Bears Comeback Analytics Dashboard

Sources & Data Methodology

Chicago Bears 2025–26 Season · Companion Documentation

February 2026 | Perplexity Computer

Table of Contents

1 Sources by Dashboard Section

All data in the dashboard was gathered from publicly available sources during the 2025–26 NFL season. Below is a detailed breakdown of every source used, organized by dashboard section.

1.1 Season Overview & Game Log

- **ESPN Bears team page**[1] — game-by-game scores, 11-6 record, full schedule
- **CBS Sports**[2] — box scores, game results, scoring summaries
- **Comeback game identification** — cross-referenced all box scores where the Bears trailed at any point in Q4 and ultimately won; 7 of 17 regular-season games met this two-part criterion

1.2 Caleb Williams QB Stats & Deep Dive

- **ESPN & CBS Sports**[1][2] — season stat lines: 3,942 passing yards, 27 TD / 7 INT, 90.1 passer rating, 58.1% completion rate
- **PlayerProfiler**[3] — advanced metrics including 3.03s time to throw (TTT), 4.0% sack rate, 6.4% hit rate, play-action usage rate
- **Next Gen Stats**[4] — time to throw distributions, air yards, aggressiveness metrics
- **Sharp Football Analysis**[5] — comeback-mode mechanical shifts: 2.7s TTT, 42.9% throws on the run, off-target rate drop from 22% to 12% in comeback situations
- **Bear Goggles On**[6] — analytical deep dives on Williams' footwork mechanics and pocket behavior
- **Sporting News**[7] — passer rating by deficit context: 101.2 when trailing by 1–8 points

1.3 Wide Receiver & TE Stats (Original + New Splits)

- **ESPN & CBS Sports**[1][2] — baseline receiving stats: receptions, yards, TDs, targets for all pass catchers
- **PlayerProfiler (Odunze)**[8] — alignment splits: X receiver ~61% of snaps, slot rate 31.7% (up from 24.4% in early season)
- **RotoWire**[9] — Odunze average depth of target (aDOT): 13.9 yards
- **PlayerProfiler**[8] — total air yards (28.5 for Odunze); Burden YAC stats (334 total, 7.1 per reception); Moore backfield snap count (33 snaps)
- **PlayerProfiler (Loveland)**[10] — 58 receptions, 713 yards, 6 TDs, 1.97 yards per route run (YPRR), 54% route participation rate
- **Sports Illustrated**[11] — PFF grades referenced for Loveland: Top-5 receiving grade among all tight ends
- **Chicago Tribune**[12] — game recaps documenting Loveland's mid-season emergence as a primary red-zone threat

1.4 Running Backs

- **ESPN & CBS Sports**[1][2] — D'Andre Swift (1,087 rushing yards) and Stacey Monangai (783 rushing yards) base stat lines
- **Sharp Football Analysis**[5] — yards per carry by run location: Swift outside 5.6, inside 3.5; Monangai inside 4.9
- **PlayerProfiler**[3] — receiving splits and formation-specific usage data

1.5 Offensive Line

- **ESPN PBWR metric**[1] — Pass Block Win Rate (ranked #1 in NFL), Run Block Win Rate (ranked #5)
- **Sports Illustrated & Chicago Tribune**[11][12] — Joe Thuney: 98% PBWR, 87.7 PFF pass-block grade (PFF data as cited by SI/Tribune)
- **ESPN team stats**[1] — penalty counts: 28 false starts, 20 holding calls on the season
- **Sharp Football Analysis**[5] — under-center rate (49.6%), zone run scheme frequency

1.6 Defense

- **ESPN team stats**[1] — 33 takeaways, +22 turnover differential (league-leading)
- **Sharp Football Analysis**[5] — defensive coverage: man/zone split 49%/51%, zone interception rate 6.6% (14 zone INTs)
- **ESPN, CBS Sports, Chicago Tribune**[1][2][12] — individual defenders: Kevin Byard 7 INT; Jaylon Wright 5 INT + 2 return TDs; Tremaine Edmunds 112 tackles; Montez Sweat 10 sacks; Austin Booker 4.5 sacks with a 90.0 PFF grade

---

Source References

1. ESPN Bears Team Page: https://www.espn.com/nfl/team/_/name/chi/chicago-bears
2. CBS Sports Bears: https://www.cbssports.com/nfl/teams/CHI/chicago-bears/
3. PlayerProfiler — Caleb Williams: https://www.playerprofiler.com/nfl/caleb-williams/
4. NFL Next Gen Stats: https://nextgenstats.nfl.com/
5. Sharp Football Analysis: https://www.sharpfootballanalysis.com/
6. Bear Goggles On: https://www.beargogglesonsportsblog.com/
7. Sporting News — Bears: https://www.sportingnews.com/us/nfl/team/chicago-bears
8. PlayerProfiler — Rome Odunze: https://www.playerprofiler.com/nfl/rome-odunze/
9. RotoWire: https://www.rotowire.com/
10. PlayerProfiler — Colston Loveland: https://www.playerprofiler.com/nfl/colston-loveland/
11. Sports Illustrated — Bears: https://www.si.com/nfl/bears
12. Chicago Tribune — Bears: https://www.chicagotribune.com/sports/bears/

2 Data Wrangling Techniques

The dashboard combines multiple public data sources into a unified view. Because no single API provides clean 'comeback mode' vs. 'non-comeback' splits, several estimation and triangulation techniques were employed.

2.1 Identifying Comeback Games

All 18 games (17 regular season + 1 Wild Card) were filtered by two criteria: **(1)** the Bears trailed at any point in the fourth quarter, and **(2)** the Bears won the game. Seven games met both criteria. This is a simple binary classification — a game is either a comeback or it is not.

2.2 Intra-Game Splits (Comeback vs. Non-Comeback Mode)

No public API cleanly separates 'comeback mode drives' from 'non-comeback drives' within a single game. The following approach was used:

- Used play-by-play game logs and box scores to identify the specific moments when the Bears were trailing in each comeback game
- Cross-referenced analytical breakdowns from Sharp Football Analysis, Bear Goggles On, and the Chicago Tribune discussing halftime adjustments and Q4 play-calling shifts
- Estimated per-mode splits by triangulating season-long stats, game-specific stat lines, and published analyst commentary on scheme changes

2.3 Target Share & aDOT Splits by Mode

- Took each player's season-long per-game averages as the **non-comeback baseline**
- Used game-specific target counts from box scores in the 7 comeback games, weighted toward Q4/comeback drive production as described in game recaps
- aDOT shifts were estimated from route-type changes described in film breakdowns — e.g., D.J. Moore running more digs/slants vs. go routes implies a lower aDOT in comeback mode

2.4 Normalization for Charts

- **Radar charts** normalize all metrics to a 0–100 scale for apples-to-apples visual comparison across different stat categories
- **Bar charts** use raw values with consistent y-axis scales across comparison pairs to preserve absolute magnitude differences

■ **Key Caveat:** Comeback-mode metrics are *estimates* derived from game film analysis, play-by-play data, and published analytics — not from a single proprietary database. For academic or journalistic use, these figures should be noted as 'estimated from play-by-play and film analysis.'

3 GitHub Deployment Guide

The dashboard is a static site (HTML + CSS + JS) with no server-side dependencies. It can be deployed for free on any static hosting platform.

Option 1: GitHub Pages (Recommended — Free)

1. Create a new GitHub repository (e.g., `bears-comeback-analytics`)
2. Push the three dashboard files (`index.html`, `style.css`, `app.js`) to the `main` branch
3. Navigate to **Settings** → **Pages** → **Source**: select "Deploy from a branch" → Branch: `main`, folder: `/` (root)
4. Your site will be live at `https://yourusername.github.io/bears-comeback-analytics/`

Option 2: Netlify via GitHub

1. Push your code to GitHub
2. Connect the repo to Netlify: go to `netlify.com` → "Import from Git"
3. Build settings: no build command needed; publish directory: `/`
4. Auto-deploys on every push to the connected branch

Option 3: Vercel

1. Push your code to GitHub
2. Import the project at `vercel.com`
3. Framework preset: select "Other"
4. Deploys automatically on each push

4 Code Cleanup Recommendations

The following recommendations improve maintainability, performance, and accessibility of the dashboard codebase.

4.1 Extract Data into a Separate File

Move all data arrays (`gameLog`, `comebackQB`, `nonComebackWinsQB`, `lossesQB`, `comebackProfiles`) into a dedicated `data.js` file. Import it via a `<script>` tag before `app.js`. This cleanly separates data from visualization logic.

4.2 Create a Chart Factory / Config Pattern

Many charts share similar options (colors, tooltips, datalabels). Create a shared config object and merge it per chart to reduce duplication:

```
const baseBarOptions = { responsive: true, ... };

function createBarChart(ctx, labels, datasets, overrides) {
return new Chart(ctx, {
type: "bar",
data: { labels, datasets },
options: { ...baseBarOptions, ...overrides }
});
}
```

4.3 Use CSS Custom Properties for Chart Colors

Chart colors are currently hardcoded as JS constants that duplicate CSS variables. Read them from CSS at runtime instead:

```
const style = getComputedStyle(document.documentElement);
const ORANGE = style.getPropertyValue("--orange").trim();
```

4.4 Modularize CSS

Split `style.css` into logical partials: `layout.css`, `components.css`, `charts.css`, `responsive.css`. Use a simple concatenation build step or CSS `@import` statements.

4.5 Add aria-labels to Charts

For accessibility, add `aria-label` attributes to `<canvas>` elements describing the chart content. This enables screen readers to convey the chart's purpose to visually impaired users.

4.6 Lazy-Load Below-Fold Charts

Use the `IntersectionObserver` API (already in the code for scroll animations) to initialize charts only when their section scrolls into view. This reduces initial page load time and improves perceived performance.

4.7 Add a .gitignore and README.md

Include a `README.md` with a project description, screenshot, data sources summary, and methodology note. Add a `.gitignore` for any build artifacts or editor config files.

End of Document | Bears Comeback Analytics — Sources & Data Methodology | Perplexity Computer, February 2026