0.330819	-0.273902
RBI	0.334281	-0.273900	-0.179463
SB	0.232856	-0.251197	0.925770
BB_batter	0.325709	-0.259067	-0.153325
K_batter	-0.335213	0.199324	0.071729
FPCT	0.295435	0.450500	0.047989
AVG	0.322951	0.377420	0.034628
OBP	0.323354	0.380731	0.018594
SLG	0.325534	0.321159	-0.038781



	PC1	PC2	PC3
W	0.362955	0.090832	0.068181
OUT	0.390759	0.056245	0.009598
H_pitcher	-0.385142	-0.029586	0.044636
ER	-0.377655	0.003862	0.057324
BB_pitcher	-0.371136	-0.005655	-0.071756
K_pitcher	0.384145	0.062907	0.067660
ERA	-0.068957	0.703410	0.101368
WHIP	-0.119788	0.684158	0.024283
QS	0.335344	0.132369	-0.339136
SV+H	0.088006	-0.057686	0.924325

Top 30 PCA scores show how talent is spread across positions. Wider score gaps (e.g., C or RP) signal a steep drop-off after the top players, making elite picks at those spots more valuable. Shallower gaps (e.g., OF or 1B) mean more depth, so missing early stars hurts less."

Draft strategy might not just about who's best, it's about where the drop-offs are.



Top 5 batters and pitchers ranked by PCA (PC1) scores:

Name	PCA_Batter_Score	Pos
Aaron Judge	11.25🔥	OF
Shohei Ohtani	10.94🔥	DH
Elly De La Cruz	9.24	SS
Juan Soto	8.99	OF
Bobby Witt Jr.	8.51	SS

Name	PCA_Pitcher_Score	Pos
Zack Wheeler	9.92	SP
Aaron Nola	9.75	SP
Seth Lugo	9.73	SP
Logan Webb	9.58	SP
Jose Berrios	9.50	SP

4. Methodology & Model Selection

The Brain: A Deep Q-Network (DQN) Reinforcement Learning model designed to boost win rate in Head-to-Head (H2H) and H2H One Win drafts.

Strategies Showdown: *(Every strategy follows the number limitation for each position.)*
Tested five draft strategies in 10-team, 23-round snake drafts, judged by regression:

- **S1 (PCA Max):** Grabs the top PCA score, a pure talent hunt (maxes Relevant_PCA_Score).
- **S2 (Balanced Build):** Mixes PCA with position needs, 70/30 split (PCA + Remaining Position Ratios bonus).
- **S3 (Stat Chase):** Chases HR (rounds 0, 2, 4) and K_pitcher (rounds 1, 3, 5), then mixes HR/K with recent needs, and shifts weight from 70/30 to 30/70 (HR/K vs. slots) → too rigid, compute-limited (fixed rules).
- **S4 (Elite Gaps):** Some positions have stars who are way better than the rest. S4 tries to grab those stars earlier than opponents by tracking where the biggest talent drop-offs are. Not, "Who's the best player?" but "*Where is the biggest edge if I act now?*"
- **Random**: Blindly picks a position, grabs the top PCA player there.

Model Evolution:

- **Step 1**: Ran 100 drafts with S1-S4, and implemented an OLS regression model to evaluate the performance of each strategy (OLS on win rates).
- **Step 2**: Eliminated S3 as an action for RL because of its poor performance.
- **Step 3**: Unleashed RL to pick S1/S2/S4 with mid-round win-rate boosts reward (win_rate - previous_win_rate) and total win rate + (10-Rank)*0.1 as reward. Trained over 500 drafts (num_episodes=500).

Tech Stack:

- Python, TensorFlow (DQN), Statsmodels (regression), Pandas (data processing)

Game Plans:

- Run 10-team, 23-round snake drafts. RL picks adapt via state (stats, round, draft order), slugging it out in H2H matchups (e.g., 22W-14L). Real wins, instead of vague ratings, call the shots.
- *It's a system tuned to final results, not just an overall "average" rating.*

5. Model Building & Training

Baseline Regression Model:

- **Purpose:** Evaluate how well each draft strategy (S1~S4, Random) performs in simulated H2H leagues by analyzing win rates, while controlling for draft order.
- **Implementation:**
  - **Setup:** Simulated 100 full drafts using a mix of S1~S4 and Random across 10 teams. Each team played every other team in simulated H2H matchups, then collected each team's win rate and draft slot. Ran OLS regression to isolate the impact of each strategy, adjusting for the effect of draft order.
  - **Formula:**

$$\text{WinRate} = \beta_0 + \beta_1 \cdot \text{Strategy}_{S1} + \beta_2 \cdot \text{Strategy}_{S2} + \beta_3 \cdot \text{Strategy}_{S3} + \beta_4 \cdot \text{Strategy}_{S4} + \gamma_1 \cdot \text{DraftOrder}_1 + \gamma_2 \cdot \text{DraftOrder}_2 + \cdots + \gamma_9 \cdot \text{DraftOrder}_9 + \varepsilon$$

Note: β_0 represents the baseline win rate using the Random strategy with Draft Pick #0

Reinforcement Learning (RL) Training:

- **Purpose:** While strategies S1~S4 are rule-based, RL learns dynamically by simulating draft outcomes and improving from feedback.
- **Implementation:**
  - **Setup:** Used a Deep Q-Network (DQN) model to train an RL agent over 500 episodes with ε-greedy (0.01), experience replay (32 batch), and target network updates. The agent learns to draft players by maximizing win rate and category dominance. It observes its own team stats, other teams, top players remaining, round, and draft order (**state**), and chooses among S1, S2, or S4 (**actions**).
  - **Reward Signal:**
    - Immediate feedback: Change in win rate after each round.
    - Final reward: Combined win rate + Ranking Bonus after the draft.
  - **Testing:**
    - After training, the RL agent joined 100 new drafts against rule-based teams (S1, S2, S4, Random).
    - Performance was evaluated via win rate and adjusted using the same regression model from earlier.

6. Result & Conclusion

Baseline Regression for Static Strategies:

- **S1 (PCA Max) and S2 (Balanced PCA)** performed significantly better than Random.
- **S4 (Elite Gaps)**, which considers future talent drop-offs, also performed well.
- **S3 (Stat Chase)** wasn't statistically significant (likely due to its rigid rule-based design).
  - *This highlights the limitations of static logic and motivates more dynamic learning.*
  - Also, I removed S3 as RL's action due to its poor performance.
- **Draft Order**: In this Model, draft position has a strong, systematic impact on team win rate, even when controlling for strategy. Teams drafting early tend to perform better than those drafting in the middle-to-late spots.

RL Learning Result:

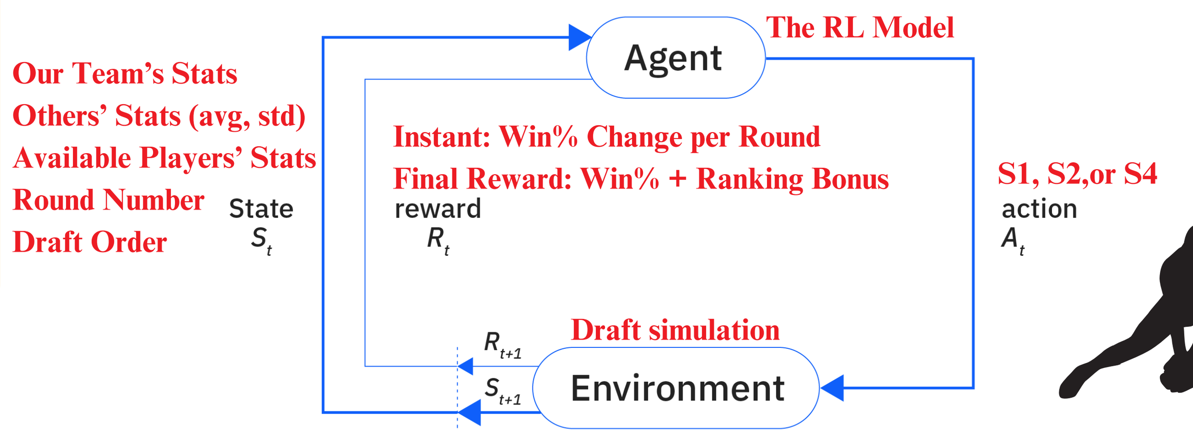
- The RL model outperformed S1, S2, and Random.
- After adjusting for draft order, S4 and RL had the highest adjusted win rates
- **Draft Order**: In this Model, the impact of draft position on win rate is not the same as the previous model. This indicated that the impact of draft position may depend on the mix of strategies used by other teams in the league.
  - This suggests that the *relative value of each draft slot is not universal*. It changes based on how your opponents draft.
  - To train a model that works best for the league, the most effective approach would be to:
    - Collect your league's past draft history (draft order, picks, team performance),
    - Simulate realistic opponent behavior,
    - Train the RL model in that environment.

Conclusion:

- Personalization may help capture league-specific tendencies like who drafts pitcher-heavy early, or how aggressive teams are with sleepers. This can help the RL model tailor its strategy accordingly and have better performance in that specific league.

7. Future Steps & Life Cycle Management

1. **Redesign Strategy S3 Using Reinforcement Learning**
   - Strategy S3 currently follows a hard-coded rule to alternate between targeting HR and K categories. I plan to upgrade this into a reinforcement learning model that learns to dynamically prioritize categories based on draft context, player availability, and round timing.
2. **Incorporate More Drafting Styles**
   - Different users approach drafts differently. Some chase high-ceiling players, others prefer stable floors. I plan to introduce a broader range of drafting strategies, including batter-heavy, pitcher-heavy, and hybrid (high variance + anchors), and compare them using performance metrics.
3. **Add Explainability: "Why Pick This Player?"**
   - Currently, the AI gives recommendations but lacks transparency. I want to build an explainable interface that simulates and compares outcomes between different pick options, helping users understand the impact of each decision and build trust in the model.
4. **Real-Time Draft Simulation with Interactive Feedback**
   - I plan to build a tool where users can input picks as the real draft progresses. The model will update position scarcity, team needs, and category projections live, then return suggestions, risk alerts, and backup picks in real time.
5. **Post-Draft Team Evaluation**
   - After the draft, I want the system to automatically evaluate each roster's predicted strengths, weaknesses, and balance across categories—helping users quickly understand where their team might dominate or struggle, and prepare for trades or pickups early.

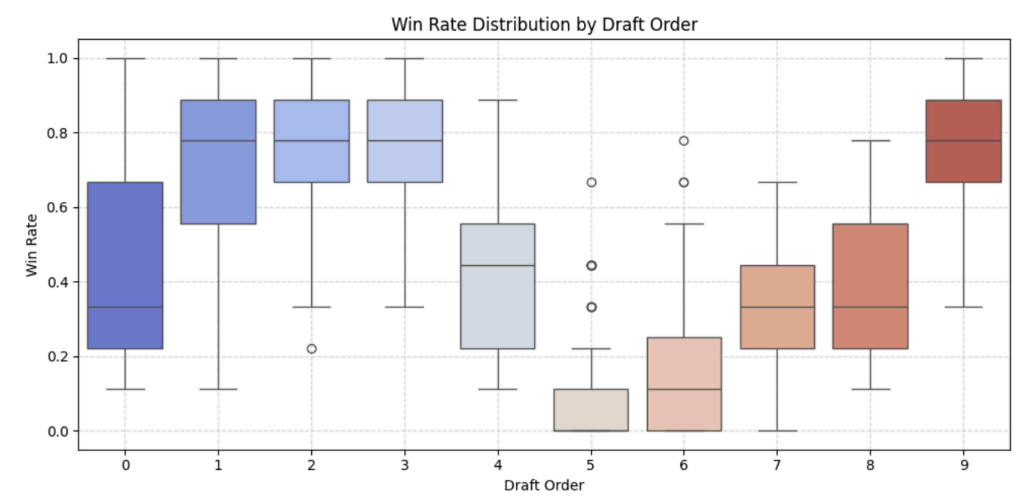


Regression Results for Baseline Model and RL Model:

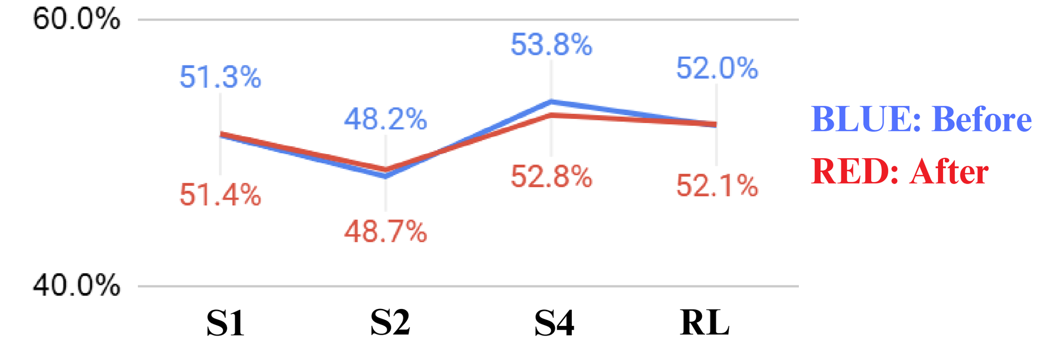
Draft Order	Baseline Coef	RL Model Coef
1	-0.2056***	0.2073***
2	-0.4213***	0.3266***
3	-0.5387***	0.2547***
4	-0.8461***	-0.0854***
5	-0.4677***	-0.3906***
6	-0.5049***	-0.3224***
7	-0.7692***	-0.1260***
8	-0.6383***	-0.1196***
9	-0.4839***	0.2554***
Strategy	**Baseline Coef**	**RL Model Coef**
S1	0.0592***	0.0653***
S2	0.0677***	0.0385***
S3	0.0239	—
S4	0.0516***	0.0797***
RL	—	0.0729***

RL significantly improved the performance, but may need to learn more actions to become even better.

Draft Order's Impact on Win Rate (RL Model):



Win Rate Before/After Adjusting for Draft Order (RL Model):



Draft AI Prototype Partial Screenshot

```
Round 0, Team 0: Picked Zack Wheeler_pitcher - Assigned SP
Round 0, Team 1: Picked Aaron Judge_batter - Assigned OF
Debug: Elite player Shohei Ohtani_batter assigned to UTIL at Round 0
Round 0, Team 2: Picked Shohei Ohtani_batter - Assigned UTIL
Round 0, Team 3: Picked Bryce Harper_batter - Assigned 1B
Round 0, Team 4: Picked Pete Alonso_batter - Assigned 1B
Round 0, Team 5: Picked Jose Ramirez_batter - Assigned 3B
Round 0, Team 6: Picked Aaron Nola_pitcher - Assigned SP
Round 0, Team 7: Picked Elly De La Cruz_batter - Assigned SS
Round 0, Team 8: Picked Luis L. Ortiz_pitcher - Assigned RP
Round 0, Team 9: Picked William Contreras_batter - Assigned C
Round 1, Team 9: Picked Bobby Witt Jr._batter - Assigned SS
Round 1, Team 8: Picked Juan Soto_batter - Assigned OF
Round 1, Team 7: Picked Nick Martinez_pitcher - Assigned RP
Round 1, Team 6: Picked Tyler Alexander_pitcher - Assigned RP
Round 1, Team 5: Picked Kyle Schwarber_batter - Assigned OF
Round 1, Team 4: Picked Jose Altuve_batter - Assigned 2B
Round 1, Team 3: Picked Brent Rooker_batter - Assigned OF
Round 1, Team 2: Picked Gunnar Henderson_batter - Assigned SS
Round 1, Team 1: Picked Jose Urena_pitcher - Assigned RP
Round 1, Team 0: Picked Jarren Duran_batter - Assigned OF
```

RL recommends for Team 1: Hunter Brown, SP

Accept | Nah, choose others: Input Player Name