sportsdataverse-py :: CHEAT SHEET

The Python sister to the SportsDataverse R packages (wehoop, hoopR, cfbfastR, baseballr, fastRhockey, nflfastR). One `pip install` gives polars-first, tidy access to play-by-play, box scores, schedules, rosters and odds across 15+ sports, plus bundled EP / WP / rating / projection models and release-dataset loaders. Page 1 of 4: getting started, coverage map, the ESPN cross-league layer, utilities.

v0.0.75

Python 3.9 – 3.14 · polars 1.x
py.sportsdataverse.org

GET STARTED

```
pip install sportsdataverse
pip install "sportsdataverse[all]"  # + models, curl_cffi, tests
uv add sportsdataverse

import sportsdataverse as sdv
from sportsdataverse.cfb import load_cfb_pbp
```

Every public callable is also on the flat `sdv.*` namespace (4,600+ names). Sub-packages: `cfb nfl nba nbagl wnba mbb wbb nhl pwhl hockey hockeytech mlb baseball soccer cricket football odds wexp`.

RETURN CONVENTIONS

Flag	Effect
default	`polars.DataFrame`, snake_case columns
`return_as_pandas=True`	pandas frame (same data)
`raw=True`	scrapers: untouched JSON `dict`
`return_parsed=False`	ESPN wrappers: tidy frame (default) → raw `dict`
multi-table	`dict[str, pl.DataFrame]` keyed by section
empty / malformed	zero-row frame with documented schema — never raises

THREE SURFACES

1 · Release-dataset loaders

`load_<sport>_<dataset>(seasons, return_as_pandas=False)` reads parquet from the SportsDataverse GitHub releases (the `*-data` repos). Cached; zero scraping.

```
pbp = load_cfb_pbp(seasons=[2023, 2024])
sch = sdv.load_nba_schedule(seasons=range(2020, 2025))
```

2 · Live API wrappers

`espn_<league>_*`, `nba_stats_*`, `wnba_stats_*`, `nhl_web_*`, `nhl_edge_*`, `mlb_*`, `mlb_statcast_*`, `<hockeytech-league>_*`, `ncaa_mbb_*`, `toa_*` (odds).

3 · Processing + models

PBP processors (`CFBPlayProcess`, `NFLPlayProcess`), `enrich_nfl_pbp`, `nba_possessions`, ratings, simulations, projections — see pages 2–4.

COVERAGE MAP

Sport	Sub-package	Live sources	R sibling
NFL	`nfl`	ESPN · api.nfl.com · PFF	nflfastR · nflreadr
CFB	`cfb`	ESPN · stats.ncaa.org · Fox · Yahoo · 247 · On3 · PFF	cfbfastR · recruitR
NBA · G-League · Summer	`nba nbagl`	ESPN · stats.nba.com · Fox	hoopR
WNBA	`wnba`	ESPN · stats.wnba.com	wehoop
Men's college bb	`mbb`	ESPN · stats.ncaa.org · Torvik	hoopR · bigballR
Women's college bb	`wbb`	ESPN · stats.ncaa.org · Bart	wehoop · wbigballR
NHL	`nhl`	ESPN · api-web.nhle.com · EDGE · stats REST · records	fastRhockey
PWHL + 19 minor/junior	`pwhl hockey.*`	HockeyTech / LeagueStat	fastRhockey
College hockey M/W	`hockey.mch .wch`	ESPN	—
MLB	`mlb`	ESPN · statsapi.mlb.com · Baseball Savant	baseballr
College baseball / softball	`baseball`	ESPN · stats.ncaa.org	baseballr · softballR
Soccer (12 leagues)	`soccer.*`	ESPN	—
Cricket	`cricket`	ESPN + bundled WP	—
UFL · XFL · CFL · AAF	`football.*`	ESPN + spring EP/WP	—
Odds & lines	`odds`	The Odds API	oddsapiR
Win-expectancy lab	`wexp`	CFB / NFL market oracles	—

Release data repos: `cfbfastR-data`, `nfl-data`, `hoopR-nba-data`, `hoopR-mbb-data`, `wehoop-wnba-data`, `wehoop-wbb-data`, `fastRhockey-*-data`, `baseballr-data`, `ncaa-mbb/wbb-data`, `*-stats-data`.

ESPN CROSS-LEAGUE LAYER

One core (`_common_espn`) × N thin league modules. Every league gets the same **121 endpoint short names** as `espn_<prefix>_<short>` (≈125 wrappers per league; 800+ total across nba wnba nfl cfb mbb wbb nhl mlb, plus soccer, cricket, college baseball/hockey and spring football).

```
from sportsdataverse.nba import espn_nba_team_roster
df  = espn_nba_team_roster(team_id=13)
# polars (default)
raw = espn_nba_team_roster(team_id=13,

return_parsed=False)   # raw dict
```

Family	Short names
Schedule	`schedule calendar scoreboard games season_weeks season_week_games`
Teams	`teams team team_roster team_schedule team_record team_injuries team_depthcharts franchise venue`
Games	`summary game game_rosters game_plays play_participants game_odds game_officials game_predictor game_probabilities game_propbets`
Players	`player_stats player_gamelog player_splits player_career_stats player_bio player_contracts player_vs_player`
League	`standings leaders news injuries transactions draft awards conferences futures season_qbr`
NCAA extras	`rankings fpi recruits recruiting_rankings` (mbb wbb cfb …)

Parser layer

`parse_summary(payload, section=None)` → 21 sub-frames: `boxscore_player boxscore_team plays winprobability leaders game_info officials header season_series against_the_spread standings broadcasts format pickcenter odds article injuries news drives drive_plays scoring_plays`. Generic fall-throughs: `parse_single_entity`, `parse_items`.

Full PBP processors (tidy, per game)

```
espn_nba_pbp(401584793)   espn_wnba_pbp(...)
espn_mbb_pbp(...)         espn_wbb_pbp(...)
espn_nhl_pbp(...)         espn_mlb_pbp(...)
CFBPlayProcess(gameId=401628334)   # page 2
NFLPlayProcess(gameId=401671889)   # page 2
```

DISCOVERY

```
sdv.find_team("Duke", "mbb")
sdv.find_athlete("Caitlin Clark", "wnba", team="Fever")
sdv.find_event("2024-11-30", "cfb", home="Ohio State")
sdv.function_count()            # per league
sdv.list_functions("nhl", search="pbp")
```

CLI mirrors it: `sdv find-team Duke --league mbb`, `sdv function-count`, `sdv cache stats|clear|mode`.

CACHE

```
sdv.set_cache_mode("filesystem")  # memory | filesystem | off
sdv.set_default_ttl(3600)
sdv.cache_stats();  sdv.clear_cache()
# NFL has its own nflreadpy-style config
from sportsdataverse.nfl import update_config
update_config(cache_mode="filesystem", cache_duration=3600)
```

Env: `SDV_PY_NFL_CACHE` · `SDV_PY_NFL_CACHE_DIR` · `SDV_PY_NFL_CACHE_DURATION`.

HTTP, TIDY & RELEASE HELPERS

`dl_utils.download(url)` — single retrying HTTP chokepoint. `underscore() camelize() kebabize()`, `flatten_json_iterative()`, `key_check()`.

`sportsdataverse.release` — port of the sportsdataversedata R package: `sportsdataverse_upload()`, `sportsdataverse_save()` (parquet / rds / csv.gz), `gh_cli_release_upload()`, `gh_cli_release_assets()`, `gh_cli_release_tags()`.

Errors: `NoESPNDataError` · `SeasonNotFoundError` · `RawStoreMissError`.

ENV VARS WORTH KNOWING

`SDV_PY_LIVE_TESTS=1`	run gated live tests
`SDV_PY_NBA_STATS_LIVE=1`	stats.nba.com tests (residential IP)
`SDV_PY_NBA_RAW_JSON_DIR`	read-through raw JSON store
`SDV_PY_NBA_RAW_JSON_READONLY=1`	offline: miss → error
`SDV_<LEAGUE>_API_KEY`	override a HockeyTech key
`CFBD_API_KEY` · `ODDS_API_KEY`	third-party keys

COLLEGE FOOTBALL · SPORTSDATAVERSE.CFB

LIVE SCRAPERS

```
from sportsdataverse.cfb import *
espn_cfb_schedule(dates=2024, week=1)
espn_cfb_teams();
espn_cfb_calendar(season=2024)
espn_cfb_game_rosters(game_id=401628334)
espn_cfb_team_roster(team_id=52)
espn_cfb_player_stats(athlete_id=4426502,
season=2024)
```

Play-by-play (ESPN, full pipeline)

```
proc = CFBPlayProcess(gameId=401628334)
pbp  = proc.espn_cfb_pbp()          # dict of
frames
pbp  = proc.run_processing_pipeline()  #
EPA/WPA added
box  = proc.create_box_score()
# offline rebuild from on-disk raw JSON
CFBPlayProcess(gameId, path_to_json=raw_dir,
               odds_override=
{...}).cfb_pbp_disk()
```

Per-play participants come from ESPN's `participants[]` (`cfb_play_participants`) → `{type}_player_name` / `_id` + list columns.

Other sources

stats.ncaa.org: `parse_cfb_ncaa_pbp` · `parse_cfb_ncaa_drives` · `_linescore` · `_officials` · `_team_stats` · `_player_stats` (`cfb_ncaa_pbp` / `cfb_ncaa_box`) · `fox_cfb_pbp` · Yahoo ext · `pff_*` (premium) · Recruiting: `sports247_*` (247 RDB), `on3_*`, `espn_cfb_recruits`.

CFB MODELS (BUNDLED)

Artifact	Purpose
`ep_model`	expected points (rule-era)
`wp_naive` · `wp_spread`	win probability ± Vegas spread
`cfb_cp_model`	completion prob · CPOE
`qbr_model`	QBR-style rating
`fg_model` · `two_pt_model`	kick / conversion prob
`fd_model` · `xpass_model`	4th-down go/kick/punt · pass rate
`punt_distribution` · `cfb_field_position_ep`	decision surfaces

CFB ANALYTICS

```
r = cfb_ratings(seasons=2024, as_of_date=...)
# opp-adjusted
cfb_predict_games(games, r)             #
spread / WP / total
cfb_simulations(games, teams,
simulations=10000)
cfb_adjusted_epa(plays);
cfb_adjusted_tempo(seasons=2024)
cfb_advanced_stats(seasons=2024)        #
success, explosiveness…
cfb_standings(...); cfb_playoff_seeds(...);
cfb_resume(...)
```

Also `cfb_fourth_down` · `cfb_two_point` · `cfb_field_position` · `cfb_opponent_adjust` · `cfb_season_odds` · `cfb_returning_production` · `cfb_roster_talent` · `cfb_transfer_impact` · `cfb_draft_projection` · `cfb_recruiting_projection` · `cfb_crosswalk`.

CFB RELEASE LOADERS (`LOAD_CFB_*`)

Group	Datasets
Core	`pbp pbp_r model_pbp schedule rosters game_rosters team_box player_box drives linescores team_info play_participants`
Stats	`passing rushing receiving percentiles team_summaries team_summaries_weekly`
Advanced	`adv_team adv_team_gamelog adv_passing adv_rushing adv_receiving adv_defensive adv_defensive_players adv_drives adv_situational adv_specialists adv_turnover`
Ratings	`ratings ratings_weekly fpi_weekly power_index`
Betting	`betting betting_lines`
Recruiting	`recruits recruiting_proj team_talent returning_production` + `load_recruit_classes`
Crosswalks	`teams_crosswalk rosters_crosswalk schedule_crosswalk`

```
load_cfb_pbp(seasons=[2024])
load_cfb_ratings_weekly(seasons=[2024])
load_cfb_betting_lines(seasons=[2024])
```

NFL · SPORTSDATAVERSE.NFL

NFLREADPY-PARITY LOADERS

24 canonical `load_nfl_*` loaders; inside `sportsdataverse.nfl` they are also exported under nflreadpy's names (`load_pbp`, `load_schedules`, …).

```
import sportsdataverse.nfl as nfl
pbp = nfl.load_pbp([2024])
sch = nfl.load_schedules([2024])
ngs =
nfl.load_nextgen_stats(stat_type="passing")
adv = nfl.load_pfr_advstats(stat_type="pass",

summary_level="season")
qbr = nfl.load_espn_qbr(seasons=[2024])
```

Loaders

`pbp schedule teams players rosters weekly_rosters player_stats team_stats pbp_participation snap_counts depth_charts injuries officials trades contracts draft_picks combine ff_playerids ff_rankings ff_opportunity ftn_charting nextgen_stats pfr_advstats espn_qbr` + `load_nfl_ratings_weekly`

Config: `get_config()` `update_config()` `reset_config()` `clear_cache()` · `datasets.team_abbr_mapping`, `team_abbr_mapping_norelocate`, `player_name_mapping` · `get_current_season()` `get_current_week()`.

LIVE NFL SOURCES

```
espn_nfl_schedule(season=2024, week=1)
espn_nfl_teams();
espn_nfl_game_rosters(401671889)
NFLPlayProcess(gameId=401671889).run_processing_pipeline()
# api.nfl.com "Shield" (11 endpoints, bearer
auth)
nfl_standings(season=2024); nfl_rosters(...);
nfl_weeks(...)
```

Parsers: 11 dedicated `parse_nfl_*`. PFF: `pff_*` (premium API, Clerk session).

EP / WP / EPA / WPA (EP_WP)

Single owner of NFL model application — a faithful port of nflfastR's `helper_add_ep_wp.R`. Construction modules emit a frame; `ep_wp` enriches it.

```
from sportsdataverse.nfl.ep_wp import *
df = enrich_nfl_pbp(pbp)     #
ep→epa→wp→wpa→cp→cpoe→xyac
calculate_expected_points(df);
calculate_epa(df)
calculate_win_probability(df);
calculate_wpa(df)
calculate_completion_probability(df);
calculate_xyac(df)
calculate_xpass(df)
```

Parity vs nflverse: `ep` 0.996 · `epa` 0.994 · `wp` 0.997 · `vegas_wp` 0.998. Every lag is `.over("game_id")`.

Decision surfaces

`nfl_fourth_down`: `get_go_wp()` `get_fg_wp()` `get_punt_wp()` (nfl4th-style, offline). `nfl_playcall` · `nfl_kicker_rating` · `nfl_special_teams` · `nfl_gamescript` · `nfl_series`.

NFL MODELS (BUNDLED .UBJ)

`ep_model` (18 feat) · `wp_spread` (12) · `wp_naive` (11) · `cp_model` (18) · `fg_model` · `qbr_model` · `two_pt_model` · `xpass_model` · `nfl_playcall` · `punt_data.parquet`. xYAC (76-class) downloads on first use from `nfl_model_artifacts`.

NFL RATINGS, SIMS, PROJECTIONS

`nfl_ratings` · `nfl_simulations` (season / playoff sim; nflseedR-style `nfl_standings_calc`) · `nfl_projection` · `nfl_usage_projection` · `nfl_player_props` · `nfl_market` · `nfl_draft_model` · `nfl_availability` · `nfl_line_grades` · `nfl_roster_builder` · `nfl_ngs_tracking`.

SPRING FOOTBALL · UFL / XFL / CFL / AAF

```
from sportsdataverse.football import *
from sportsdataverse.football.ufl import
espn_ufl_scoreboard, espn_ufl_summary
from
sportsdataverse.football.spring_football_ep_wp
import *
pbp =
build_spring_football_pbp(espn_ufl_summary(game_id),
league="ufl")
enrich_spring_football_pbp(pbp)   # EP/WP via
the NFL ep_wp engine
```

Leagues: `football.ufl` · `.xfl` · `.cfl` · `.aaf` — full ESPN short-name surface each (`espn_ufl_scoreboard`, `espn_ufl_team_roster`, `espn_cfl_standings`, …).

NBA · G-LEAGUE · SPORTSDATAVERSE.NBA

ESPN

```
from sportsdataverse.nba import *
espn_nba_schedule(dates=2025);
espn_nba_teams()
espn_nba_pbp(game_id=401584793)   # plays,
box, rosters, WP
espn_nba_game_rosters(401584793);
espn_nba_team_roster(13)
espn_nba_player_stats(athlete_id=3975,
season=2025)
```

STATS.NBA.COM (128 WRAPPERS)

One stem, one host; `league_id` picks **"00"** NBA · **"20"** G-League · **"15"** Summer League. Uses `curl_cffi impersonate="chrome"` (plain `requests` stalls on the TLS fingerprint block).

```
from sportsdataverse.nba import nba_stats
nba_stats.nba_stats_leaguedashplayerstats(league_id
nba_stats.nba_stats_playercareerstats(player_id="25
# dict of frames
nba_stats.nba_stats_playbyplayv3(game_id="0022400001")
nba_stats.nba_stats_boxscoretraditionalv3(game_id=...)
# return_parsed=False → raw resultSets envelope
parse_nba_stats_result_sets(raw,
result_set=None)
```

Families: `leaguedash*` · `playerdash*` · `teamdash*` · `boxscore*v3` · `playbyplayv3` · `shotchartdetail` · `synergyplaytypes` · `leaguegamefinder` · `gamerotation` · `video*` · `draftcombine*` · `commonallplayers` · `scoreboardv3`.

Raw JSON read-through store: `SDV_PY_NBA_RAW_JSON_DIR` (per-endpoint `_DIR_{ENDPOINT}`), `_READONLY=1` = fully offline. Season dirs use the END-year convention (2024 = 2023-24).

NBA RELEASE LOADERS

ESPN-derived: `load_nba_pbp schedule team_boxscore player_boxscore rosters game_rosters officials standings player_core player_season_stats team_season_stats shots draft player_impact`. stats.nba.com-derived: `load_nba_stats_pbp(_v3) possessions(_v3) lineups(_v3) shots schedules rosters game_rosters game_lineups player_boxscores team_boxscores player_game_logs player_season_stats team_season_stats standings officials coaches`. Crosswalks: `load_nba_player_crosswalk team_crosswalk schedule_crosswalk`.

POSSESSION ENGINE (PBPSTATS-STYLE)

```
from sportsdataverse.nba.nba_possessions
import nba_possessions
poss = nba_possessions("0022400001",
league_id="00")
compile_nba_season(2025, season_type="Regular
Season")
from sportsdataverse.nba.nba_rapm import
nba_rapm, nba_rapm_from_games
nba_rapm(poss)                  # ridge, CV
over alphas
nba_adj_rapm(poss, prior)       # Bayesian
prior-shrunk
nba_la_rapm(...) nba_decay_rapm(...)
nba_four_factor_rapm(...)
nba_matchup_drapm("2024-25")    # playtype-
matchup dRAPM
```

On-court: `nba_lineups.nba_on_court` · `nba_enhanced_pbp` · `nba_play_context(game_id, transition_seconds=6.0)` (CTG-style start types) · `nba_playtype.nba_playtype_ratings` · `nba_shot_zones`. G-League: `nbagl_possessions nbagl_enhanced_pbp nbagl_on_court nbagl_rapm_from_games`.

NBA PLAYER & TEAM VALUE

```
nba_shot_value(player_ids, season="2024-25")
# xPts, shooter talent
nba_bpm(player_logs, team_logs, positions)
nba_spm(box_features, coefficients)
nba_war(ratings, poss, replacement_level=...,
pts_per_win=...)
nba_darko(panel, ages)
# Kalman DPM
nba_team_ratings(seasons=2025, as_of_date=...)
nba_game_predict.nba_predict_games(games,
ratings)
nba_game_predict.nba_in_game_win_prob(pbp,
0.58)
```

Also `nba_expected_turnovers` · `nba_foul_drawing` · `nba_clutch` · `nba_tracking_value` · `nba_player_props` · `nba_draft_model` · `nba_rookie_projection` · `nba_aging_curve` · `nba_availability`.

Bundled models

`nba_in_game_wp.ubj` · `nba_aging_curve` · `nba_draft_value` · `nba_rookie_projection` · `nba_availability` (+ `wnba_*` twins).

External oracles

`load_darko_dpm(path)` · `load_epm` · `load_lebron_daily/_season` · `load_rapm_ryan_davis` · `load_dunks_threes_stats` · `load_draft_outcomes` (`nba_oracle_data`).

WNBA · SPORTSDATAVERSE.WNBA

SAME SHAPE, WNBA HOST

```
from sportsdataverse.wnba import *
espn_wnba_schedule(dates=2025);
espn_wnba_pbp(game_id)
espn_wnba_team_roster(team_id=5);
espn_wnba_player_stats(...)
from sportsdataverse.wnba import wnba_stats
# 111 wrappers, LeagueID=10
wnba_stats.wnba_stats_leaguedashplayerstats()
wnba_stats.wnba_stats_playbyplayv3(game_id="1022500
```

Engine: `wnba_engine.wnba_possessions` · `wnba_enhanced_pbp` · `wnba_on_court` · `wnba_play_context`. Models: `wnba_team_ratings` · `wnba_game_predict.wnba_predict_games` · `wnba_in_game_win_prob` · `wnba_shot_value` · `wnba_playtype_impact` · `wnba_tracking_value` · `wnba_draft_model` · `wnba_rookie_projection` · `wnba_aging_curve` · `wnba_career_trajectory` · `wnba_availability` · `wnba_clutch` · `wnba_player_props`.

Loaders: `load_wnba_pbp schedule team_boxscore player_boxscore rosters game_rosters officials standings player_core player_season_stats team_season_stats shots draft player_impact` + `load_wnba_stats_*` (pbp possessions lineups shots schedules rosters game_rosters game_lineups player/team boxscores player_game_logs player/team season_stats standings officials coaches draft) + crosswalks.

CROSSWALKS (LEAGUE ↔ COLLEGE)

WNBA↔NBA and WBB↔MBB identity crosswalks: `load_wnba_player_crosswalk` · `load_mbb_player_crosswalk` · `load_wbb_team_crosswalk` · `load_wbb_schedule_crosswalk` · `_common_crosswalk_basketball`. ESPN↔stats.ncaa.org: `mbb_ncaa_espn_crosswalk`.

COLLEGE · SPORTSDATAVERSE.MBB / .WBB

ESPN + STATS.NCAA.ORG

```
from sportsdataverse.mbb import *
espn_mbb_schedule(dates=2025);
espn_mbb_pbp(game_id)
espn_mbb_teams();
espn_mbb_game_rosters(game_id)
# bigballR / wbigballR port — 16 fns per
league
ncaa_mbb_team_schedule(team="Duke",
season="2024-25")
pbp = ncaa_mbb_game_pbp(game_id)
ncaa_mbb_lineups(pbp);
ncaa_mbb_possessions(pbp)
ncaa_mbb_on_off(...);
ncaa_mbb_player_combos(...)
ncaa_mbb_shot_locations(...);
ncaa_mbb_box_scores(...)
```

Also `ncaa_mbb_date_games team_ids team_roster team_stats player_stats player_lineups play_by_play join_pbp_shots`; `ncaa_wbb_*` mirrors (WBB is halves before 2016). Engine: `mbb_ncaa_*` (fetch → pbp → lineups → stints → possessions → strength, with stint self-healing); names via `display_name_to_roster_key`. Torvik: `torvik_ratings(year)`, `torvik_team_factors`; WBB: `bart_wbb_ratings`.

COLLEGE MODELS & PROJECTIONS

```
r = mbb_team_ratings(seasons=2025)        #
league="womens" for WBB
mbb_predict_games(games, r); mbb_season_sim(r,
remaining)
mbb_bracketology(2025); mbb_bracket_sim(field,
r)
mbb_in_game_win_prob(pbp,
pregame_home_prob=0.6)
mbb_box_bpm(2025); mbb_archetypes(2025)
mbb_shot_quality(shots);
mbb_shooter_talent(scored)
mbb_strength_of_schedule([2025])
mbb_transfer_projection(2025);
mbb_recruiting_projection(2025)
mbb_draft_projection(2025)
```

Plus `mbb_rapm` · `solve_rapm_league` (D-I-wide, released as `ncaa_mbb/wbb_rapm`) · `mbb_lineup_stats` · `mbb_luck` · `mbb_positions` · `mbb_shot_selection` · `wbb_*` twins. Bundled: `mbb/wbb_in_game_wp.ubj`, `_box_bpm`, `_archetypes`, `_draft`, `_recruiting`, `_transfer` (.json).

Loaders

`load_mbb_pbp schedule team_boxscore player_boxscore rosters game_rosters officials standings player_core player_season_stats team_season_stats shots ratings player_value` + crosswalks; `load_wbb_*` identical. WP enriched in place in the released pbp.

Hockey · Baseball · Soccer · Cricket · Odds :: and the wider SportsDataverse

R siblings: fastRhockey · baseballr · softballR · oddsapiR · sportsdataversedata

HOCKEY · NHL · PWHL · HOCKEY.*

NHL — FIVE LIVE APIS

```
from sportsdataverse.nhl import *
nhl_web_schedule("2025-01-15")
# api-web.nhle.com
plays = nhl_web_pbp(2024020500)
# parsed by default
nhl_edge_skater_detail(8478402,
season=20242025)  # EDGE tracking
nhl_stats_rest_skater_report(...)
# api.nhle.com/stats/rest
nhl_records_awards()
# records.nhl.com
espn_nhl_pbp(game_id=401559000);
espn_nhl_schedule(dates=2025)
```

Parsers: 19 api-web (`parse_nhl_web_pbp` …), 4 EDGE families, generic stats-REST / records. Loaders: `load_nhl_pbp schedule games rosters player_box team_box goalie_box skater_box shifts scoring penalties shootout linescore officials three_stars scratches shots_by_period`.

NHL analytics

```
nhl_xg(pbp)                          # shot xG
(bundled model)
nhl_expected_assists(pbp);
nhl_faceoff_value(pbp)
nhl_gsax.nhl_goalie_gsax(pbp, shifts)
nhl_rapm.nhl_skater_rapm(pbp, shifts);
nhl_war.nhl_skater_war(...)
nhl_unit_ratings(pbp, shifts);
nhl_team_ratings(seasons=2025)
nhl_in_game_win_prob(pbp,
pregame_home_prob=0.55)
```

Also `nhl_penalty_value` · `nhl_zone_transitions` · `nhl_special_teams` · `nhl_edge_value` · `nhl_player_props`.

PWHL + 19 HOCKEYTECH LEAGUES

```
import sportsdataverse as sdv
sdv.pwhl_schedule(season=2025);
sdv.pwhl_pbp(game_id)
sdv.pwhl_game_shifts(game_id);
sdv.pwhl_game_corsi(game_id)
sdv.echl_schedule(season=2026);
sdv.ushl_standings(season=2025)
```

One registry-driven core (`hockeytech.LEAGUES`): 13 callables per league — `<lg>_schedule _pbp _standings _teams _team_roster _player_stats _leaders _game_summary _game_shifts _player_toi _game_corsi _season_id` + `most_recent_<lg>_season`. Leagues: `pwhl ahl ohl whl qmjhl echl sphl chl ushl bchl ajhl sjhl ojhl cchl gojhl mhl nojhl vijhl kijhl mjhl`. PWHL extras: `pwhl_goalie_gsax` · `pwhl_skater_rapm` · `pwhl_in_game_win_prob` · `pwhl_team_ratings`. College hockey: `espn_mch_*` / `espn_wch_*`, `college_hockey_ratings`.

BASEBALL · MLB · BASEBALL

MLB STATS API + ESPN

```
from sportsdataverse.mlb import *
mlb_schedule(date="2025-07-04")          #
statsapi.mlb.com
mlb_boxscore(game_pk=745000);
mlb_play_by_play(game_pk)
espn_mlb_pbp(game_id=401570000);
espn_mlb_schedule(dates=2025)
```

~80 Stats-API wrappers (`mlb_*`). `load_mlb_*`: `batter_projection catcher_framing command_plus expected_hr expected_stats oaa re24_matrix stuff_plus we_table wpa xera`.

STATCAST — BASEBALL SAVANT (43 ENDPOINTS)

```
mlb_statcast_search("2025-04-01", "2025-04-
07",
                    player_type="pitcher",
pitch_type="FF")
mlb_statcast_leaderboard_expected_stats(year=2025)
# 37 boards
mlb_statcast_gamefeed(game_pk=745000)
mlb_statcast_player(660271,
section="statcast")
```

Families `search` (chunked; `_minors`, `_wbc`) · `leaderboard` · `gamefeed` · `player`; friendly filters (`season pitch_type at_bat_result`) translate to Savant params.

MLB ANALYTICS

```
mlb_stuff_plus(pitches);
mlb_command_plus(pitches)
mlb_expected_stats("2025-04-01", "2025-09-30")
mlb_catcher_framing(pitches);
mlb_fielding_oaa(bip)
mlb_run_expectancy_matrix(seasons=2024);
mlb_win_expectancy(pbp, results)
mlb_pitch_classify(pitches);
mlb_batter_projection(2026)
```

Also `mlb_expected_home_runs` · `mlb_run_values` · `mlb_swing_decision` · `mlb_pitch_sequencing` · `mlb_pitch_fatigue` · `mlb_stolen_base` · `mlb_baserunning` · `mlb_umpire_zone` · `mlb_team_projection` · `mlb_prop_projection`.

COLLEGE BASEBALL & SOFTBALL

```
from sportsdataverse.baseball.college_baseball
import *
espn_college_baseball_scoreboard(dates=2025)
parse_college_baseball_ncaa_pbp(html)    #
stats.ncaa.org
college_baseball_re24(...);
college_baseball_wpa(...)  # + softball
```

SOCCER · CRICKET · ODDS · WIN-EXPECTANCY

SOCCER (ESPN, 12 LEAGUES)

```
from sportsdataverse.soccer.epl import
espn_epl_scoreboard
espn_epl_scoreboard(dates=20250315)
espn_soccer_team_schedule("eng.1",
team_id=359, season=2025)
```

Per-league modules `epl mls nwsl laliga bundesliga seriea ligue1 ligamx ucl uel wc wwc` (full ESPN surface each) + league-parameterized `espn_soccer_*`.

CRICKET (ESPN + BUNDLED WP)

```
from sportsdataverse.cricket import *
espn_cricket_scoreboard(league="icc",
dates=2025)
state = cricket_match_state(pbp)
cricket_win_probability(state);
cricket_wpa(state_wp)
```

ODDS — THE ODDS API (`ODDS_API_KEY`)

```
from sportsdataverse.odds import *
toa_sports(); toa_usage()
toa_sports_odds("basketball_nba",
regions="us", markets="h2h,spreads")
toa_event_odds(sport, event_id,
markets="player_points")
toa_sports_odds_history(sport, date="2024-11-
29T22:45:00Z")
toa_sports_scores(...); toa_sports_events(...)
```

Historical lines also ride the release loaders (`load_cfb_betting_lines`, `load_nfl_schedule`).

WIN-EXPECTANCY LAB (`WEXP`)

```
from sportsdataverse import wexp
oracle = wexp.build_cfb_market_oracle(...)  #
or _nfl_
pred =
wexp.build_predictor(wexp.VariantConfig(...))
wexp.backtest.run_backtest(oracle, pred,
model_id="ridge")
```

Ridge-margin, Elo, Glickman–Stern predictors vs market closing lines; Brier + calibration scoring. Football-only today.

THE WIDER SPORTSDATAVERSE

R PACKAGES (SPORTSDATAVERSE.ORG)

Package	Domain	sdv-py mirror
cfbfastR	college football	`cfb` · cfbfastR-data loaders
hoopR	NBA · men's college	`nba mbb`
wehoop	WNBA · women's college	`wnba wbb`
baseballr	MLB · Statcast · college	`mlb baseball`
fastRhockey	NHL · PWHL · HockeyTech	`nhl pwhl hockey`
softballR	college softball	`baseball.college_softball`
recruitR	CFB recruiting	`cfb.sports247_* on3_*`
oddsapiR	The Odds API	`odds`
nflseedR · cfbseedR	season sims / seeding	`nfl_simulations cfb_simulations`
sportsdataversedata	release publishing	`release`

Related: nflfastR / nflreadr (nflverse) · cfbplotR · sportyR / sportypy · sportsdataverse-js.

DATA REPOS BEHIND THE LOADERS

Each sport has a `*-raw` (scraped JSON, committed per game) and a `*-data` (tidy parquet on GitHub releases) repo under `github.com/sportsdataverse`. Loaders read the release asset; `sportsdataverse.release` writes them.

```
sportsdataverse_save(df, "pbp_2025", "cfb",
release_tag="cfb_pbp")
```

CONVENTIONS TO REMEMBER

- polars 1.x API everywhere; `return_as_pandas=True` converts at the edge.
- IDs are join keys — one dtype per id; never paper over with a float→str cast.
- Live tests gate on `SDV_PY_LIVE_TESTS=1`; stats.nba.com on `SDV_PY_NBA_STATS_LIVE=1`.
- Reference docs + returns tables are generated from endpoint YAML (`tools/codegen`).

Models :: the cross-sport modeling grid, bundled artifacts, validation gates

Every row below is code that ships in the wheel — no notebooks, no downloads (except NFL xYAC).

ONE SPINE, EVERY SPORT

Each league implements the same five layers. Names are the public entry points; `·` separates alternatives, — means not built for that league.

League	Team ratings	Game predict	Season / bracket sim	In-game WP	Player value
CFB	`cfb_ratings`	`cfb_predict_games`	`cfb_simulations` · `cfb_season_odds` · `cfb_playoff_seeds`	`wp_spread` / `wp_naive` via `CFBPlayProcess`	`cfb_adjusted_epa` · `cfb_roster_talent` · `cfb_draft_projection`
NFL	`nfl_ratings`	`nfl_market` · `nfl_gamescript`	`nfl_simulations` (+ `nfl_standings_calc`)	`enrich_nfl_pbp` → `wp`, `vegas_wp`	`nfl_projection` · `nfl_kicker_rating` · `nfl_line_grades`
NBA / G-Lg	`nba_team_ratings`	`nba_game_predict.nba_predict_games`	— (schedule-driven via predict)	`nba_in_game_win_prob`	`nba_rapm` · `nba_adj_rapm` · `nba_bpm` · `nba_spm` · `nba_war` · `nba_darko` · `nba_shot_value`
WNBA	`wnba_team_ratings`	`wnba_game_predict.wnba_predict_games`	—	`wnba_in_game_win_prob`	`wnba_engine` RAPM · `wnba_shot_value` · `wnba_playtype_impact`
MBB / WBB	`mbb_team_ratings` · `torvik_ratings`	`mbb_predict_games`	`mbb_season_sim` · `mbb_bracketology` · `mbb_bracket_sim`	`mbb_in_game_win_prob`	`mbb_box_bpm` · `mbb_rapm` · `mbb_shot_quality` · `mbb_archetypes` (+ `wbb_*`)
NHL	`nhl_team_ratings`	`nhl_market.nhl_predict_games`	—	`nhl_in_game_win_prob`	`nhl_xg` · `nhl_skater_rapm` · `nhl_skater_war` · `nhl_goalie_gsax` · `nhl_unit_ratings`
PWHL	`pwhl_team_ratings`	`pwhl_market.pwhl_predict_games`	—	`pwhl_in_game_win_prob`	`pwhl_skater_rapm` · `pwhl_skater_war` · `pwhl_goalie_gsax`
MLB	`mlb_team_projection`	— (projection-driven)	—	`mlb_win_expectancy`	`mlb_stuff_plus` · `mlb_command_plus` · `mlb_expected_stats` · `mlb_catcher_framing` · `mlb_fielding_oaa`
Others	college hockey `college_hockey_ratings` · cricket `cricket_win_probability` / `cricket_wpa` · spring football reuses the NFL engine (`build_spring_football_pbp` → `enrich_spring_football_pbp`)				

Shared shape: ratings take `seasons=` + `as_of_date=` (leak-safe, exclusive cut); predictors take `(games, ratings)`; sims take `(ratings | games, …, n_sims / simulations)`; in-game WP takes `(pbp, pregame_home_prob)`.

DRAFT, PROJECTION & ROSTER MODELS

Family	Entry points
Draft value / rookies	`nba_draft_model` · `nba_rookie_projection` · `wnba_draft_model` · `nfl_draft_model` · `mbb_draft_projection` · `cfb_draft_projection`
Aging & availability	`nba_aging_curve` · `nba_availability` · `wnba_career_trajectory` · `nfl_availability`
Recruiting / transfers	`cfb_recruiting_projection` · `cfb_transfer_impact` · `mbb_recruiting_projection` · `mbb_transfer_projection`
Props & markets	`nba_player_props` · `wnba_player_props` · `nfl_player_props` · `nhl_player_props` · `pwhl_player_props` · `mlb_prop_projection` · `nfl_market` · `nhl_market`

BUNDLED ARTIFACT REGISTRY

Shipped as package data (`importlib.resources`); XGBoost `.ubj` unless noted. Feature counts from the model cards.

Dir	Artifacts
`cfb/models`	`ep_model` 8f · `wp_naive` 12f · `wp_spread` 13f · `cfb_cp_model` · `qbr_model` 10f · `fg_model` 5f · `fd_model` · `two_pt_model` 4f · `xpass_model` 7f · `punt_distribution.parquet` · `cfb_field_position_ep.parquet`
`nfl/models`	`ep_model` 18f · `wp_spread` 12f · `wp_naive` 11f · `cp_model` 18f · `fg_model` · `qbr_model` · `two_pt_model` · `xpass_model` · `nfl_playcall` · `punt_data.parquet`; xYAC (76-class) fetched on first use
`nba/models`	`nba_in_game_wp` · `wnba_in_game_wp` · `{nba,wnba}_aging_curve` · `_draft_value` · `_rookie_projection` · `_availability` (.json)
`mbb/models`	`{mbb,wbb}_in_game_wp` · `_box_bpm` · `_archetypes` · `_draft` · `_recruiting` · `_transfer` (.json)
`nhl/models`	`nhl_in_game_wp.json` (+ xG coefficients in `nhl_xg`)
`mlb/models`	`mlb_stuff_plus` · `mlb_command_plus`
`cricket/models`	`cricket_resource_surface.parquet` · `cricket_winprob_calibration.parquet`

Every model's fitting script and era cuts live in the `*-data` producer repo that trains it (cfbfastR-cfb-data, nfl-data, hoopR-nba-data …); constants in `*_constants.py` / `model_vars.py` cite them.

RULE-ERA & FEATURE CONVENTIONS

- NFL era cuts `ERA_SEASON_CUTS` = 2001 / 2005 / 2013 / 2017; CFB models are per rule era.
- Kickoff / PAT feature substitution: touchback yardline 80 pre-2016, 75 from 2016; `down→1`, `ydstogo→10`.
- Spread WP: `spread_time` decays with exponent −4.0 (`SPREAD_TIME_DECAY_EXPONENT`).
- Outputs are `Float64`; models emit float32 — cast at the boundary.

VALIDATION & ORACLE GATES

Every model ships with a gate against an external oracle captured as a committed fixture. Gates are measured from observed values and never lowered.

Metric	Used for
Brier · log-loss · calibration table	win probabilities, in-game WP, sims (`brier_score`, `calibration_table` in `*_prediction_constants` / `wexp`)
MAE vs market	spreads / totals (`nfl_market`, `cfb_season_odds`, wexp market oracles)
Spearman ρ	ratings vs Torvik / KenPom / ESPN FPI / MoneyPuck; player value vs EPM, DARKO, RAPM, LEBRON
Correlation floors	parity with the R producers (nflfastR EP 0.996, CFB ratings ρ 0.97)

Oracle loaders

`load_darko_dpm` · `load_epm` · `load_lebron_daily` · `load_rapm_ryan_davis` · `load_dunks_threes_stats` · `load_draft_outcomes` · `torvik_ratings` · `bart_wbb_ratings` · `load_cfb_fpi_weekly` · `load_cfb_power_index` · `wexp.build_cfb_market_oracle` / `_nfl_`.

LEAK-SAFETY CONTRACT

- `as_of_date` / `through_week` are **exclusive**: the cut game is never in the training window.
- Every lag, cumulative sum or shift is `.over("game_id")` (or season) — no cross-game bleed when frames are concatenated.
- Ratings for a date use only games with `game_date < as_of_date`; sims draw from those ratings.
- Fitted constants (HFA, σ, λ) name the fitting script; assert the **output** changed, not that the code ran.

WIN-EXPECTANCY LAB (WEXP)

```
from sportsdataverse import wexp
o  = wexp.build_cfb_market_oracle(...)
p  =
wexp.build_predictor(wexp.VariantConfig(...))
wexp.backtest.run_backtest(o, p,
model_id="ridge")
```

Ridge-margin, Elo, Glickman–Stern vs closing lines; vintage store for reproducible walk-forward runs.

Recipes :: end-to-end pipelines, polars idioms, gotchas

Copy-paste starting points — each runs as written given the named inputs.

CFB · RAW JSON → ENRICHED PBP (OFFLINE)

```
from sportsdataverse.cfb import CFBPlayProcess
proc = CFBPlayProcess(gameId=401628334,
    path_to_json="raw/2024/",
    odds_override={"gameSpread": -6.5,
"overUnder": 55.5,
                   "homeFavorite": True,
                   "gameSpreadAvailable":
True})
proc.cfb_pbp_disk()
pbp = proc.run_processing_pipeline()   #
EPA/WPA, no network
box = proc.create_box_score()
```

The override injects persisted odds so the rebuild never falls back to the live odds endpoint (provenance: `pbp_txt["odds_source"] == "injected"`).

NFL · LOAD → ENRICH → 4TH-DOWN TABLE

```
import sportsdataverse.nfl as nfl
from sportsdataverse.nfl.ep_wp import
enrich_nfl_pbp
from sportsdataverse.nfl.nfl_fourth_down
import get_4th_down_probs
pbp = nfl.load_pbp([2024])
df  = enrich_nfl_pbp(pbp)          #
ep→epa→wp→wpa→cp→xyac
fourth = get_4th_down_probs(
    df.filter(pl.col("down") == 4))
```

`get_4th_down_probs` adds go / FG / punt WP + the recommendation columns (nfl4th-style, fully offline).

SPRING FOOTBALL ON THE NFL ENGINE

```
from sportsdataverse.football.ufl import
espn_ufl_summary
from
sportsdataverse.football.spring_football_ep_wp
import *
raw = espn_ufl_summary(game_id)
pbp = build_spring_football_pbp(raw,
league="ufl")
pbp = enrich_spring_football_pbp(pbp)
```

NBA · POSSESSIONS → RAPM → WAR

```
from sportsdataverse.nba.nba_season_compile
import compile_nba_season
from sportsdataverse.nba.nba_rapm import
nba_rapm_from_games
from sportsdataverse.nba import nba_war
poss  = compile_nba_season(2025)   # end-year:
2024-25
rapm  =
nba_rapm_from_games(poss["game_id"].unique())
value = nba_war(rapm, poss,
                replacement_level=-2.0,
pts_per_win=30.0)
```

Set `SDV_PY_NBA_RAW_JSON_DIR` first and the compile reads disk before network; add `_READONLY=1` for a fully offline, reproducible build (a miss raises `RawStoreMissError`).

MBB · STATS.NCAA.ORG PBP → LINEUPS → ON/OFF

```
from sportsdataverse.mbb import
(ncaa_mbb_game_pbp,
    ncaa_mbb_lineups, ncaa_mbb_possessions,
ncaa_mbb_on_off)
pbp  = ncaa_mbb_game_pbp(6305900)
lu   = ncaa_mbb_lineups(pbp)
poss = ncaa_mbb_possessions(pbp)
oo   = ncaa_mbb_on_off("PlayerName", lu)
```

WBB mirrors every call (`ncaa_wbb_*`); pre-2016 WBB pbp is halves, not quarters — 0 rows for a quarters filter is data, not an error.

MLB · STATCAST SEARCH → STUFF+

```
from sportsdataverse.mlb import
mlb_statcast_search, mlb_stuff_plus
pitches = mlb_statcast_search("2025-04-01",
"2025-04-30",
    player_type="pitcher", pitch_type="FF")
stuff = mlb_stuff_plus(pitches, level="pitch")
```

POLARS IDIOMS FOR PBP WORK

```
# EPA per drive, success rate — group, don't
loop
pbp.group_by("game_id", "drive_id",
maintain_order=True) \
   .agg(pl.col("epa").sum(), (pl.col("epa") >
0).mean()
        .alias("success_rate"))

# lags NEVER cross a game boundary
pbp.with_columns(pl.col("score_diff").shift(1)

.over("game_id").alias("prev_diff"))

# explicit bool masks (house style)
pbp.filter((pl.col("pass") == True) &
           (pl.col("garbage_time") == False))

# no lookaround in Rust regex — case-toggle
instead
pl.col("text").str.extract(
    r"(?i)sacked by(?-i: ([A-Z][\w'\.\-]+))",
1)

# pandas only at the very edge
df = any_loader(..., return_as_pandas=True)
```

CROSS-LEAGUE JOINS

```
# pin ONE dtype per id, assert before joining
a =
a.with_columns(pl.col("athlete_id").cast(pl.Int64))
assert a.schema["athlete_id"] ==
b.schema["athlete_id"]
a.join(b, on="athlete_id", how="left")
# ESPN ⇔ stats.nba.com identity
xwalk = sdv.load_nba_player_crosswalk(seasons=
[2025])
```

Never stringify a float id (`"123.0"` ≠ `"123"`); if you must go Utf8, cast through Int64 first.

GOTCHA STRIP

Trap	Rule
Season year	NBA/WNBA-stats dirs use the **end year** (2024 = 2023-24); ESPN loaders use kickoff/tip year; NBA span params want `"2023-24"`
stats.nba.com	TLS-fingerprint block → silent hang, not an error; the runtime uses `curl_cffi impersonate="chrome"`; run live only from residential IPs
Game ids	stats.nba.com `game_id` is a zero-padded **string** (`"0022400001"`) — never int-cast it
WBB history	halves before 2016 — quarter-keyed logic silently empties six seasons
Cache	`@cached_loader` keys on (name, args) not URL — `clear_cache()` after changing loader targets
Empty ≠ error	parsers return zero-row frames with the documented schema; check `df.is_empty()`, don't try/except
ESPN rate	Core v2 403s under aggressive parallelism — keep scrapes low-concurrency

TEST GATES

```
uv run pytest                          # offline
suite
SDV_PY_LIVE_TESTS=1 uv run pytest      # live
APIs
SDV_PY_NBA_STATS_LIVE=1 ...            #
stats.nba.com only,
                                       #
residential IP
```

CONTRIBUTING A NEW ENDPOINT

- Wrapper metadata lives in `tools/codegen/endpoints/*.yaml`; run `uv run python tools/codegen/generate.py`, never hand-edit generated files.
- Returns-table descriptions go in `manual_column_descriptions.yaml` (schemas/** is clobbered on re-capture).
- Validate parsers against **real captured fixtures**, never synthetic ones.
- New modules are fully typed and join the mypy ratchet in `pyproject.toml`.
- `generate.py --check` is the CI drift gate — regenerate before every push.

sportsdataverse-py :: CHEAT SHEET

The Python sister to the SportsDataverse R packages (wehoop, hoopR, cfbfastR, baseballr, fastRhockey, nflfastR). One `pip install` gives polars-first, tidy access to play-by-play, box scores, schedules, rosters and odds across 15+ sports, plus bundled EP / WP / rating / projection models and release-dataset loaders. Page 1 of 4: getting started, coverage map, the ESPN cross-league layer, utilities.

v0.0.75

Python 3.9 – 3.14 · polars 1.x
py.sportsdataverse.org

GET STARTED

```
pip install sportsdataverse
pip install "sportsdataverse[all]"  # + models, curl_cffi, tests
uv add sportsdataverse

import sportsdataverse as sdv
from sportsdataverse.cfb import load_cfb_pbp
```

Every public callable is also on the flat `sdv.*` namespace (4,600+ names). Sub-packages: `cfb nfl nba nbagl wnba mbb wbb nhl pwhl hockey hockeytech mlb baseball soccer cricket football odds wexp`.

RETURN CONVENTIONS

Flag	Effect
default	`polars.DataFrame`, snake_case columns
`return_as_pandas=True`	pandas frame (same data)
`raw=True`	scrapers: untouched JSON `dict`
`return_parsed=False`	ESPN wrappers: tidy frame (default) → raw `dict`
multi-table	`dict[str, pl.DataFrame]` keyed by section
empty / malformed	zero-row frame with documented schema — never raises

THREE SURFACES

1 · Release-dataset loaders

`load_<sport>_<dataset>(seasons, return_as_pandas=False)` reads parquet from the SportsDataverse GitHub releases (the `*-data` repos). Cached; zero scraping.

```
pbp = load_cfb_pbp(seasons=[2023, 2024])
sch = sdv.load_nba_schedule(seasons=range(2020, 2025))
```

2 · Live API wrappers

`espn_<league>_*`, `nba_stats_*`, `wnba_stats_*`, `nhl_web_*`, `nhl_edge_*`, `mlb_*`, `mlb_statcast_*`, `<hockeytech-league>_*`, `ncaa_mbb_*`, `toa_*` (odds).

3 · Processing + models

PBP processors (`CFBPlayProcess`, `NFLPlayProcess`), `enrich_nfl_pbp`, `nba_possessions`, ratings, simulations, projections — see pages 2–4.

COVERAGE MAP

Sport	Sub-package	Live sources	R sibling
NFL	`nfl`	ESPN · api.nfl.com · PFF	nflfastR · nflreadr
CFB	`cfb`	ESPN · stats.ncaa.org · Fox · Yahoo · 247 · On3 · PFF	cfbfastR · recruitR
NBA · G-League · Summer	`nba` `nbagl`	ESPN · stats.nba.com · Fox	hoopR
WNBA	`wnba`	ESPN · stats.wnba.com	wehoop
Men's college bb	`mbb`	ESPN · stats.ncaa.org · Torvik	hoopR · bigballR
Women's college bb	`wbb`	ESPN · stats.ncaa.org · Bart	wehoop · wbigballR
NHL	`nhl`	ESPN · api-web.nhle.com · EDGE · stats REST · records	fastRhockey
PWHL + 19 minor/junior	`pwhl` `hockey.*`	HockeyTech / LeagueStat	fastRhockey
College hockey M/W	`hockey.mch` `.wch`	ESPN	—
MLB	`mlb`	ESPN · statsapi.mlb.com · Baseball Savant	baseballr
College baseball / softball	`baseball`	ESPN · stats.ncaa.org	baseballr · softballR
Soccer (12 leagues)	`soccer.*`	ESPN	—
Cricket	`cricket`	ESPN + bundled WP	—
UFL · XFL · CFL · AAF	`football.*`	ESPN + spring EP/WP	—
Odds & lines	`odds`	The Odds API	oddsapiR
Win-expectancy lab	`wexp`	CFB / NFL market oracles	—

Release data repos: `cfbfastR-data`, `nfl-data`, `hoopR-nba-data`, `hoopR-mbb-data`, `wehoop-wnba-data`, `wehoop-wbb-data`, `fastRhockey-*-data`, `baseballr-data`, `ncaa-mbb/wbb-data`, `*-stats-data`.

ESPN CROSS-LEAGUE LAYER

One core (`_common_espn`) × N thin league modules. Every league gets the same **121 endpoint short names** as `espn_<prefix>_<short>` (≈125 wrappers per league; 800+ total across nba wnba nfl cfb mbb wbb nhl mlb, plus soccer, cricket, college baseball/hockey and spring football).

```
from sportsdataverse.nba import espn_nba_team_roster
df  = espn_nba_team_roster(team_id=13)
# polars (default)
raw = espn_nba_team_roster(team_id=13,
                           return_parsed=False)
# raw dict
```

Family	Short names
Schedule	`schedule calendar scoreboard games season_weeks season_week_games`
Teams	`teams team team_roster team_schedule team_record team_injuries team_depthcharts franchise venue`
Games	`summary game game_rosters game_plays play_participants game_odds game_officials game_predictor game_probabilities game_propbets`
Players	`player_stats player_gamelog player_splits player_career_stats player_bio player_contracts player_vs_player`
League	`standings leaders news injuries transactions draft awards conferences futures season_qbr`
NCAA extras	`rankings fpi recruits recruiting_rankings` (mbb wbb cfb …)

Parser layer

`parse_summary(payload, section=None)` → 21 sub-frames: `boxscore_player boxscore_team plays winprobability leaders game_info officials header season_series against_the_spread standings broadcasts format pickcenter odds article injuries news drives drive_plays scoring_plays`. Generic fall-throughs: `parse_single_entity`, `parse_items`.

Full PBP processors (tidy, per game)

```
espn_nba_pbp(401584793)    espn_wnba_pbp(...)
espn_mbb_pbp(...)          espn_wbb_pbp(...)
espn_nhl_pbp(...)          espn_mlb_pbp(...)
CFBPlayProcess(gameId=401628334)   # page 2
NFLPlayProcess(gameId=401671889)   # page 2
```

DISCOVERY

```
sdv.find_team("Duke", "mbb")
sdv.find_athlete("Caitlin Clark", "wnba", team="Fever")
sdv.find_event("2024-11-30", "cfb", home="Ohio State")
sdv.function_count()            # per league
sdv.list_functions("nhl", search="pbp")
```

CLI mirrors it: `sdv find-team Duke --league mbb`, `sdv function-count`, `sdv cache stats|clear|mode`.

CACHE

```
sdv.set_cache_mode("filesystem")  # memory | filesystem | off
sdv.set_default_ttl(3600)
sdv.cache_stats();  sdv.clear_cache()
# NFL has its own nflreadpy-style config
from sportsdataverse.nfl import update_config
update_config(cache_mode="filesystem",
cache_duration=3600)
```

Env: `SDV_PY_NFL_CACHE` · `SDV_PY_NFL_CACHE_DIR` · `SDV_PY_NFL_CACHE_DURATION`.

HTTP, TIDY & RELEASE HELPERS

`dl_utils.download(url)` — single retrying HTTP chokepoint. `underscore() camelize() kebabize()`, `flatten_json_iterative()`, `key_check()`.

`sportsdataverse.release` — port of the sportsdataversedata R package: `sportsdataverse_upload()`, `sportsdataverse_save()` (parquet / rds / csv.gz), `gh_cli_release_upload()`, `gh_cli_release_assets()`, `gh_cli_release_tags()`.

Errors: `NoESPNDataError` · `SeasonNotFoundError` · `RawStoreMissError`.

ENV VARS WORTH KNOWING

`SDV_PY_LIVE_TESTS=1`	run gated live tests
`SDV_PY_NBA_STATS_LIVE=1`	stats.nba.com tests (residential IP)
`SDV_PY_NBA_RAW_JSON_DIR`	read-through raw JSON store
`SDV_PY_NBA_RAW_JSON_READONLY=1`	offline: miss → error
`SDV_<LEAGUE>_API_KEY`	override a HockeyTech key
`CFBD_API_KEY` · `ODDS_API_KEY`	third-party keys

Football :: college & NFL

R siblings: **cfbfastR** · **recruitR** · **cfbplotR** · **nflfastR** · **nflreadr** · **nflseedR** · Python parity target: **nflreadpy**

COLLEGE FOOTBALL · SPORTSDATAVERSE.CFB

LIVE SCRAPERS

```
from sportsdataverse.cfb import *
espn_cfb_schedule(dates=2024, week=1)
espn_cfb_teams();
espn_cfb_calendar(season=2024)
espn_cfb_game_rosters(game_id=401628334)
espn_cfb_team_roster(team_id=52)
espn_cfb_player_stats(athlete_id=4426502,
season=2024)
```

Play-by-play (ESPN, full pipeline)

```
proc = CFBPlayProcess(gameId=401628334)
pbp  = proc.espn_cfb_pbp()          # dict of
frames
pbp  = proc.run_processing_pipeline()  #
EPA/WPA added
box  = proc.create_box_score()
# offline rebuild from on-disk raw JSON
CFBPlayProcess(gameId, path_to_json=raw_dir,
               odds_override=
{...}).cfb_pbp_disk()
```

Per-play participants come from ESPN's `participants[]` (`cfb_play_participants`) → `{type}_player_name` / `_id` + list columns.

Other sources

stats.ncaa.org: `parse_cfb_ncaa_pbp` · `parse_cfb_ncaa_drives` · `_linescore` · `_officials` · `_team_stats` · `_player_stats` (`cfb_ncaa_pbp` / `cfb_ncaa_box`) · `fox_cfb_pbp` · Yahoo ext · `pff_*` (premium) · Recruiting: `sports247_*` (247 RDB), `on3_*`, `espn_cfb_recruits`.

CFB MODELS (BUNDLED)

Artifact	Purpose
`ep_model`	expected points (rule-era)
`wp_naive` · `wp_spread`	win probability ± Vegas spread
`cfb_cp_model`	completion prob · CPOE
`qbr_model`	QBR-style rating
`fg_model` · `two_pt_model`	kick / conversion prob
`fd_model` · `xpass_model`	4th-down go/kick/punt · pass rate
`punt_distribution` · `cfb_field_position_ep`	decision surfaces

CFB ANALYTICS

```
r = cfb_ratings(seasons=2024, as_of_date=...)
# opp-adjusted
cfb_predict_games(games, r)             #
spread / WP / total
cfb_simulations(games, teams,
simulations=10000)
cfb_adjusted_epa(plays);
cfb_adjusted_tempo(seasons=2024)
cfb_advanced_stats(seasons=2024)        #
success, explosiveness…
cfb_standings(...); cfb_playoff_seeds(...);
cfb_resume(...)
```

Also `cfb_fourth_down` · `cfb_two_point` · `cfb_field_position` · `cfb_opponent_adjust` · `cfb_season_odds` · `cfb_returning_production` · `cfb_roster_talent` · `cfb_transfer_impact` · `cfb_draft_projection` · `cfb_recruiting_projection` · `cfb_crosswalk`.

CFB RELEASE LOADERS (`load_cfb_*`)

Group	Datasets
Core	`pbp pbp_r model_pbp schedule rosters game_rosters team_box player_box drives linescores team_info play_participants`
Stats	`passing rushing receiving percentiles team_summaries team_summaries_weekly`
Advanced	`adv_team adv_team_gamelog adv_passing adv_rushing adv_receiving adv_defensive adv_defensive_players adv_drives adv_situational adv_specialists adv_turnover`
Ratings	`ratings ratings_weekly fpi_weekly power_index`
Betting	`betting betting_lines`
Recruiting	`recruits recruiting_proj team_talent returning_production` + `load_recruit_classes`
Crosswalks	`teams_crosswalk rosters_crosswalk schedule_crosswalk`

```
load_cfb_pbp(seasons=[2024])
load_cfb_ratings_weekly(seasons=[2024])
load_cfb_betting_lines(seasons=[2024])
```

NFL · SPORTSDATAVERSE.NFL

NFLREADPY-PARITY LOADERS

24 canonical `load_nfl_*` loaders; inside `sportsdataverse.nfl` they are also exported under nflreadpy's names (`load_pbp`, `load_schedules`, …).

```
import sportsdataverse.nfl as nfl
pbp = nfl.load_pbp([2024])
sch = nfl.load_schedules([2024])
ngs =
nfl.load_nextgen_stats(stat_type="passing")
adv = nfl.load_pfr_advstats(stat_type="pass",

summary_level="season")
qbr = nfl.load_espn_qbr(seasons=[2024])
```

Loaders

`pbp schedule teams players rosters weekly_rosters player_stats team_stats pbp_participation snap_counts depth_charts injuries officials trades contracts draft_picks combine ff_playerids ff_rankings ff_opportunity ftn_charting nextgen_stats pfr_advstats espn_qbr` + `load_nfl_ratings_weekly`

Config: `get_config()` `update_config()` `reset_config()` `clear_cache()` · `datasets.team_abbr_mapping`, `team_abbr_mapping_norelocate`, `player_name_mapping` · `get_current_season()` `get_current_week()`.

LIVE NFL SOURCES

```
espn_nfl_schedule(season=2024, week=1)
espn_nfl_teams();
espn_nfl_game_rosters(401671889)
NFLPlayProcess(gameId=401671889).run_processing_pipeline()
# api.nfl.com "Shield" (11 endpoints, bearer
auth)
nfl_standings(season=2024); nfl_rosters(...);
nfl_weeks(...)
```

Parsers: 11 dedicated `parse_nfl_*`. PFF: `pff_*` (premium API, Clerk session).

EP / WP / EPA / WPA (EP_WP)

Single owner of NFL model application — a faithful port of nflfastR's `helper_add_ep_wp.R`. Construction modules emit a frame; `ep_wp` enriches it.

```
from sportsdataverse.nfl.ep_wp import *
df = enrich_nfl_pbp(pbp)     #
ep→epa→wp→wpa→cp→cpoe→xyac
calculate_expected_points(df);
calculate_epa(df)
calculate_win_probability(df);
calculate_wpa(df)
calculate_completion_probability(df);
calculate_xyac(df)
calculate_xpass(df)
```

Parity vs nflverse: `ep` 0.996 · `epa` 0.994 · `wp` 0.997 · `vegas_wp` 0.998. Every lag is `.over("game_id")`.

Decision surfaces

`nfl_fourth_down`: `get_go_wp()` `get_fg_wp()` `get_punt_wp()` (nfl4th-style, offline). `nfl_playcall` · `nfl_kicker_rating` · `nfl_special_teams` · `nfl_gamescript` · `nfl_series`.

NFL MODELS (BUNDLED .UBJ)

`ep_model` (18 feat) · `wp_spread` (12) · `wp_naive` (11) · `cp_model` (18) · `fg_model` · `qbr_model` · `two_pt_model` · `xpass_model` · `nfl_playcall` · `punt_data.parquet`. xYAC (76-class) downloads on first use from `nfl_model_artifacts`.

NFL RATINGS, SIMS, PROJECTIONS

`nfl_ratings` · `nfl_simulations` (season / playoff sim; nflseedR-style `nfl_standings_calc`) · `nfl_projection` · `nfl_usage_projection` · `nfl_player_props` · `nfl_market` · `nfl_draft_model` · `nfl_availability` · `nfl_line_grades` · `nfl_roster_builder` · `nfl_ngs_tracking`.

SPRING FOOTBALL · UFL / XFL / CFL / AAF

```
from sportsdataverse.football import *
from sportsdataverse.football.ufl import
espn_ufl_scoreboard, espn_ufl_summary
from
sportsdataverse.football.spring_football_ep_wp
import *
pbp =
build_spring_football_pbp(espn_ufl_summary(game_id),
league="ufl")
enrich_spring_football_pbp(pbp)   # EP/WP via
the NFL ep_wp engine
```

Leagues: `football.ufl` · `.xfl` · `.cfl` · `.aaf` — full ESPN short-name surface each (`espn_ufl_scoreboard`, `espn_ufl_team_roster`, `espn_cfl_standings`, …).

NBA · G-LEAGUE · SPORTSDATAVERSE.NBA

ESPN

```
from sportsdataverse.nba import *
espn_nba_schedule(dates=2025); espn_nba_teams()
espn_nba_pbp(game_id=401584793)   # plays, box, rosters, WP
espn_nba_game_rosters(401584793);
espn_nba_team_roster(13)
espn_nba_player_stats(athlete_id=3975, season=2025)
```

STATS.NBA.COM (128 WRAPPERS)

One stem, one host; `league_id` picks **"00"** NBA · **"20"** G-League · **"15"** Summer League. Uses `curl_cffi impersonate="chrome"` (plain `requests` stalls on the TLS fingerprint block).

```
from sportsdataverse.nba import nba_stats
nba_stats.nba_stats_leaguedashplayerstats(league_id="00")
nba_stats.nba_stats_playercareerstats(player_id="2544")
# dict of frames
nba_stats.nba_stats_playbyplayv3(game_id="0022400001")
nba_stats.nba_stats_boxscoretraditionalv3(game_id="...")
# return_parsed=False → raw resultSets envelope
parse_nba_stats_result_sets(raw, result_set=None)
```

Families: `leaguedash*` · `playerdash*` · `teamdash*` · `boxscore*v3` · `playbyplayv3` · `shotchartdetail` · `synergyplaytypes` · `leaguegamefinder` · `gamerotation` · `video*` · `draftcombine*` · `commonallplayers` · `scoreboardv3`.

Raw JSON read-through store: `SDV_PY_NBA_RAW_JSON_DIR` (per-endpoint `_DIR_{ENDPOINT}`), `_READONLY=1` = fully offline. Season dirs use the END-year convention (2024 = 2023-24).

NBA RELEASE LOADERS

ESPN-derived: `load_nba_pbp schedule team_boxscore player_boxscore rosters game_rosters officials standings player_core player_season_stats team_season_stats shots draft player_impact`. stats.nba.com-derived: `load_nba_stats_pbp(_v3) possessions(_v3) lineups(_v3) shots schedules rosters game_rosters game_lineups player_boxscores team_boxscores player_game_logs player_season_stats team_season_stats standings officials coaches`. Crosswalks: `load_nba_player_crosswalk team_crosswalk schedule_crosswalk`.

POSSESSION ENGINE (PBPSTATS-STYLE)

```
from sportsdataverse.nba.nba_possessions import nba_possessions
poss = nba_possessions("0022400001", league_id="00")
compile_nba_season(2025, season_type="Regular Season")
from sportsdataverse.nba.nba_rapm import nba_rapm, nba_rapm_from_games
nba_rapm(poss)                  # ridge, CV over alphas
nba_adj_rapm(poss, prior)       # Bayesian prior-shrunk
nba_la_rapm(...) nba_decay_rapm(...)
nba_four_factor_rapm(...)
nba_matchup_drapm("2024-25")    # playtype-matchup dRAPM
```

On-court: `nba_lineups.nba_on_court` · `nba_enhanced_pbp` · `nba_play_context(game_id, transition_seconds=6.0)` (CTG-style start types) · `nba_playtype.nba_playtype_ratings` · `nba_shot_zones`. G-League: `nbagl_possessions` `nbagl_enhanced_pbp` `nbagl_on_court` `nbagl_rapm_from_games`.

NBA PLAYER & TEAM VALUE

```
nba_shot_value(player_ids, season="2024-25")  # xPts, shooter talent
nba_bpm(player_logs, team_logs, positions)
nba_spm(box_features, coefficients)
nba_war(ratings, poss, replacement_level=..., pts_per_win=...)
nba_darko(panel, ages)
# Kalman DPM
nba_team_ratings(seasons=2025, as_of_date=...)
nba_game_predict.nba_predict_games(games, ratings)
nba_game_predict.nba_in_game_win_prob(pbp, 0.58)
```

Also `nba_expected_turnovers` · `nba_foul_drawing` · `nba_clutch` · `nba_tracking_value` · `nba_player_props` · `nba_draft_model` · `nba_rookie_projection` · `nba_aging_curve` · `nba_availability`.

Bundled models

`nba_in_game_wp.ubj` · `nba_aging_curve` · `nba_draft_value` · `nba_rookie_projection` · `nba_availability` (+ `wnba_*` twins).

External oracles

`load_darko_dpm(path)` · `load_epm` · `load_lebron_daily/_season` · `load_rapm_ryan_davis` · `load_dunks_threes_stats` · `load_draft_outcomes` (`nba_oracle_data`).

WNBA · SPORTSDATAVERSE.WNBA

SAME SHAPE, WNBA HOST

```
from sportsdataverse.wnba import *
espn_wnba_schedule(dates=2025);
espn_wnba_pbp(game_id)
espn_wnba_team_roster(team_id=5);
espn_wnba_player_stats(...)
from sportsdataverse.wnba import wnba_stats   # 111 wrappers, LeagueID=10
wnba_stats.wnba_stats_leaguedashplayerstats()
wnba_stats.wnba_stats_playbyplayv3(game_id="1022500
```

Engine: `wnba_engine.wnba_possessions` · `wnba_enhanced_pbp` · `wnba_on_court` · `wnba_play_context`. Models: `wnba_team_ratings` · `wnba_game_predict.wnba_predict_games` · `wnba_in_game_win_prob` · `wnba_shot_value` · `wnba_playtype_impact` · `wnba_tracking_value` · `wnba_draft_model` · `wnba_rookie_projection` · `wnba_aging_curve` · `wnba_career_trajectory` · `wnba_availability` · `wnba_clutch` · `wnba_player_props`.

Loaders: `load_wnba_pbp schedule team_boxscore player_boxscore rosters game_rosters officials standings player_core player_season_stats team_season_stats shots draft player_impact` + `load_wnba_stats_*` (pbp possessions lineups shots schedules rosters game_rosters game_lineups player/team boxscores player_game_logs player/team season_stats standings officials coaches draft) + crosswalks.

CROSSWALKS (LEAGUE ↔ COLLEGE)

WNBA↔NBA and WBB↔MBB identity crosswalks: `load_wnba_player_crosswalk` · `load_mbb_player_crosswalk` · `load_wbb_team_crosswalk` · `load_wbb_schedule_crosswalk` · `_common_crosswalk_basketball`. ESPN↔stats.ncaa.org: `mbb_ncaa_espn_crosswalk`.

COLLEGE · SPORTSDATAVERSE.MBB / .WBB

ESPN + STATS.NCAA.ORG

```
from sportsdataverse.mbb import *
espn_mbb_schedule(dates=2025);
espn_mbb_pbp(game_id)
espn_mbb_teams();
espn_mbb_game_rosters(game_id)
# bigballR / wbigballR port — 16 fns per league
ncaa_mbb_team_schedule(team="Duke", season="2024-25")
pbp = ncaa_mbb_game_pbp(game_id)
ncaa_mbb_lineups(pbp);
ncaa_mbb_possessions(pbp)
ncaa_mbb_on_off(...);
ncaa_mbb_player_combos(...)
ncaa_mbb_shot_locations(...);
ncaa_mbb_box_scores(...)
```

Also `ncaa_mbb_date_games team_ids team_roster team_stats player_stats player_lineups play_by_play join_pbp_shots`; `ncaa_wbb_*` mirrors (WBB is halves before 2016). Engine: `mbb_ncaa_*` (fetch → pbp → lineups → stints → possessions → strength, with stint self-healing); names via `display_name_to_roster_key`. Torvik: `torvik_ratings(year)`, `torvik_team_factors`; WBB: `bart_wbb_ratings`.

COLLEGE MODELS & PROJECTIONS

```
r = mbb_team_ratings(seasons=2025)        # league="womens" for WBB
mbb_predict_games(games, r); mbb_season_sim(r, remaining)
mbb_bracketology(2025); mbb_bracket_sim(field, r)
mbb_in_game_win_prob(pbp, pregame_home_prob=0.6)
mbb_box_bpm(2025); mbb_archetypes(2025)
mbb_shot_quality(shots);
mbb_shooter_talent(scored)
mbb_strength_of_schedule([2025])
mbb_transfer_projection(2025);
mbb_recruiting_projection(2025)
mbb_draft_projection(2025)
```

Plus `mbb_rapm` · `solve_rapm_league` (D-I-wide, released as `ncaa_mbb/wbb_rapm`) · `mbb_lineup_stats` · `mbb_luck` · `mbb_positions` · `mbb_shot_selection` · `wbb_*` twins. Bundled: `mbb/wbb_in_game_wp.ubj`, `_box_bpm`, `_archetypes`, `_draft`, `_recruiting`, `_transfer` (.json).

Loaders

`load_mbb_pbp schedule team_boxscore player_boxscore rosters game_rosters officials standings player_core player_season_stats team_season_stats shots ratings player_value` + crosswalks; `load_wbb_*` identical. WP enriched in place in the released pbp.

Hockey · Baseball · Soccer · Cricket · Odds :: and the wider SportsDataverse

R siblings: fastRhockey · baseballr · softballR · oddsapiR · sportsdataversedata

HOCKEY · NHL · PWHL · HOCKEY.*

NHL — FIVE LIVE APIS

```
from sportsdataverse.nhl import *
nhl_web_schedule("2025-01-15")
# api-web.nhle.com
plays = nhl_web_pbp(2024020500)                # parsed by default
nhl_edge_skater_detail(8478402, season=20242025)  # EDGE tracking
nhl_stats_rest_skater_report(...)              # api.nhle.com/stats/rest
nhl_records_awards()                           # records.nhl.com
espn_nhl_pbp(game_id=401559000);
espn_nhl_schedule(dates=2025)
```

Parsers: 19 api-web (`parse_nhl_web_pbp` …), 4 EDGE families, generic stats-REST / records. Loaders: `load_nhl_pbp schedule games rosters player_box team_box goalie_box skater_box shifts scoring penalties shootout linescore officials three_stars scratches shots_by_period`.

NHL analytics

```
nhl_xg(pbp)                                    # shot xG (bundled model)
nhl_expected_assists(pbp);
nhl_faceoff_value(pbp)
nhl_gsax.nhl_goalie_gsax(pbp, shifts)
nhl_rapm.nhl_skater_rapm(pbp, shifts);
nhl_war.nhl_skater_war(...)
nhl_unit_ratings(pbp, shifts);
nhl_team_ratings(seasons=2025)
nhl_in_game_win_prob(pbp, pregame_home_prob=0.55)
```

Also `nhl_penalty_value` · `nhl_zone_transitions` · `nhl_special_teams` · `nhl_edge_value` · `nhl_player_props`.

PWHL + 19 HOCKEYTECH LEAGUES

```
import sportsdataverse as sdv
sdv.pwhl_schedule(season=2025);
sdv.pwhl_pbp(game_id)
sdv.pwhl_game_shifts(game_id);
sdv.pwhl_game_corsi(game_id)
sdv.echl_schedule(season=2026);
sdv.ushl_standings(season=2025)
```

One registry-driven core (`hockeytech.LEAGUES`): 13 callables per league — `<lg>_schedule _pbp _standings _teams _team_roster _player_stats _leaders _game_summary _game_shifts _player_toi _game_corsi _season_id` + `most_recent_<lg>_season`. Leagues: `pwhl ahl ohl whl qmjhl echl sphl chl ushl bchl ajhl sjhl ojhl cchl gojhl mhl nojhl vijhl kijhl mjhl`. PWHL extras: `pwhl_goalie_gsax` · `pwhl_skater_rapm` · `pwhl_in_game_win_prob` · `pwhl_team_ratings`. College hockey: `espn_mch_*` / `espn_wch_*`, `college_hockey_ratings`.

BASEBALL · MLB · BASEBALL

MLB STATS API + ESPN

```
from sportsdataverse.mlb import *
mlb_schedule(date="2025-07-04")          # statsapi.mlb.com
mlb_boxscore(game_pk=745000);
mlb_play_by_play(game_pk)
espn_mlb_pbp(game_id=401570000);
espn_mlb_schedule(dates=2025)
```

~80 Stats-API wrappers (`mlb_*`). `load_mlb_*`: `batter_projection catcher_framing command_plus expected_hr expected_stats oaa re24_matrix stuff_plus we_table wpa xera`.

STATCAST — BASEBALL SAVANT (43 ENDPOINTS)

```
mlb_statcast_search("2025-04-01", "2025-04-07",
                    player_type="pitcher",
pitch_type="FF")
mlb_statcast_leaderboard_expected_stats(year=2025)
# 37 boards
mlb_statcast_gamefeed(game_pk=745000)
mlb_statcast_player(660271, section="statcast")
```

Families `search` (chunked; `_minors`, `_wbc`) · `leaderboard` · `gamefeed` · `player`; friendly filters (`season pitch_type at_bat_result`) translate to Savant params.

MLB ANALYTICS

```
mlb_stuff_plus(pitches);
mlb_command_plus(pitches)
mlb_expected_stats("2025-04-01", "2025-09-30")
mlb_catcher_framing(pitches);
mlb_fielding_oaa(bip)
mlb_run_expectancy_matrix(seasons=2024);
mlb_win_expectancy(pbp, results)
mlb_pitch_classify(pitches);
mlb_batter_projection(2026)
```

Also `mlb_expected_home_runs` · `mlb_run_values` · `mlb_swing_decision` · `mlb_pitch_sequencing` · `mlb_pitch_fatigue` · `mlb_stolen_base` · `mlb_baserunning` · `mlb_umpire_zone` · `mlb_team_projection` · `mlb_prop_projection`.

COLLEGE BASEBALL & SOFTBALL

```
from sportsdataverse.baseball.college_baseball import *
espn_college_baseball_scoreboard(dates=2025)
parse_college_baseball_ncaa_pbp(html)   # stats.ncaa.org
college_baseball_re24(...);
college_baseball_wpa(...)  # + softball
```

SOCCER · CRICKET · ODDS · WIN-EXPECTANCY

SOCCER (ESPN, 12 LEAGUES)

```
from sportsdataverse.soccer.epl import
espn_epl_scoreboard
espn_epl_scoreboard(dates=20250315)
espn_soccer_team_schedule("eng.1", team_id=359, season=2025)
```

Per-league modules `epl mls nwsl laliga bundesliga seriea ligue1 ligamx ucl uel wc wwc` (full ESPN surface each) + league-parameterized `espn_soccer_*`.

CRICKET (ESPN + BUNDLED WP)

```
from sportsdataverse.cricket import *
espn_cricket_scoreboard(league="icc", dates=2025)
state = cricket_match_state(pbp)
cricket_win_probability(state);
cricket_wpa(state_wp)
```

ODDS — THE ODDS API (`ODDS_API_KEY`)

```
from sportsdataverse.odds import *
toa_sports(); toa_usage()
toa_sports_odds("basketball_nba", regions="us", markets="h2h,spreads")
toa_event_odds(sport, event_id, markets="player_points")
toa_sports_odds_history(sport, date="2024-11-29T22:45:00Z")
toa_sports_scores(...); toa_sports_events(...)
```

Historical lines also ride the release loaders (`load_cfb_betting_lines`, `load_nfl_schedule`).

WIN-EXPECTANCY LAB (`WEXP`)

```
from sportsdataverse import wexp
oracle = wexp.build_cfb_market_oracle(...)  # or _nfl_
pred = wexp.build_predictor(wexp.VariantConfig(...))
wexp.backtest.run_backtest(oracle, pred, model_id="ridge")
```

Ridge-margin, Elo, Glickman–Stern predictors vs market closing lines; Brier + calibration scoring. Football-only today.

THE WIDER SPORTSDATAVERSE

R PACKAGES (SPORTSDATAVERSE.ORG)

Package	Domain	sdv-py mirror
cfbfastR	college football	`cfb` · cfbfastR-data loaders
hoopR	NBA · men's college	`nba mbb`
wehoop	WNBA · women's college	`wnba wbb`
baseballr	MLB · Statcast · college	`mlb baseball`
fastRhockey	NHL · PWHL · HockeyTech	`nhl pwhl hockey`
softballR	college softball	`baseball.college_softball`
recruitR	CFB recruiting	`cfb.sports247_* on3_*`
oddsapiR	The Odds API	`odds`
nflseedR · cfbseedR	season sims / seeding	`nfl_simulations cfb_simulations`
sportsdataversedata	release publishing	`release`

Related: nflfastR / nflreadr (nflverse) · cfbplotR · sportyR / sportypy · sportsdataverse-js.

DATA REPOS BEHIND THE LOADERS

Each sport has a `*-raw` (scraped JSON, committed per game) and a `*-data` (tidy parquet on GitHub releases) repo under `github.com/sportsdataverse`. Loaders read the release asset; `sportsdataverse.release` writes them.

```
sportsdataverse_save(df, "pbp_2025", "cfb", release_tag="cfb_pbp")
```

CONVENTIONS TO REMEMBER

- polars 1.x API everywhere; `return_as_pandas=True` converts at the edge.
- IDs are join keys — one dtype per id; never paper over with a float→str cast.
- Live tests gate on `SDV_PY_LIVE_TESTS=1`; stats.nba.com on `SDV_PY_NBA_STATS_LIVE=1`.
- Reference docs + returns tables are generated from endpoint YAML (`tools/codegen`).

Models :: the cross-sport modeling grid, bundled artifacts, validation gates

Every row below is code that ships in the wheel — no notebooks, no downloads (except NFL xYAC).

ONE SPINE, EVERY SPORT

Each league implements the same five layers. Names are the public entry points; `·` separates alternatives, — means not built for that league.

League	Team ratings	Game predict	Season / bracket sim	In-game WP	Player value
CFB	`cfb_ratings`	`cfb_predict_games`	`cfb_simulations` · `cfb_season_odds` · `cfb_playoff_seeds`	`wp_spread` / `wp_naive` via `CFBPlayProcess`	`cfb_adjusted_epa` · `cfb_roster_talent` · `cfb_draft_projection`
NFL	`nfl_ratings`	`nfl_market` · `nfl_gamescript`	`nfl_simulations` (+ `nfl_standings_calc`)	`enrich_nfl_pbp` → `wp`, `vegas_wp`	`nfl_projection` · `nfl_kicker_rating` · `nfl_line_grades`
NBA / G-Lg	`nba_team_ratings`	`nba_game_predict.nba_predict_games`	— (schedule-driven via predict)	`nba_in_game_win_prob`	`nba_rapm` · `nba_adj_rapm` · `nba_bpm` · `nba_spm` · `nba_war` · `nba_darko` · `nba_shot_value`
WNBA	`wnba_team_ratings`	`wnba_game_predict.wnba_predict_games`	—	`wnba_in_game_win_prob`	`wnba_engine` RAPM · `wnba_shot_value` · `wnba_playtype_impact`
MBB / WBB	`mbb_team_ratings` · `torvik_ratings`	`mbb_predict_games`	`mbb_season_sim` · `mbb_bracketology` · `mbb_bracket_sim`	`mbb_in_game_win_prob`	`mbb_box_bpm` · `mbb_rapm` · `mbb_shot_quality` · `mbb_archetypes` (+ `wbb_*`)
NHL	`nhl_team_ratings`	`nhl_market.nhl_predict_games`	—	`nhl_in_game_win_prob`	`nhl_xg` · `nhl_skater_rapm` · `nhl_skater_war` · `nhl_goalie_gsax` · `nhl_unit_ratings`
PWHL	`pwhl_team_ratings`	`pwhl_market.pwhl_predict_games`	—	`pwhl_in_game_win_prob`	`pwhl_skater_rapm` · `pwhl_skater_war` · `pwhl_goalie_gsax`
MLB	`mlb_team_projection`	— (projection-driven)	—	`mlb_win_expectancy`	`mlb_stuff_plus` · `mlb_command_plus` · `mlb_expected_stats` · `mlb_catcher_framing` · `mlb_fielding_oaa`
Others	college hockey `college_hockey_ratings` · cricket `cricket_win_probability` / `cricket_wpa` · spring football reuses the NFL engine (`build_spring_football_pbp` → `enrich_spring_football_pbp`)				

Shared shape: ratings take `seasons=` + `as_of_date=` (leak-safe, exclusive cut); predictors take `(games, ratings)`; sims take `(ratings | games, …, n_sims / simulations)`; in-game WP takes `(pbp, pregame_home_prob)`.

DRAFT, PROJECTION & ROSTER MODELS

Family	Entry points
Draft value / rookies	`nba_draft_model` · `nba_rookie_projection` · `wnba_draft_model` · `nfl_draft_model` · `mbb_draft_projection` · `cfb_draft_projection`
Aging & availability	`nba_aging_curve` · `nba_availability` · `wnba_career_trajectory` · `nfl_availability`
Recruiting / transfers	`cfb_recruiting_projection` · `cfb_transfer_impact` · `mbb_recruiting_projection` · `mbb_transfer_projection`
Props & markets	`nba_player_props` · `wnba_player_props` · `nfl_player_props` · `nhl_player_props` · `pwhl_player_props` · `mlb_prop_projection` · `nfl_market` · `nhl_market`

BUNDLED ARTIFACT REGISTRY

Shipped as package data (`importlib.resources`); XGBoost `.ubj` unless noted. Feature counts from the model cards.

Dir	Artifacts
`cfb/models`	`ep_model` 8f · `wp_naive` 12f · `wp_spread` 13f · `cfb_cp_model` · `qbr_model` 10f · `fg_model` 5f · `fd_model` · `two_pt_model` 4f · `xpass_model` 7f · `punt_distribution.parquet` · `cfb_field_position_ep.parquet`
`nfl/models`	`ep_model` 18f · `wp_spread` 12f · `wp_naive` 11f · `cp_model` 18f · `fg_model` · `qbr_model` · `two_pt_model` · `xpass_model` · `nfl_playcall` · `punt_data.parquet`; xYAC (76-class) fetched on first use
`nba/models`	`nba_in_game_wp` · `wnba_in_game_wp` · `{nba,wnba}_aging_curve` · `_draft_value` · `_rookie_projection` · `_availability` (.json)
`mbb/models`	`{mbb,wbb}_in_game_wp` · `_box_bpm` · `_archetypes` · `_draft` · `_recruiting` · `_transfer` (.json)
`nhl/models`	`nhl_in_game_wp.json` (+ xG coefficients in `nhl_xg`)
`mlb/models`	`mlb_stuff_plus` · `mlb_command_plus`
`cricket/models`	`cricket_resource_surface.parquet` · `cricket_winprob_calibration.parquet`

Every model's fitting script and era cuts live in the `*-data` producer repo that trains it (cfbfastR-cfb-data, nfl-data, hoopR-nba-data …); constants in `*_constants.py` / `model_vars.py` cite them.

RULE-ERA & FEATURE CONVENTIONS

- NFL era cuts `ERA_SEASON_CUTS` = 2001 / 2005 / 2013 / 2017; CFB models are per rule era.
- Kickoff / PAT feature substitution: touchback yardline 80 pre-2016, 75 from 2016; `down→1`, `ydstogo→10`.
- Spread WP: `spread_time` decays with exponent −4.0 (`SPREAD_TIME_DECAY_EXPONENT`).
- Outputs are `Float64`; models emit float32 — cast at the boundary.

VALIDATION & ORACLE GATES

Every model ships with a gate against an external oracle captured as a committed fixture. Gates are measured from observed values and never lowered.

Metric	Used for
Brier · log-loss · calibration table	win probabilities, in-game WP, sims (`brier_score`, `calibration_table` in `*_prediction_constants` / `wexp`)
MAE vs market	spreads / totals (`nfl_market`, `cfb_season_odds`, wexp market oracles)
Spearman ρ	ratings vs Torvik / KenPom / ESPN FPI / MoneyPuck; player value vs EPM, DARKO, RAPM, LEBRON
Correlation floors	parity with the R producers (nflfastR EP 0.996, CFB ratings ρ 0.97)

Oracle loaders

`load_darko_dpm` · `load_epm` · `load_lebron_daily` · `load_rapm_ryan_davis` · `load_dunks_threes_stats` · `load_draft_outcomes` · `torvik_ratings` · `bart_wbb_ratings` · `load_cfb_fpi_weekly` · `load_cfb_power_index` · `wexp.build_cfb_market_oracle` / `_nfl_`.

LEAK-SAFETY CONTRACT

- `as_of_date` / `through_week` are **exclusive**: the cut game is never in the training window.
- Every lag, cumulative sum or shift is `.over("game_id")` (or season) — no cross-game bleed when frames are concatenated.
- Ratings for a date use only games with `game_date < as_of_date`; sims draw from those ratings.
- Fitted constants (HFA, σ, λ) name the fitting script; assert the **output** changed, not that the code ran.

WIN-EXPECTANCY LAB (WEXP)

```
from sportsdataverse import wexp
o  = wexp.build_cfb_market_oracle(...)
p  =
wexp.build_predictor(wexp.VariantConfig(...))
wexp.backtest.run_backtest(o, p,
model_id="ridge")
```

Ridge-margin, Elo, Glickman–Stern vs closing lines; vintage store for reproducible walk-forward runs.

Recipes :: end-to-end pipelines, polars idioms, gotchas

Copy-paste starting points — each runs as written given the named inputs.

CFB · RAW JSON → ENRICHED PBP (OFFLINE)

```
from sportsdataverse.cfb import CFBPlayProcess
proc = CFBPlayProcess(gameId=401628334,
    path_to_json="raw/2024/",
    odds_override={"gameSpread": -6.5,
"overUnder": 55.5,
                   "homeFavorite": True,
                   "gameSpreadAvailable":
True})
proc.cfb_pbp_disk()
pbp = proc.run_processing_pipeline()   #
EPA/WPA, no network
box = proc.create_box_score()
```

The override injects persisted odds so the rebuild never falls back to the live odds endpoint (provenance: `pbp_txt["odds_source"] == "injected"`).

NFL · LOAD → ENRICH → 4TH-DOWN TABLE

```
import sportsdataverse.nfl as nfl
from sportsdataverse.nfl.ep_wp import
enrich_nfl_pbp
from sportsdataverse.nfl.nfl_fourth_down import
get_4th_down_probs
pbp = nfl.load_pbp([2024])
df  = enrich_nfl_pbp(pbp)          #
ep→epa→wp→wpa→cp→xyac
fourth = get_4th_down_probs(
    df.filter(pl.col("down") == 4))
```

`get_4th_down_probs` adds go / FG / punt WP + the recommendation columns (nfl4th-style, fully offline).

SPRING FOOTBALL ON THE NFL ENGINE

```
from sportsdataverse.football.ufl import
espn_ufl_summary
from
sportsdataverse.football.spring_football_ep_wp
import *
raw = espn_ufl_summary(game_id)
pbp = build_spring_football_pbp(raw,
league="ufl")
pbp = enrich_spring_football_pbp(pbp)
```

NBA · POSSESSIONS → RAPM → WAR

```
from sportsdataverse.nba.nba_season_compile
import compile_nba_season
from sportsdataverse.nba.nba_rapm import
nba_rapm_from_games
from sportsdataverse.nba import nba_war
poss  = compile_nba_season(2025)   # end-year:
2024-25
rapm  =
nba_rapm_from_games(poss["game_id"].unique())
value = nba_war(rapm, poss,
                replacement_level=-2.0,
pts_per_win=30.0)
```

Set `SDV_PY_NBA_RAW_JSON_DIR` first and the compile reads disk before network; add `_READONLY=1` for a fully offline, reproducible build (a miss raises `RawStoreMissError`).

MBB · STATS.NCAA.ORG PBP → LINEUPS → ON/OFF

```
from sportsdataverse.mbb import
(ncaa_mbb_game_pbp,
    ncaa_mbb_lineups, ncaa_mbb_possessions,
ncaa_mbb_on_off)
pbp  = ncaa_mbb_game_pbp(6305900)
lu   = ncaa_mbb_lineups(pbp)
poss = ncaa_mbb_possessions(pbp)
oo   = ncaa_mbb_on_off("PlayerName", lu)
```

WBB mirrors every call (`ncaa_wbb_*`); pre-2016 WBB pbp is halves, not quarters — 0 rows for a quarters filter is data, not an error.

MLB · STATCAST SEARCH → STUFF+

```
from sportsdataverse.mlb import
mlb_statcast_search, mlb_stuff_plus
pitches = mlb_statcast_search("2025-04-01",
"2025-04-30",
    player_type="pitcher", pitch_type="FF")
stuff = mlb_stuff_plus(pitches, level="pitch")
```

POLARS IDIOMS FOR PBP WORK

```
# EPA per drive, success rate — group, don't
loop
pbp.group_by("game_id", "drive_id",
maintain_order=True) \
   .agg(pl.col("epa").sum(), (pl.col("epa") >
0).mean()
        .alias("success_rate"))

# lags NEVER cross a game boundary
pbp.with_columns(pl.col("score_diff").shift(1)

.over("game_id").alias("prev_diff"))

# explicit bool masks (house style)
pbp.filter((pl.col("pass") == True) &
           (pl.col("garbage_time") == False))

# no lookaround in Rust regex — case-toggle
instead
pl.col("text").str.extract(
    r"(?i)sacked by(?-i: ([A-Z][\w'\.\-]+))",
1)

# pandas only at the very edge
df = any_loader(..., return_as_pandas=True)
```

CROSS-LEAGUE JOINS

```
# pin ONE dtype per id, assert before joining
a =
a.with_columns(pl.col("athlete_id").cast(pl.Int64))
assert a.schema["athlete_id"] ==
b.schema["athlete_id"]
a.join(b, on="athlete_id", how="left")
# ESPN ⇔ stats.nba.com identity
xwalk = sdv.load_nba_player_crosswalk(seasons=
[2025])
```

Never stringify a float id (`"123.0"` ≠ `"123"`); if you must go Utf8, cast through Int64 first.

GOTCHA STRIP

Trap	Rule
Season year	NBA/WNBA-stats dirs use the **end year** (2024 = 2023-24); ESPN loaders use kickoff/tip year; NBA span params want `"2023-24"`
stats.nba.com	TLS-fingerprint block → silent hang, not an error; the runtime uses `curl_cffi impersonate="chrome"`; run live only from residential IPs
Game ids	stats.nba.com `game_id` is a zero-padded **string** (`"0022400001"`) — never int-cast it
WBB history	halves before 2016 — quarter-keyed logic silently empties six seasons
Cache	`@cached_loader` keys on (name, args) not URL — `clear_cache()` after changing loader targets
Empty ≠ error	parsers return zero-row frames with the documented schema; check `df.is_empty()`, don't try/except
ESPN rate	Core v2 403s under aggressive parallelism — keep scrapes low-concurrency

TEST GATES

```
uv run pytest                              # offline
suite
SDV_PY_LIVE_TESTS=1 uv run pytest          # live APIs
SDV_PY_NBA_STATS_LIVE=1 ...                #
stats.nba.com only,
                                           #
residential IP
```

CONTRIBUTING A NEW ENDPOINT

- Wrapper metadata lives in `tools/codegen/endpoints/*.yaml`; run `uv run python tools/codegen/generate.py`, never hand-edit generated files.
- Returns-table descriptions go in `manual_column_descriptions.yaml` (schemas/** is clobbered on re-capture).
- Validate parsers against **real captured fixtures**, never synthetic ones.
- New modules are fully typed and join the mypy ratchet in `pyproject.toml`.
- `generate.py --check` is the CI drift gate — regenerate before every push.

SPORTSDATAVERSE-PY :: CHEAT SHEET

The Python sister to the SportsDataverse R packages (wehoop, hoopR, cfbfastR, baseballr, fastRhockey, nflfastR). One `pip install` gives polars-first, tidy access to play-by-play, box scores, schedules, rosters and odds across 15+ sports, plus bundled EP / WP / rating / projection models and release-dataset loaders. Page 1 of 4: getting started, coverage map, the ESPN cross-league layer, utilities.

v0.0.75

Python 3.9 – 3.14 · polars 1.x

py.sportsdataverse.org

GET STARTED

```
pip install sportsdataverse
pip install "sportsdataverse[all]"  # + models, curl_cffi, tests
uv add sportsdataverse

import sportsdataverse as sdv
from sportsdataverse.cfb import load_cfb_pbp
```

Every public callable is also on the flat `sdv.*` namespace (4,600+ names). Sub-packages: `cfb` `nfl` `nba` `nbagl` `wnba` `mbb` `wbb` `nhl` `pwhl` `hockey` `hockeytech` `mlb` `baseball` `soccer` `cricket` `football` `odds` `wexp`.

RETURN CONVENTIONS

Flag	Effect
default	`polars.DataFrame`, snake_case columns
`return_as_pandas=True`	pandas frame (same data)
`raw=True`	scrapers: untouched JSON `dict`
`return_parsed=False`	ESPN wrappers: tidy frame (default) → raw `dict`
multi-table	`dict[str, pl.DataFrame]` keyed by section
empty / malformed	zero-row frame with documented schema — never raises

THREE SURFACES

1 · Release-dataset loaders

`load_<sport>_<dataset>(seasons, return_as_pandas=False)` reads parquet from the SportsDataverse GitHub releases (the `*-data` repos). Cached; zero scraping.

```
pbp = load_cfb_pbp(seasons=[2023, 2024])
sch = sdv.load_nba_schedule(seasons=range(2020, 2025))
```

2 · Live API wrappers

`espn_<league>_*`, `nba_stats_*`, `wnba_stats_*`, `nhl_web_*`, `nhl_edge_*`, `mlb_*`, `mlb_statcast_*`, `<hockeytech-league>_*`, `ncaa_mbb_*`, `toa_*` (odds).

3 · Processing + models

PBP processors (`CFBPlayProcess`, `NFLPlayProcess`), `enrich_nfl_pbp`, `nba_possessions`, ratings, simulations, projections — see pages 2–4.

COVERAGE MAP

Sport	Sub-package	Live sources	R sibling
NFL	`nfl`	ESPN · api.nfl.com · PFF	nflfastR · nflreadr
CFB	`cfb`	ESPN · stats.ncaa.org · Fox · Yahoo · 247 · On3 · PFF	cfbfastR · recruitR
NBA · G-League · Summer	`nba` `nbagl`	ESPN · stats.nba.com · Fox	hoopR
WNBA	`wnba`	ESPN · stats.wnba.com	wehoop
Men's college bb	`mbb`	ESPN · stats.ncaa.org · Torvik	hoopR · bigballR
Women's college bb	`wbb`	ESPN · stats.ncaa.org · Bart	wehoop · wbigballR
NHL	`nhl`	ESPN · api-web.nhle.com · EDGE · stats REST · records	fastRhockey
PWHL + 19 minor/junior	`pwhl` `hockey.*`	HockeyTech / LeagueStat	fastRhockey
College hockey M/W	`hockey.mch` `.wch`	ESPN	—
MLB	`mlb`	ESPN · statsapi.mlb.com · Baseball Savant	baseballr
College baseball / softball	`baseball`	ESPN · stats.ncaa.org	baseballr · softballR
Soccer (12 leagues)	`soccer.*`	ESPN	—
Cricket	`cricket`	ESPN + bundled WP	—
UFL · XFL · CFL · AAF	`football.*`	ESPN + spring EP/WP	—
Odds & lines	`odds`	The Odds API	oddsapiR
Win-expectancy lab	`wexp`	CFB / NFL market oracles	—

Release data repos: `cfbfastR-data`, `nfl-data`, `hoopR-nba-data`, `hoopR-mbb-data`, `wehoop-wnba-data`, `wehoop-wbb-data`, `fastRhockey-*-data`, `baseballr-data`, `ncaa-mbb/wbb-data`, `*-stats-data`.

ESPN CROSS-LEAGUE LAYER

One core (`_common_espn`) × N thin league modules. Every league gets the same **121 endpoint short names** as `espn_<prefix>_<short>` (≈125 wrappers per league; 800+ total across nba wnba nfl cfb mbb wbb nhl mlb, plus soccer, cricket, college baseball/hockey and spring football).

```
from sportsdataverse.nba import espn_nba_team_roster
df  = espn_nba_team_roster(team_id=13)
# polars (default)
raw = espn_nba_team_roster(team_id=13,

return_parsed=False)   # raw dict
```

Family	Short names
Schedule	`schedule calendar scoreboard games season_weeks season_week_games`
Teams	`teams team team_roster team_schedule team_record team_injuries team_depthcharts franchise venue`
Games	`summary game game_rosters game_plays play_participants game_odds game_officials game_predictor game_probabilities game_propbets`
Players	`player_stats player_gamelog player_splits player_career_stats player_bio player_contracts player_vs_player`
League	`standings leaders news injuries transactions draft awards conferences futures season_qbr`
NCAA extras	`rankings fpi recruits recruiting_rankings` (mbb wbb cfb …)

Parser layer

`parse_summary(payload, section=None)` → 21 sub-frames: `boxscore_player` `boxscore_team` `plays` `winprobability` `leaders` `game_info` `officials` `header` `season_series` `against_the_spread` `standings` `broadcasts` `format` `pickcenter` `odds` `article` `injuries` `news` `drives` `drive_plays` `scoring_plays`. Generic fall-throughs: `parse_single_entity`, `parse_items`.

Full PBP processors (tidy, per game)

```
espn_nba_pbp(401584793)    espn_wnba_pbp(...)
espn_mbb_pbp(...)          espn_wbb_pbp(...)
espn_nhl_pbp(...)          espn_mlb_pbp(...)
CFBPlayProcess(gameId=401628334)   # page 2
NFLPlayProcess(gameId=401671889)   # page 2
```

DISCOVERY

```
sdv.find_team("Duke", "mbb")
sdv.find_athlete("Caitlin Clark", "wnba", team="Fever")
sdv.find_event("2024-11-30", "cfb", home="Ohio State")
sdv.function_count()            # per league
sdv.list_functions("nhl", search="pbp")
```

CLI mirrors it: `sdv find-team Duke --league mbb`, `sdv function-count`, `sdv cache stats|clear|mode`.

CACHE

```
sdv.set_cache_mode("filesystem")  # memory | filesystem | off
sdv.set_default_ttl(3600)
sdv.cache_stats();  sdv.clear_cache()
# NFL has its own nflreadpy-style config
from sportsdataverse.nfl import update_config
update_config(cache_mode="filesystem", cache_duration=3600)
```

Env: `SDV_PY_NFL_CACHE` · `SDV_PY_NFL_CACHE_DIR` · `SDV_PY_NFL_CACHE_DURATION`.

HTTP, TIDY & RELEASE HELPERS

`dl_utils.download(url)` — single retrying HTTP chokepoint. `underscore()` `camelize()` `kebabize()`, `flatten_json_iterative()`, `key_check()`.

`sportsdataverse.release` — port of the **sportsdataversedata** R package: `sportsdataverse_upload()`, `sportsdataverse_save()` (parquet / rds / csv.gz), `gh_cli_release_upload()`, `gh_cli_release_assets()`, `gh_cli_release_tags()`.

Errors: `NoESPNDataError` · `SeasonNotFoundError` · `RawStoreMissError`.

ENV VARS WORTH KNOWING

`SDV_PY_LIVE_TESTS=1`	run gated live tests
`SDV_PY_NBA_STATS_LIVE=1`	stats.nba.com tests (residential IP)
`SDV_PY_NBA_RAW_JSON_DIR`	read-through raw JSON store
`SDV_PY_NBA_RAW_JSON_READONLY=1`	offline: miss → error
`SDV_<LEAGUE>_API_KEY`	override a HockeyTech key
`CFBD_API_KEY` · `ODDS_API_KEY`	third-party keys

COLLEGE FOOTBALL · SPORTSDATAVERSE.CFB

LIVE SCRAPERS

```
from sportsdataverse.cfb import *
espn_cfb_schedule(dates=2024, week=1)
espn_cfb_teams();
espn_cfb_calendar(season=2024)
espn_cfb_game_rosters(game_id=401628334)
espn_cfb_team_roster(team_id=52)
espn_cfb_player_stats(athlete_id=4426502,
season=2024)
```

Play-by-play (ESPN, full pipeline)

```
proc = CFBPlayProcess(gameId=401628334)
pbp  = proc.espn_cfb_pbp()          # dict of
frames
pbp  = proc.run_processing_pipeline()  #
EPA/WPA added
box  = proc.create_box_score()
# offline rebuild from on-disk raw JSON
CFBPlayProcess(gameId, path_to_json=raw_dir,
               odds_override=
{...}).cfb_pbp_disk()
```

Per-play participants come from ESPN's `participants[]` (`cfb_play_participants`) → `{type}_player_name` / `_id` + list columns.

Other sources

stats.ncaa.org: `parse_cfb_ncaa_pbp` · `parse_cfb_ncaa_drives` · `_linescore` · `_officials` · `_team_stats` · `_player_stats` (`cfb_ncaa_pbp` / `cfb_ncaa_box`) · `fox_cfb_pbp` · Yahoo ext · `pff_*` (premium) · Recruiting: `sports247_*` (247 RDB), `on3_*`, `espn_cfb_recruits`.

CFB MODELS (BUNDLED)

Artifact	Purpose
`ep_model`	expected points (rule-era)
`wp_naive · wp_spread`	win probability ± Vegas spread
`cfb_cp_model`	completion prob · CPOE
`qbr_model`	QBR-style rating
`fg_model · two_pt_model`	kick / conversion prob
`fd_model · xpass_model`	4th-down go/kick/punt · pass rate
`punt_distribution · cfb_field_position_ep`	decision surfaces

CFB ANALYTICS

```
r = cfb_ratings(seasons=2024, as_of_date=...)
# opp-adjusted
cfb_predict_games(games, r)             #
spread / WP / total
cfb_simulations(games, teams,
simulations=10000)
cfb_adjusted_epa(plays);
cfb_adjusted_tempo(seasons=2024)
cfb_advanced_stats(seasons=2024)        #
success, explosiveness…
cfb_standings(...); cfb_playoff_seeds(...);
cfb_resume(...)
```

Also `cfb_fourth_down` · `cfb_two_point` · `cfb_field_position` · `cfb_opponent_adjust` · `cfb_season_odds` · `cfb_returning_production` · `cfb_roster_talent` · `cfb_transfer_impact` · `cfb_draft_projection` · `cfb_recruiting_projection` · `cfb_crosswalk`.

CFB RELEASE LOADERS (`LOAD_CFB_*`)

Group	Datasets
Core	`pbp pbp_r model_pbp schedule rosters game_rosters team_box player_box drives linescores team_info play_participants`
Stats	`passing rushing receiving percentiles team_summaries team_summaries_weekly`
Advanced	`adv_team adv_team_gamelog adv_passing adv_rushing adv_receiving adv_defensive adv_defensive_players adv_drives adv_situational adv_specialists adv_turnover`
Ratings	`ratings ratings_weekly fpi_weekly power_index`
Betting	`betting betting_lines`
Recruiting	`recruits recruiting_proj team_talent returning_production` + `load_recruit_classes`
Crosswalks	`teams_crosswalk rosters_crosswalk schedule_crosswalk`

```
load_cfb_pbp(seasons=[2024])
load_cfb_ratings_weekly(seasons=[2024])
load_cfb_betting_lines(seasons=[2024])
```

NFL · SPORTSDATAVERSE.NFL

NFLREADPY-PARITY LOADERS

24 canonical `load_nfl_*` loaders; inside `sportsdataverse.nfl` they are also exported under nflreadpy's names (`load_pbp`, `load_schedules`, …).

```
import sportsdataverse.nfl as nfl
pbp = nfl.load_pbp([2024])
sch = nfl.load_schedules([2024])
ngs =
nfl.load_nextgen_stats(stat_type="passing")
adv = nfl.load_pfr_advstats(stat_type="pass",

summary_level="season")
qbr = nfl.load_espn_qbr(seasons=[2024])
```

Loaders

`pbp schedule teams players rosters weekly_rosters player_stats team_stats pbp_participation snap_counts depth_charts injuries officials trades contracts draft_picks combine ff_playerids ff_rankings ff_opportunity ftn_charting nextgen_stats pfr_advstats espn_qbr` + `load_nfl_ratings_weekly`

Config: `get_config() update_config() reset_config() clear_cache()` · `datasets.team_abbr_mapping`, `team_abbr_mapping_norelocate`, `player_name_mapping` · `get_current_season()` `get_current_week()`.

LIVE NFL SOURCES

```
espn_nfl_schedule(season=2024, week=1)
espn_nfl_teams();
espn_nfl_game_rosters(401671889)
NFLPlayProcess(gameId=401671889).run_processing_pip
# api.nfl.com "Shield" (11 endpoints, bearer
auth)
nfl_standings(season=2024); nfl_rosters(...);
nfl_weeks(...)
```

Parsers: 11 dedicated `parse_nfl_*`. PFF: `pff_*` (premium API, Clerk session).

EP / WP / EPA / WPA (EP_WP)

Single owner of NFL model application — a faithful port of nflfastR's `helper_add_ep_wp.R`. Construction modules emit a frame; `ep_wp` enriches it.

```
from sportsdataverse.nfl.ep_wp import *
df = enrich_nfl_pbp(pbp)     #
ep→epa→wp→wpa→cp→cpoe→xyac
calculate_expected_points(df);
calculate_epa(df)
calculate_win_probability(df);
calculate_wpa(df)
calculate_completion_probability(df);
calculate_xyac(df)
calculate_xpass(df)
```

Parity vs nflverse: `ep` 0.996 · `epa` 0.994 · `wp` 0.997 · `vegas_wp` 0.998. Every lag is `.over("game_id")`.

Decision surfaces

`nfl_fourth_down`: `get_go_wp()` `get_fg_wp()` `get_punt_wp()` (nfl4th-style, offline). `nfl_playcall` · `nfl_kicker_rating` · `nfl_special_teams` · `nfl_gamescript` · `nfl_series`.

NFL MODELS (BUNDLED .UBJ)

`ep_model` (18 feat) · `wp_spread` (12) · `wp_naive` (11) · `cp_model` (18) · `fg_model` · `qbr_model` · `two_pt_model` · `xpass_model` · `nfl_playcall` · `punt_data.parquet`. xYAC (76-class) downloads on first use from `nfl_model_artifacts`.

NFL RATINGS, SIMS, PROJECTIONS

`nfl_ratings` · `nfl_simulations` (season / playoff sim; nflseedR-style `nfl_standings_calc`) · `nfl_projection` · `nfl_usage_projection` · `nfl_player_props` · `nfl_market` · `nfl_draft_model` · `nfl_availability` · `nfl_line_grades` · `nfl_roster_builder` · `nfl_ngs_tracking`.

SPRING FOOTBALL · UFL / XFL / CFL / AAF

```
from sportsdataverse.football import *
from sportsdataverse.football.ufl import
espn_ufl_scoreboard, espn_ufl_summary
from
sportsdataverse.football.spring_football_ep_wp
import *
pbp =
build_spring_football_pbp(espn_ufl_summary(game_id),
league="ufl")
enrich_spring_football_pbp(pbp)   # EP/WP via
the NFL ep_wp engine
```

Leagues: `football.ufl` · `.xfl` · `.cfl` · `.aaf` — full ESPN short-name surface each (`espn_ufl_scoreboard`, `espn_ufl_team_roster`, `espn_cfl_standings`, …).

BASKETBALL :: NBA · WNBA · G-LEAGUE · MEN'S & WOMEN'S COLLEGE

R siblings: hoopR · wehoop · bigballR · wbigballR · Python: nba_api (stats API reference)

NBA · G-LEAGUE · SPORTSDATAVERSE.NBA

ESPN

```
from sportsdataverse.nba import *
espn_nba_schedule(dates=2025);
espn_nba_teams()
espn_nba_pbp(game_id=401584793)   # plays,
box, rosters, WP
espn_nba_game_rosters(401584793);
espn_nba_team_roster(13)
espn_nba_player_stats(athlete_id=3975,
season=2025)
```

STATS.NBA.COM (128 WRAPPERS)

One stem, one host; `league_id` picks "00" NBA · "20" G-League · "15" Summer League. Uses `curl_cffi impersonate="chrome"` (plain `requests` stalls on the TLS fingerprint block).

```
from sportsdataverse.nba import nba_stats
nba_stats.nba_stats_leaguedashplayerstats(league_id
nba_stats.nba_stats_playercareerstats(player_id="25
# dict of frames
nba_stats.nba_stats_playbyplayv3(game_id="0022400001
nba_stats.nba_stats_boxscoretraditionalv3(game_id=...)
# return_parsed=False → raw resultSets
envelope
parse_nba_stats_result_sets(raw,
result_set=None)
```

Families: `leaguedash*` · `playerdash*` · `teamdash*` · `boxscore*v3` · `playbyplayv3` · `shotchartdetail` · `synergyplaytypes` · `leaguegamefinder` · `gamerotation` · `video*` · `draftcombine*` · `commonallplayers` · `scoreboardv3`.

Raw JSON read-through store: `SDV_PY_NBA_RAW_JSON_DIR` (per-endpoint `_DIR_{ENDPOINT}`), `_READONLY=1` = fully offline. Season dirs use the END-year convention (2024 = 2023-24).

NBA RELEASE LOADERS

ESPN-derived: `load_nba_pbp schedule team_boxscore player_boxscore rosters game_rosters officials standings player_core player_season_stats team_season_stats shots draft player_impact`. stats.nba.com-derived: `load_nba_stats_pbp(_v3) possessions(_v3) lineups(_v3) shots schedules rosters game_rosters game_lineups player_boxscores team_boxscores player_game_logs player_season_stats team_season_stats standings officials coaches`. Crosswalks: `load_nba_player_crosswalk team_crosswalk schedule_crosswalk`.

POSSESSION ENGINE (PBPSTATS-STYLE)

```
from sportsdataverse.nba.nba_possessions
import nba_possessions
poss = nba_possessions("0022400001",
league_id="00")
compile_nba_season(2025, season_type="Regular
Season")
from sportsdataverse.nba.nba_rapm import
nba_rapm, nba_rapm_from_games
nba_rapm(poss)                  # ridge, CV
over alphas
nba_adj_rapm(poss, prior)       # Bayesian
prior-shrunk
nba_la_rapm(...) nba_decay_rapm(...)
nba_four_factor_rapm(...)
nba_matchup_drapm("2024-25")    # playtype-
matchup dRAPM
```

On-court: `nba_lineups.nba_on_court` · `nba_enhanced_pbp` · `nba_play_context(game_id, transition_seconds=6.0)` (CTG-style start types) · `nba_playtype.nba_playtype_ratings` · `nba_shot_zones`. G-League: `nbagl_possessions nbagl_enhanced_pbp nbagl_on_court nbagl_rapm_from_games`.

NBA PLAYER & TEAM VALUE

```
nba_shot_value(player_ids, season="2024-25")
# xPts, shooter talent
nba_bpm(player_logs, team_logs, positions)
nba_spm(box_features, coefficients)
nba_war(ratings, poss, replacement_level=...,
pts_per_win=...)
nba_darko(panel, ages)
# Kalman DPM
nba_team_ratings(seasons=2025, as_of_date=...)
nba_game_predict.nba_predict_games(games,
ratings)
nba_game_predict.nba_in_game_win_prob(pbp,
0.58)
```

Also `nba_expected_turnovers` · `nba_foul_drawing` · `nba_clutch` · `nba_tracking_value` · `nba_player_props` · `nba_draft_model` · `nba_rookie_projection` · `nba_aging_curve` · `nba_availability`.

Bundled models

`nba_in_game_wp.ubj` · `nba_aging_curve` · `nba_draft_value` · `nba_rookie_projection` · `nba_availability` (+ `wnba_*` twins).

External oracles

`load_darko_dpm(path)` · `load_epm` · `load_lebron_daily/_season` · `load_rapm_ryan_davis` · `load_dunks_threes_stats` · `load_draft_outcomes` (`nba_oracle_data`).

WNBA · SPORTSDATAVERSE.WNBA

SAME SHAPE, WNBA HOST

```
from sportsdataverse.wnba import *
espn_wnba_schedule(dates=2025);
espn_wnba_pbp(game_id)
espn_wnba_team_roster(team_id=5);
espn_wnba_player_stats(...)
from sportsdataverse.wnba import wnba_stats
# 111 wrappers, LeagueID=10
wnba_stats.wnba_stats_leaguedashplayerstats()
wnba_stats.wnba_stats_playbyplayv3(game_id="1022500
```

Engine: `wnba_engine.wnba_possessions` · `wnba_enhanced_pbp` · `wnba_on_court` · `wnba_play_context`. Models: `wnba_team_ratings` · `wnba_game_predict.wnba_predict_games` · `wnba_in_game_win_prob` · `wnba_shot_value` · `wnba_playtype_impact` · `wnba_tracking_value` · `wnba_draft_model` · `wnba_rookie_projection` · `wnba_aging_curve` · `wnba_career_trajectory` · `wnba_availability` · `wnba_clutch` · `wnba_player_props`.

Loaders: `load_wnba_pbp schedule team_boxscore player_boxscore rosters game_rosters officials standings player_core player_season_stats team_season_stats shots draft player_impact` + `load_wnba_stats_*` (pbp possessions lineups shots schedules rosters game_rosters game_lineups player/team boxscores player_game_logs player/team season_stats standings officials coaches draft) + crosswalks.

CROSSWALKS (LEAGUE ↔ COLLEGE)

WNBA↔NBA and WBB↔MBB identity crosswalks: `load_wnba_player_crosswalk` · `load_mbb_player_crosswalk` · `load_wbb_team_crosswalk` · `load_wbb_schedule_crosswalk` · `_common_crosswalk_basketball`. ESPN↔stats.ncaa.org: `mbb_ncaa_espn_crosswalk`.

COLLEGE · SPORTSDATAVERSE.MBB / .WBB

ESPN + STATS.NCAA.ORG

```
from sportsdataverse.mbb import *
espn_mbb_schedule(dates=2025);
espn_mbb_pbp(game_id)
espn_mbb_teams();
espn_mbb_game_rosters(game_id)
# bigballR / wbigballR port — 16 fns per
league
ncaa_mbb_team_schedule(team="Duke",
season="2024-25")
pbp = ncaa_mbb_game_pbp(game_id)
ncaa_mbb_lineups(pbp);
ncaa_mbb_possessions(pbp)
ncaa_mbb_on_off(...);
ncaa_mbb_player_combos(...)
ncaa_mbb_shot_locations(...);
ncaa_mbb_box_scores(...)
```

Also `ncaa_mbb_date_games team_ids team_roster team_stats player_stats player_lineups play_by_play join_pbp_shots`; `ncaa_wbb_*` mirrors (WBB is halves before 2016). Engine: `mbb_ncaa_*` (fetch → pbp → lineups → stints → possessions → strength, with stint self-healing); names via `display_name_to_roster_key`. Torvik: `torvik_ratings(year)`, `torvik_team_factors`; WBB: `bart_wbb_ratings`.

COLLEGE MODELS & PROJECTIONS

```
r = mbb_team_ratings(seasons=2025)        #
league="womens" for WBB
mbb_predict_games(games, r); mbb_season_sim(r,
remaining)
mbb_bracketology(2025); mbb_bracket_sim(field,
r)
mbb_in_game_win_prob(pbp,
pregame_home_prob=0.6)
mbb_box_bpm(2025); mbb_archetypes(2025)
mbb_shot_quality(shots);
mbb_shooter_talent(scored)
mbb_strength_of_schedule([2025])
mbb_transfer_projection(2025);
mbb_recruiting_projection(2025)
mbb_draft_projection(2025)
```

Plus `mbb_rapm` · `solve_rapm_league` (D-I-wide, released as `ncaa_mbb/wbb_rapm`) · `mbb_lineup_stats` · `mbb_luck` · `mbb_positions` · `mbb_shot_selection` · `wbb_*` twins. Bundled: `mbb/wbb_in_game_wp.ubj`, `_box_bpm`, `_archetypes`, `_draft`, `_recruiting`, `_transfer` (.json).

Loaders

`load_mbb_pbp schedule team_boxscore player_boxscore rosters game_rosters officials standings player_core player_season_stats team_season_stats shots ratings player_value` + crosswalks; `load_wbb_*` identical. WP enriched in place in the released pbp.

HOCKEY · BASEBALL · SOCCER · CRICKET · ODDS :: AND THE WIDER SPORTSDATAVERSE

R siblings: fastRhockey · baseballr · softballR · oddsapiR · sportsdataversedata

HOCKEY · NHL · PWHL · HOCKEY.*

NHL — FIVE LIVE APIS

```
from sportsdataverse.nhl import *
nhl_web_schedule("2025-01-15")
# api-web.nhle.com
plays = nhl_web_pbp(2024020500)
# parsed by default
nhl_edge_skater_detail(8478402,
season=20242025)  # EDGE tracking
nhl_stats_rest_skater_report(...)
# api.nhle.com/stats/rest
nhl_records_awards()
# records.nhl.com
espn_nhl_pbp(game_id=401559000);
espn_nhl_schedule(dates=2025)
```

Parsers: 19 api-web (`parse_nhl_web_pbp` …), 4 EDGE families, generic stats-REST / records. Loaders: `load_nhl_pbp schedule games rosters player_box team_box goalie_box skater_box shifts scoring penalties shootout linescore officials three_stars scratches shots_by_period`.

NHL analytics

```
nhl_xg(pbp)                          # shot xG
(bundled model)
nhl_expected_assists(pbp);
nhl_faceoff_value(pbp)
nhl_gsax.nhl_goalie_gsax(pbp, shifts)
nhl_rapm.nhl_skater_rapm(pbp, shifts);
nhl_war.nhl_skater_war(...)
nhl_unit_ratings(pbp, shifts);
nhl_team_ratings(seasons=2025)
nhl_in_game_win_prob(pbp,
pregame_home_prob=0.55)
```

Also `nhl_penalty_value` · `nhl_zone_transitions` · `nhl_special_teams` · `nhl_edge_value` · `nhl_player_props`.

PWHL + 19 HOCKEYTECH LEAGUES

```
import sportsdataverse as sdv
sdv.pwhl_schedule(season=2025);
sdv.pwhl_pbp(game_id)
sdv.pwhl_game_shifts(game_id);
sdv.pwhl_game_corsi(game_id)
sdv.echl_schedule(season=2026);
sdv.ushl_standings(season=2025)
```

One registry-driven core (`hockeytech.LEAGUES`): 13 callables per league — `<lg>_schedule _pbp _standings _teams _team_roster _player_stats _leaders _game_summary _game_shifts _player_toi _game_corsi _season_id` + `most_recent_<lg>_season`. Leagues: `pwhl ahl ohl whl qmjhl echl sphl chl ushl bchl ajhl sjhl ojhl cchl gojhl mhl nojhl vijhl kijhl mjhl`. PWHL extras: `pwhl_goalie_gsax` · `pwhl_skater_rapm` · `pwhl_in_game_win_prob` · `pwhl_team_ratings`. College hockey: `espn_mch_*` / `espn_wch_*`, `college_hockey_ratings`.

BASEBALL · MLB · BASEBALL

MLB STATS API + ESPN

```
from sportsdataverse.mlb import *
mlb_schedule(date="2025-07-04")         #
statsapi.mlb.com
mlb_boxscore(game_pk=745000);
mlb_play_by_play(game_pk)
espn_mlb_pbp(game_id=401570000);
espn_mlb_schedule(dates=2025)
```

~80 Stats-API wrappers (`mlb_*`). `load_mlb_*`: `batter_projection catcher_framing command_plus expected_hr expected_stats oaa re24_matrix stuff_plus we_table wpa xera`.

STATCAST — BASEBALL SAVANT (43 ENDPOINTS)

```
mlb_statcast_search("2025-04-01", "2025-04-
07",
                    player_type="pitcher",
pitch_type="FF")
mlb_statcast_leaderboard_expected_stats(year=2025)
# 37 boards
mlb_statcast_gamefeed(game_pk=745000)
mlb_statcast_player(660271,
section="statcast")
```

Families `search` (chunked; `_minors`, `_wbc`) · `leaderboard` · `gamefeed` · `player`; friendly filters (`season pitch_type at_bat_result`) translate to Savant params.

MLB ANALYTICS

```
mlb_stuff_plus(pitches);
mlb_command_plus(pitches)
mlb_expected_stats("2025-04-01", "2025-09-30")
mlb_catcher_framing(pitches);
mlb_fielding_oaa(bip)
mlb_run_expectancy_matrix(seasons=2024);
mlb_win_expectancy(pbp, results)
mlb_pitch_classify(pitches);
mlb_batter_projection(2026)
```

Also `mlb_expected_home_runs` · `mlb_run_values` · `mlb_swing_decision` · `mlb_pitch_sequencing` · `mlb_pitch_fatigue` · `mlb_stolen_base` · `mlb_baserunning` · `mlb_umpire_zone` · `mlb_team_projection` · `mlb_prop_projection`.

COLLEGE BASEBALL & SOFTBALL

```
from sportsdataverse.baseball.college_baseball
import *
espn_college_baseball_scoreboard(dates=2025)
parse_college_baseball_ncaa_pbp(html)   #
stats.ncaa.org
college_baseball_re24(...);
college_baseball_wpa(...)  # + softball
```

SOCCER · CRICKET · ODDS · WIN-EXPECTANCY

SOCCER (ESPN, 12 LEAGUES)

```
from sportsdataverse.soccer.epl import
espn_epl_scoreboard
espn_epl_scoreboard(dates=20250315)
espn_soccer_team_schedule("eng.1",
team_id=359, season=2025)
```

Per-league modules `epl mls nwsl laliga bundesliga seriea ligue1 ligamx ucl uel wc wwc` (full ESPN surface each) + league-parameterized `espn_soccer_*`.

CRICKET (ESPN + BUNDLED WP)

```
from sportsdataverse.cricket import *
espn_cricket_scoreboard(league="icc",
dates=2025)
state = cricket_match_state(pbp)
cricket_win_probability(state);
cricket_wpa(state_wp)
```

ODDS — THE ODDS API (`ODDS_API_KEY`)

```
from sportsdataverse.odds import *
toa_sports(); toa_usage()
toa_sports_odds("basketball_nba",
regions="us", markets="h2h,spreads")
toa_event_odds(sport, event_id,
markets="player_points")
toa_sports_odds_history(sport, date="2024-11-
29T22:45:00Z")
toa_sports_scores(...); toa_sports_events(...)
```

Historical lines also ride the release loaders (`load_cfb_betting_lines`, `load_nfl_schedule`).

WIN-EXPECTANCY LAB (`WEXP`)

```
from sportsdataverse import wexp
oracle = wexp.build_cfb_market_oracle(...)  #
or _nfl_
pred =
wexp.build_predictor(wexp.VariantConfig(...))
wexp.backtest.run_backtest(oracle, pred,
model_id="ridge")
```

Ridge-margin, Elo, Glickman–Stern predictors vs market closing lines; Brier + calibration scoring. Football-only today.

THE WIDER SPORTSDATAVERSE

R PACKAGES (SPORTSDATAVERSE.ORG)

Package	Domain	sdv-py mirror
cfbfastR	college football	`cfb` · cfbfastR-data loaders
hoopR	NBA · men's college	`nba mbb`
wehoop	WNBA · women's college	`wnba wbb`
baseballr	MLB · Statcast · college	`mlb baseball`
fastRhockey	NHL · PWHL · HockeyTech	`nhl pwhl hockey`
softballR	college softball	`baseball.college_softball`
recruitR	CFB recruiting	`cfb.sports247_*` `on3_*`
oddsapiR	The Odds API	`odds`
nflseedR · cfbseedR	season sims / seeding	`nfl_simulations` `cfb_simulations`
sportsdataversedata	release publishing	`release`

Related: **nflfastR** / **nflreadr** (nflverse) · **cfbplotR** · **sportyR** / **sportypy** · **sportsdataverse-js**.

DATA REPOS BEHIND THE LOADERS

Each sport has a `*-raw` (scraped JSON, committed per game) and a `*-data` (tidy parquet on GitHub releases) repo under `github.com/sportsdataverse`. Loaders read the release asset; `sportsdataverse.release` writes them.

```
sportsdataverse_save(df, "pbp_2025", "cfb",
release_tag="cfb_pbp")
```

CONVENTIONS TO REMEMBER

- polars 1.x API everywhere; `return_as_pandas=True` converts at the edge.
- IDs are join keys — one dtype per id; never paper over with a float→str cast.
- Live tests gate on `SDV_PY_LIVE_TESTS=1`; stats.nba.com on `SDV_PY_NBA_STATS_LIVE=1`.
- Reference docs + returns tables are generated from endpoint YAML (`tools/codegen`).

MODELS :: THE CROSS-SPORT MODELING GRID, BUNDLED ARTIFACTS, VALIDATION GATES

Every row below is code that ships in the wheel — no notebooks, no downloads (except NFL xYAC).

ONE SPINE, EVERY SPORT

Each league implements the same five layers. Names are the public entry points; `·` separates alternatives, — means not built for that league.

League	Team ratings	Game predict	Season / bracket sim	In-game WP	Player value
CFB	`cfb_ratings`	`cfb_predict_games`	`cfb_simulations` · `cfb_season_odds` · `cfb_playoff_seeds`	`wp_spread` / `wp_naive` via `CFBPlayProcess`	`cfb_adjusted_epa` · `cfb_roster_talent` · `cfb_draft_projection`
NFL	`nfl_ratings`	`nfl_market` · `nfl_gamescript`	`nfl_simulations` (+ `nfl_standings_calc`)	`enrich_nfl_pbp` → `wp`, `vegas_wp`	`nfl_projection` · `nfl_kicker_rating` · `nfl_line_grades`
NBA / G-Lg	`nba_team_ratings`	`nba_game_predict.nba_predict_games`	— (schedule-driven via predict)	`nba_in_game_win_prob`	`nba_rapm` · `nba_adj_rapm` · `nba_bpm` · `nba_spm` · `nba_war` · `nba_darko` · `nba_shot_value`
WNBA	`wnba_team_ratings`	`wnba_game_predict.wnba_predict_games`	—	`wnba_in_game_win_prob`	`wnba_engine` RAPM · `wnba_shot_value` · `wnba_playtype_impact`
MBB / WBB	`mbb_team_ratings` · `torvik_ratings`	`mbb_predict_games`	`mbb_season_sim` · `mbb_bracketology` · `mbb_bracket_sim`	`mbb_in_game_win_prob`	`mbb_box_bpm` · `mbb_rapm` · `mbb_shot_quality` · `mbb_archetypes` (+ `wbb_*`)
NHL	`nhl_team_ratings`	`nhl_market.nhl_predict_games`	—	`nhl_in_game_win_prob`	`nhl_xg` · `nhl_skater_rapm` · `nhl_skater_war` · `nhl_goalie_gsax` · `nhl_unit_ratings`
PWHL	`pwhl_team_ratings`	`pwhl_market.pwhl_predict_games`	—	`pwhl_in_game_win_prob`	`pwhl_skater_rapm` · `pwhl_skater_war` · `pwhl_goalie_gsax`
MLB	`mlb_team_projection`	— (projection-driven)	—	`mlb_win_expectancy`	`mlb_stuff_plus` · `mlb_command_plus` · `mlb_expected_stats` · `mlb_catcher_framing` · `mlb_fielding_oaa`
Others	college hockey `college_hockey_ratings` · cricket `cricket_win_probability` / `cricket_wpa` · spring football reuses the NFL engine (`build_spring_football_pbp` → `enrich_spring_football_pbp`)				

Shared shape: ratings take `seasons=` + `as_of_date=` (leak-safe, exclusive cut); predictors take `(games, ratings)`; sims take `(ratings | games, …, n_sims / simulations)`; in-game WP takes `(pbp, pregame_home_prob)`.

DRAFT, PROJECTION & ROSTER MODELS

Family	Entry points
Draft value / rookies	`nba_draft_model` · `nba_rookie_projection` · `wnba_draft_model` · `nfl_draft_model` · `mbb_draft_projection` · `cfb_draft_projection`
Aging & availability	`nba_aging_curve` · `nba_availability` · `wnba_career_trajectory` · `nfl_availability`
Recruiting / transfers	`cfb_recruiting_projection` · `cfb_transfer_impact` · `mbb_recruiting_projection` · `mbb_transfer_projection`
Props & markets	`nba_player_props` · `wnba_player_props` · `nfl_player_props` · `nhl_player_props` · `pwhl_player_props` · `mlb_prop_projection` · `nfl_market` · `nhl_market`

BUNDLED ARTIFACT REGISTRY

Shipped as package data (`importlib.resources`); XGBoost `.ubj` unless noted. Feature counts from the model cards.

Dir	Artifacts
`cfb/models`	`ep_model` 8f · `wp_naive` 12f · `wp_spread` 13f · `cfb_cp_model` · `qbr_model` 10f · `fg_model` 5f · `fd_model` · `two_pt_model` 4f · `xpass_model` 7f · `punt_distribution.parquet` · `cfb_field_position_ep.parquet`
`nfl/models`	`ep_model` 18f · `wp_spread` 12f · `wp_naive` 11f · `cp_model` 18f · `fg_model` · `qbr_model` · `two_pt_model` · `xpass_model` · `nfl_playcall` · `punt_data.parquet`; xYAC (76-class) fetched on first use
`nba/models`	`nba_in_game_wp` · `wnba_in_game_wp` · `{nba,wnba}_aging_curve` · `_draft_value` · `_rookie_projection` · `_availability` (.json)
`mbb/models`	`{mbb,wbb}_in_game_wp` · `_box_bpm` · `_archetypes` · `_draft` · `_recruiting` · `_transfer` (.json)
`nhl/models`	`nhl_in_game_wp.json` (+ xG coefficients in `nhl_xg`)
`mlb/models`	`mlb_stuff_plus` · `mlb_command_plus`
`cricket/models`	`cricket_resource_surface.parquet` · `cricket_winprob_calibration.parquet`

Every model's fitting script and era cuts live in the `*-data` producer repo that trains it (cfbfastR-cfb-data, nfl-data, hoopR-nba-data …); constants in `*_constants.py` / `model_vars.py` cite them.

RULE-ERA & FEATURE CONVENTIONS

- NFL era cuts `ERA_SEASON_CUTS` = 2001 / 2005 / 2013 / 2017; CFB models are per rule era.
- Kickoff / PAT feature substitution: touchback yardline 80 pre-2016, 75 from 2016; `down→1`, `ydstogo→10`.
- Spread WP: `spread_time` decays with exponent −4.0 (`SPREAD_TIME_DECAY_EXPONENT`).
- Outputs are `Float64`; models emit float32 — cast at the boundary.

VALIDATION & ORACLE GATES

Every model ships with a gate against an external oracle captured as a committed fixture. Gates are measured from observed values and never lowered.

Metric	Used for
Brier · log-loss · calibration table	win probabilities, in-game WP, sims (`brier_score`, `calibration_table` in `*_prediction_constants` / `wexp`)
MAE vs market	spreads / totals (`nfl_market`, `cfb_season_odds`, wexp market oracles)
Spearman ρ	ratings vs Torvik / KenPom / ESPN FPI / MoneyPuck; player value vs EPM, DARKO, RAPM, LEBRON
Correlation floors	parity with the R producers (nflfastR EP 0.996, CFB ratings ρ 0.97)

Oracle loaders

`load_darko_dpm` · `load_epm` · `load_lebron_daily` · `load_rapm_ryan_davis` · `load_dunks_threes_stats` · `load_draft_outcomes` · `torvik_ratings` · `bart_wbb_ratings` · `load_cfb_fpi_weekly` · `load_cfb_power_index` · `wexp.build_cfb_market_oracle` / `_nfl_`.

LEAK-SAFETY CONTRACT

- `as_of_date` / `through_week` are **exclusive**: the cut game is never in the training window.
- Every lag, cumulative sum or shift is `.over("game_id")` (or season) — no cross-game bleed when frames are concatenated.
- Ratings for a date use only games with `game_date < as_of_date`; sims draw from those ratings.
- Fitted constants (HFA, σ, λ) name the fitting script; assert the **output** changed, not that the code ran.

WIN-EXPECTANCY LAB (WEXP)

```
from sportsdataverse import wexp
o  = wexp.build_cfb_market_oracle(...)
p  =
wexp.build_predictor(wexp.VariantConfig(...))
wexp.backtest.run_backtest(o, p,
model_id="ridge")
```

Ridge-margin, Elo, Glickman–Stern vs closing lines; vintage store for reproducible walk-forward runs.

RECIPES :: END-TO-END PIPELINES, POLARS IDIOMS, GOTCHAS

Copy-paste starting points — each runs as written given the named inputs.

CFB · RAW JSON → ENRICHED PBP (OFFLINE)

```
from sportsdataverse.cfb import CFBPlayProcess
proc = CFBPlayProcess(gameId=401628334,
    path_to_json="raw/2024/",
    odds_override={"gameSpread": -6.5,
"overUnder": 55.5,
                   "homeFavorite": True,
                   "gameSpreadAvailable":
True})
proc.cfb_pbp_disk()
pbp = proc.run_processing_pipeline()   #
EPA/WPA, no network
box = proc.create_box_score()
```

The override injects persisted odds so the rebuild never falls back to the live odds endpoint (provenance: `pbp_txt["odds_source"] == "injected"`).

NFL · LOAD → ENRICH → 4TH-DOWN TABLE

```
import sportsdataverse.nfl as nfl
from sportsdataverse.nfl.ep_wp import
enrich_nfl_pbp
from sportsdataverse.nfl.nfl_fourth_down
import get_4th_down_probs
pbp = nfl.load_pbp([2024])
df  = enrich_nfl_pbp(pbp)          #
ep→epa→wp→wpa→cp→xyac
fourth = get_4th_down_probs(
    df.filter(pl.col("down") == 4))
```

`get_4th_down_probs` adds go / FG / punt WP + the recommendation columns (nfl4th-style, fully offline).

SPRING FOOTBALL ON THE NFL ENGINE

```
from sportsdataverse.football.ufl import
espn_ufl_summary
from
sportsdataverse.football.spring_football_ep_wp
import *
raw = espn_ufl_summary(game_id)
pbp = build_spring_football_pbp(raw,
league="ufl")
pbp = enrich_spring_football_pbp(pbp)
```

NBA · POSSESSIONS → RAPM → WAR

```
from sportsdataverse.nba.nba_season_compile
import compile_nba_season
from sportsdataverse.nba.nba_rapm import
nba_rapm_from_games
from sportsdataverse.nba import nba_war
poss  = compile_nba_season(2025)   # end-year:
2024-25
rapm  =
nba_rapm_from_games(poss["game_id"].unique())
value = nba_war(rapm, poss,
                replacement_level=-2.0,
pts_per_win=30.0)
```

Set `SDV_PY_NBA_RAW_JSON_DIR` first and the compile reads disk before network; add `_READONLY=1` for a fully offline, reproducible build (a miss raises `RawStoreMissError`).

MBB · STATS.NCAA.ORG PBP → LINEUPS → ON/OFF

```
from sportsdataverse.mbb import
(ncaa_mbb_game_pbp,
    ncaa_mbb_lineups, ncaa_mbb_possessions,
ncaa_mbb_on_off)
pbp  = ncaa_mbb_game_pbp(6305900)
lu   = ncaa_mbb_lineups(pbp)
poss = ncaa_mbb_possessions(pbp)
oo   = ncaa_mbb_on_off("PlayerName", lu)
```

WBB mirrors every call (`ncaa_wbb_*`); pre-2016 WBB pbp is halves, not quarters — 0 rows for a quarters filter is data, not an error.

MLB · STATCAST SEARCH → STUFF+

```
from sportsdataverse.mlb import
mlb_statcast_search, mlb_stuff_plus
pitches = mlb_statcast_search("2025-04-01",
"2025-04-30",
    player_type="pitcher", pitch_type="FF")
stuff = mlb_stuff_plus(pitches, level="pitch")
```

POLARS IDIOMS FOR PBP WORK

```
# EPA per drive, success rate — group, don't
loop
pbp.group_by("game_id", "drive_id",
maintain_order=True) \
   .agg(pl.col("epa").sum(), (pl.col("epa") >
0).mean()
         .alias("success_rate"))

# lags NEVER cross a game boundary
pbp.with_columns(pl.col("score_diff").shift(1)

.over("game_id").alias("prev_diff"))

# explicit bool masks (house style)
pbp.filter((pl.col("pass") == True) &
           (pl.col("garbage_time") == False))

# no lookaround in Rust regex — case-toggle
instead
pl.col("text").str.extract(
    r"(?i)sacked by(?-i: ([A-Z][\w'\.\-]+))",
1)

# pandas only at the very edge
df = any_loader(..., return_as_pandas=True)
```

CROSS-LEAGUE JOINS

```
# pin ONE dtype per id, assert before joining
a =
a.with_columns(pl.col("athlete_id").cast(pl.Int64))
assert a.schema["athlete_id"] ==
b.schema["athlete_id"]
a.join(b, on="athlete_id", how="left")
# ESPN ⇔ stats.nba.com identity
xwalk = sdv.load_nba_player_crosswalk(seasons=
[2025])
```

Never stringify a float id (`"123.0" ≠ "123"`); if you must go Utf8, cast through Int64 first.

GOTCHA STRIP

Trap	Rule
Season year	NBA/WNBA-stats dirs use the **end year** (2024 = 2023-24); ESPN loaders use kickoff/tip year; NBA span params want `"2023-24"`
stats.nba.com	TLS-fingerprint block → silent hang, not an error; the runtime uses `curl_cffi impersonate="chrome"`; run live only from residential IPs
Game ids	stats.nba.com `game_id` is a zero-padded **string** (`"0022400001"`) — never int-cast it
WBB history	halves before 2016 — quarter-keyed logic silently empties six seasons
Cache	`@cached_loader` keys on (name, args) not URL — `clear_cache()` after changing loader targets
Empty ≠ error	parsers return zero-row frames with the documented schema; check `df.is_empty()`, don't try/except
ESPN rate	Core v2 403s under aggressive parallelism — keep scrapes low-concurrency

TEST GATES

```
uv run pytest                          # offline
suite
SDV_PY_LIVE_TESTS=1 uv run pytest   # live
APIs
SDV_PY_NBA_STATS_LIVE=1 ...         #
stats.nba.com only,
                                       #
residential IP
```

CONTRIBUTING A NEW ENDPOINT

- Wrapper metadata lives in `tools/codegen/endpoints/*.yaml`; run `uv run python tools/codegen/generate.py`, never hand-edit generated files.
- Returns-table descriptions go in `manual_column_descriptions.yaml` (schemas/** is clobbered on re-capture).
- Validate parsers against **real captured fixtures**, never synthetic ones.
- New modules are fully typed and join the mypy ratchet in `pyproject.toml`.
- `generate.py --check` is the CI drift gate — regenerate before every push.

sportsdataverse-py :: CHEAT SHEET

The Python sister to the SportsDataverse R packages (wehoop, hoopR, cfbfastR, baseballr, fastRhockey, nflfastR). One `pip install` gives polars-first, tidy access to play-by-play, box scores, schedules, rosters and odds across 15+ sports, plus bundled EP / WP / rating / projection models and release-dataset loaders. Page 1 of 4: getting started, coverage map, the ESPN cross-league layer, utilities.

v0.0.75

Python 3.9 – 3.14 · polars 1.x
py.sportsdataverse.org

GET STARTED

```
pip install sportsdataverse
pip install "sportsdataverse[all]"  # + models, curl_cffi, tests
uv add sportsdataverse

import sportsdataverse as sdv
from sportsdataverse.cfb import load_cfb_pbp
```

Every public callable is also on the flat `sdv.*` namespace (4,600+ names). Sub-packages: `cfb nfl nba nbagl wnba mbb wbb nhl pwhl hockey hockeytech mlb baseball soccer cricket football odds wexp`.

RETURN CONVENTIONS

Flag	Effect
default	`polars.DataFrame`, snake_case columns
`return_as_pandas=True`	pandas frame (same data)
`raw=True`	scrapers: untouched JSON `dict`
`return_parsed=False`	ESPN wrappers: tidy frame (default) → raw `dict`
multi-table	`dict[str, pl.DataFrame]` keyed by section
empty / malformed	zero-row frame with documented schema — never raises

THREE SURFACES

1 · Release-dataset loaders

`load_<sport>_<dataset>(seasons, return_as_pandas=False)` reads parquet from the SportsDataverse GitHub releases (the `*-data` repos). Cached; zero scraping.

```
pbp = load_cfb_pbp(seasons=[2023, 2024])
sch =
sdv.load_nba_schedule(seasons=range(2020, 2025))
```

2 · Live API wrappers

`espn_<league>_*`, `nba_stats_*`, `wnba_stats_*`, `nhl_web_*`, `nhl_edge_*`, `mlb_*`, `mlb_statcast_*`, `<hockeytech-league>_*`, `ncaa_mbb_*`, `toa_*` (odds).

3 · Processing + models

PBP processors (`CFBPlayProcess`, `NFLPlayProcess`), `enrich_nfl_pbp`, `nba_possessions`, ratings, simulations, projections — see pages 2–4.

COVERAGE MAP

Sport	Sub-package	Live sources	R sibling
NFL	`nfl`	ESPN · api.nfl.com · PFF	nflfastR · nflreadr
CFB	`cfb`	ESPN · stats.ncaa.org · Fox · Yahoo · 247 · On3 · PFF	cfbfastR · recruitR
NBA · G-League · Summer	`nba` `nbagl`	ESPN · stats.nba.com · Fox	hoopR
WNBA	`wnba`	ESPN · stats.wnba.com	wehoop
Men's college bb	`mbb`	ESPN · stats.ncaa.org · Torvik	hoopR · bigballR
Women's college bb	`wbb`	ESPN · stats.ncaa.org · Bart	wehoop · wbigballR
NHL	`nhl`	ESPN · api-web.nhle.com · EDGE · stats REST · records	fastRhockey
PWHL + 19 minor/junior	`pwhl` `hockey.*`	HockeyTech / LeagueStat	fastRhockey
College hockey M/W	`hockey.mch` `.wch`	ESPN	—
MLB	`mlb`	ESPN · statsapi.mlb.com · Baseball Savant	baseballr
College baseball / softball	`baseball`	ESPN · stats.ncaa.org	baseballr · softballR
Soccer (12 leagues)	`soccer.*`	ESPN	—
Cricket	`cricket`	ESPN + bundled WP	—
UFL · XFL · CFL · AAF	`football.*`	ESPN + spring EP/WP	—
Odds & lines	`odds`	The Odds API	oddsapiR
Win-expectancy lab	`wexp`	CFB / NFL market oracles	—

Release data repos: `cfbfastR-data`, `nfl-data`, `hoopR-nba-data`, `hoopR-mbb-data`, `wehoop-wnba-data`, `wehoop-wbb-data`, `fastRhockey-*-data`, `baseballr-data`, `ncaa-mbb/wbb-data`, `*-stats-data`.

ESPN CROSS-LEAGUE LAYER

One core (`_common_espn`) × N thin league modules. Every league gets the same **121 endpoint short names** as `espn_<prefix>_<short>` (≈125 wrappers per league; 800+ total across nba wnba nfl cfb mbb wbb nhl mlb, plus soccer, cricket, college baseball/hockey and spring football).

```
from sportsdataverse.nba import
espn_nba_team_roster
df  = espn_nba_team_roster(team_id=13)
# polars (default)
raw = espn_nba_team_roster(team_id=13,

return_parsed=False)   # raw dict
```

Family	Short names
Schedule	`schedule calendar scoreboard games season_weeks season_week_games`
Teams	`teams team team_roster team_schedule team_record team_injuries team_depthcharts franchise venue`
Games	`summary game game_rosters game_plays play_participants game_odds game_officials game_predictor game_probabilities game_propbets`
Players	`player_stats player_gamelog player_splits player_career_stats player_bio player_contracts player_vs_player`
League	`standings leaders news injuries transactions draft awards conferences futures season_qbr`
NCAA extras	`rankings fpi recruits recruiting_rankings` (mbb wbb cfb …)

Parser layer

`parse_summary(payload, section=None)` → 21 sub-frames: `boxscore_player boxscore_team plays winprobability leaders game_info officials header season_series against_the_spread standings broadcasts format pickcenter odds article injuries news drives drive_plays scoring_plays`. Generic fall-throughs: `parse_single_entity`, `parse_items`.

Full PBP processors (tidy, per game)

```
espn_nba_pbp(401584793)    espn_wnba_pbp(...)
espn_mbb_pbp(...)          espn_wbb_pbp(...)
espn_nhl_pbp(...)          espn_mlb_pbp(...)
CFBPlayProcess(gameId=401628334)   # page 2
NFLPlayProcess(gameId=401671889)   # page 2
```

DISCOVERY

```
sdv.find_team("Duke", "mbb")
sdv.find_athlete("Caitlin Clark", "wnba",
team="Fever")
sdv.find_event("2024-11-30", "cfb", home="Ohio
State")
sdv.function_count()            # per league
sdv.list_functions("nhl", search="pbp")
```

CLI mirrors it: `sdv find-team Duke --league mbb`, `sdv function-count`, `sdv cache stats|clear|mode`.

CACHE

```
sdv.set_cache_mode("filesystem")  # memory |
filesystem | off
sdv.set_default_ttl(3600)
sdv.cache_stats();  sdv.clear_cache()
# NFL has its own nflreadpy-style config
from sportsdataverse.nfl import update_config
update_config(cache_mode="filesystem",
cache_duration=3600)
```

Env: `SDV_PY_NFL_CACHE` · `SDV_PY_NFL_CACHE_DIR` · `SDV_PY_NFL_CACHE_DURATION`.

HTTP, TIDY & RELEASE HELPERS

`dl_utils.download(url)` — single retrying HTTP chokepoint. `underscore() camelize() kebabize()`, `flatten_json_iterative()`, `key_check()`.

`sportsdataverse.release` — port of the sportsdataversedata R package: `sportsdataverse_upload()`, `sportsdataverse_save()` (parquet / rds / csv.gz), `gh_cli_release_upload()`, `gh_cli_release_assets()`, `gh_cli_release_tags()`.

Errors: `NoESPNDataError` · `SeasonNotFoundError` · `RawStoreMissError`.

ENV VARS WORTH KNOWING

`SDV_PY_LIVE_TESTS=1`	run gated live tests
`SDV_PY_NBA_STATS_LIVE=1`	stats.nba.com tests (residential IP)
`SDV_PY_NBA_RAW_JSON_DIR`	read-through raw JSON store
`SDV_PY_NBA_RAW_JSON_READONLY=1`	offline: miss → error
`SDV_<LEAGUE>_API_KEY`	override a HockeyTech key
`CFBD_API_KEY` · `ODDS_API_KEY`	third-party keys

COLLEGE FOOTBALL · SPORTSDATAVERSE.CFB

LIVE SCRAPERS

```
from sportsdataverse.cfb import *
espn_cfb_schedule(dates=2024, week=1)
espn_cfb_teams();
espn_cfb_calendar(season=2024)
espn_cfb_game_rosters(game_id=401628334)
espn_cfb_team_roster(team_id=52)
espn_cfb_player_stats(athlete_id=4426502,
season=2024)
```

Play-by-play (ESPN, full pipeline)

```
proc = CFBPlayProcess(gameId=401628334)
pbp  = proc.espn_cfb_pbp()          # dict of
frames
pbp  = proc.run_processing_pipeline()  #
EPA/WPA added
box  = proc.create_box_score()
# offline rebuild from on-disk raw JSON
CFBPlayProcess(gameId, path_to_json=raw_dir,
               odds_override=
{...}).cfb_pbp_disk()
```

Per-play participants come from ESPN's `participants[]` (`cfb_play_participants`) → `{type}_player_name` / `_id` + list columns.

Other sources

stats.ncaa.org: `parse_cfb_ncaa_pbp` · `parse_cfb_ncaa_drives` · `_linescore` · `_officials` · `_team_stats` · `_player_stats` (`cfb_ncaa_pbp` / `cfb_ncaa_box`) · `fox_cfb_pbp` · Yahoo ext · `pff_*` (premium) · Recruiting: `sports247_*` (247 RDB), `on3_*`, `espn_cfb_recruits`.

CFB MODELS (BUNDLED)

Artifact	Purpose
`ep_model`	expected points (rule-era)
`wp_naive` · `wp_spread`	win probability ± Vegas spread
`cfb_cp_model`	completion prob · CPOE
`qbr_model`	QBR-style rating
`fg_model` · `two_pt_model`	kick / conversion prob
`fd_model` · `xpass_model`	4th-down go/kick/punt · pass rate
`punt_distribution` · `cfb_field_position_ep`	decision surfaces

CFB ANALYTICS

```
r = cfb_ratings(seasons=2024, as_of_date=...)
# opp-adjusted
cfb_predict_games(games, r)             #
spread / WP / total
cfb_simulations(games, teams,
simulations=10000)
cfb_adjusted_epa(plays);
cfb_adjusted_tempo(seasons=2024)
cfb_advanced_stats(seasons=2024)        #
success, explosiveness…
cfb_standings(...); cfb_playoff_seeds(...);
cfb_resume(...)
```

Also `cfb_fourth_down` · `cfb_two_point` · `cfb_field_position` · `cfb_opponent_adjust` · `cfb_season_odds` · `cfb_returning_production` · `cfb_roster_talent` · `cfb_transfer_impact` · `cfb_draft_projection` · `cfb_recruiting_projection` · `cfb_crosswalk`.

CFB RELEASE LOADERS (`LOAD_CFB_*`)

Group	Datasets
Core	`pbp pbp_r model_pbp schedule rosters game_rosters team_box player_box drives linescores team_info play_participants`
Stats	`passing rushing receiving percentiles team_summaries team_summaries_weekly`
Advanced	`adv_team adv_team_gamelog adv_passing adv_rushing adv_receiving adv_defensive adv_defensive_players adv_drives adv_situational adv_specialists adv_turnover`
Ratings	`ratings ratings_weekly fpi_weekly power_index`
Betting	`betting betting_lines`
Recruiting	`recruits recruiting_proj team_talent returning_production` + `load_recruit_classes`
Crosswalks	`teams_crosswalk rosters_crosswalk schedule_crosswalk`

```
load_cfb_pbp(seasons=[2024])
load_cfb_ratings_weekly(seasons=[2024])
load_cfb_betting_lines(seasons=[2024])
```

NFL · SPORTSDATAVERSE.NFL

NFLREADPY-PARITY LOADERS

24 canonical `load_nfl_*` loaders; inside `sportsdataverse.nfl` they are also exported under nflreadpy's names (`load_pbp`, `load_schedules`, …).

```
import sportsdataverse.nfl as nfl
pbp = nfl.load_pbp([2024])
sch = nfl.load_schedules([2024])
ngs =
nfl.load_nextgen_stats(stat_type="passing")
adv = nfl.load_pfr_advstats(stat_type="pass",

summary_level="season")
qbr = nfl.load_espn_qbr(seasons=[2024])
```

Loaders

`pbp schedule teams players rosters weekly_rosters player_stats team_stats pbp_participation snap_counts depth_charts injuries officials trades contracts draft_picks combine ff_playerids ff_rankings ff_opportunity ftn_charting nextgen_stats pfr_advstats espn_qbr` + `load_nfl_ratings_weekly`

Config: `get_config()` `update_config()` `reset_config()` `clear_cache()` · `datasets.team_abbr_mapping`, `team_abbr_mapping_norelocate`, `player_name_mapping` · `get_current_season()` `get_current_week()`.

LIVE NFL SOURCES

```
espn_nfl_schedule(season=2024, week=1)
espn_nfl_teams();
espn_nfl_game_rosters(401671889)
NFLPlayProcess(gameId=401671889).run_processing_pipeline()
# api.nfl.com "Shield" (11 endpoints, bearer
auth)
nfl_standings(season=2024); nfl_rosters(...);
nfl_weeks(...)
```

Parsers: 11 dedicated `parse_nfl_*`. PFF: `pff_*` (premium API, Clerk session).

EP / WP / EPA / WPA (EP_WP)

Single owner of NFL model application — a faithful port of nflfastR's `helper_add_ep_wp.R`. Construction modules emit a frame; `ep_wp` enriches it.

```
from sportsdataverse.nfl.ep_wp import *
df = enrich_nfl_pbp(pbp)     #
ep→epa→wp→wpa→cp→cpoe→xyac
calculate_expected_points(df);
calculate_epa(df)
calculate_win_probability(df);
calculate_wpa(df)
calculate_completion_probability(df);
calculate_xyac(df)
calculate_xpass(df)
```

Parity vs nflverse: `ep` 0.996 · `epa` 0.994 · `wp` 0.997 · `vegas_wp` 0.998. Every lag is `.over("game_id")`.

Decision surfaces

`nfl_fourth_down`: `get_go_wp()` `get_fg_wp()` `get_punt_wp()` (nfl4th-style, offline). `nfl_playcall` · `nfl_kicker_rating` · `nfl_special_teams` · `nfl_gamescript` · `nfl_series`.

NFL MODELS (BUNDLED .UBJ)

`ep_model` (18 feat) · `wp_spread` (12) · `wp_naive` (11) · `cp_model` (18) · `fg_model` · `qbr_model` · `two_pt_model` · `xpass_model` · `nfl_playcall` · `punt_data.parquet`. xYAC (76-class) downloads on first use from `nfl_model_artifacts`.

NFL RATINGS, SIMS, PROJECTIONS

`nfl_ratings` · `nfl_simulations` (season / playoff sim; nflseedR-style `nfl_standings_calc`) · `nfl_projection` · `nfl_usage_projection` · `nfl_player_props` · `nfl_market` · `nfl_draft_model` · `nfl_availability` · `nfl_line_grades` · `nfl_roster_builder` · `nfl_ngs_tracking`.

SPRING FOOTBALL · UFL / XFL / CFL / AAF

```
from sportsdataverse.football import *
from sportsdataverse.football.ufl import
espn_ufl_scoreboard, espn_ufl_summary
from
sportsdataverse.football.spring_football_ep_wp
import *
pbp =
build_spring_football_pbp(espn_ufl_summary(game_id),
league="ufl")
enrich_spring_football_pbp(pbp)   # EP/WP via
the NFL ep_wp engine
```

Leagues: `football.ufl` · `.xfl` · `.cfl` · `.aaf` — full ESPN short-name surface each (`espn_ufl_scoreboard`, `espn_ufl_team_roster`, `espn_cfl_standings`, …).

Basketball :: NBA · WNBA · G-League · men's & women's college

NBA · G-LEAGUE · SPORTSDATAVERSE.NBA

ESPN

```
from sportsdataverse.nba import *
espn_nba_schedule(dates=2025);
espn_nba_teams()
espn_nba_pbp(game_id=401584793)   # plays,
box, rosters, WP
espn_nba_game_rosters(401584793);
espn_nba_team_roster(13)
espn_nba_player_stats(athlete_id=3975,
season=2025)
```

STATS.NBA.COM (128 WRAPPERS)

One stem, one host; `league_id` picks **"00"** NBA · **"20"** G-League · **"15"** Summer League. Uses `curl_cffi impersonate="chrome"` (plain `requests` stalls on the TLS fingerprint block).

```
from sportsdataverse.nba import nba_stats
nba_stats.nba_stats_leaguedashplayerstats(league_id
nba_stats.nba_stats_playercareerstats(player_id="25
# dict of frames
nba_stats.nba_stats_playbyplayv3(game_id="0022400001")
nba_stats.nba_stats_boxscoretraditionalv3(game_id=.
# return_parsed=False → raw resultSets envelope
parse_nba_stats_result_sets(raw,
result_set=None)
```

Families: `leaguedash*` · `playerdash*` · `teamdash*` · `boxscore*v3` · `playbyplayv3` · `shotchartdetail` · `synergyplaytypes` · `leaguegamefinder` · `gamerotation` · `video*` · `draftcombine*` · `commonallplayers` · `scoreboardv3`.

Raw JSON read-through store: `SDV_PY_NBA_RAW_JSON_DIR` (per-endpoint `_DIR_{ENDPOINT}`), `_READONLY=1` = fully offline. Season dirs use the END-year convention (2024 = 2023-24).

NBA RELEASE LOADERS

ESPN-derived: `load_nba_pbp schedule team_boxscore player_boxscore rosters game_rosters officials standings player_core player_season_stats team_season_stats shots draft player_impact`. stats.nba.com-derived: `load_nba_stats_pbp(_v3) possessions(_v3) lineups(_v3) shots schedules rosters game_rosters game_lineups player_boxscores team_boxscores player_game_logs player_season_stats team_season_stats standings officials coaches`. Crosswalks: `load_nba_player_crosswalk team_crosswalk schedule_crosswalk`.

POSSESSION ENGINE (PBPSTATS-STYLE)

```
from sportsdataverse.nba.nba_possessions
import nba_possessions
poss = nba_possessions("0022400001",
league_id="00")
compile_nba_season(2025, season_type="Regular
Season")
from sportsdataverse.nba.nba_rapm import
nba_rapm, nba_rapm_from_games
nba_rapm(poss)                  # ridge, CV
over alphas
nba_adj_rapm(poss, prior)       # Bayesian
prior-shrunk
nba_la_rapm(...) nba_decay_rapm(...)
nba_four_factor_rapm(...)
nba_matchup_drapm("2024-25")    # playtype-
matchup dRAPM
```

On-court: `nba_lineups.nba_on_court` · `nba_enhanced_pbp` · `nba_play_context(game_id, transition_seconds=6.0)` (CTG-style start types) · `nba_playtype.nba_playtype_ratings` · `nba_shot_zones`. G-League: `nbagl_possessions nbagl_enhanced_pbp nbagl_on_court nbagl_rapm_from_games`.

NBA PLAYER & TEAM VALUE

```
nba_shot_value(player_ids, season="2024-25")
# xPts, shooter talent
nba_bpm(player_logs, team_logs, positions)
nba_spm(box_features, coefficients)
nba_war(ratings, poss, replacement_level=...,
pts_per_win=...)
nba_darko(panel, ages)
# Kalman DPM
nba_team_ratings(seasons=2025, as_of_date=...)
nba_game_predict.nba_predict_games(games,
ratings)
nba_game_predict.nba_in_game_win_prob(pbp,
0.58)
```

Also `nba_expected_turnovers` · `nba_foul_drawing` · `nba_clutch` · `nba_tracking_value` · `nba_player_props` · `nba_draft_model` · `nba_rookie_projection` · `nba_aging_curve` · `nba_availability`.

Bundled models

`nba_in_game_wp.ubj` · `nba_aging_curve` · `nba_draft_value` · `nba_rookie_projection` · `nba_availability` (+ `wnba_*` twins).

External oracles

`load_darko_dpm(path)` · `load_epm` · `load_lebron_daily/_season` · `load_rapm_ryan_davis` · `load_dunks_threes_stats` · `load_draft_outcomes` (`nba_oracle_data`).

WNBA · SPORTSDATAVERSE.WNBA

SAME SHAPE, WNBA HOST

```
from sportsdataverse.wnba import *
espn_wnba_schedule(dates=2025);
espn_wnba_pbp(game_id)
espn_wnba_team_roster(team_id=5);
espn_wnba_player_stats(...)
from sportsdataverse.wnba import wnba_stats
# 111 wrappers, LeagueID=10
wnba_stats.wnba_stats_leaguedashplayerstats()
wnba_stats.wnba_stats_playbyplayv3(game_id="1022500
```

Engine: `wnba_engine.wnba_possessions` · `wnba_enhanced_pbp` · `wnba_on_court` · `wnba_play_context`. Models: `wnba_team_ratings` · `wnba_game_predict.wnba_predict_games` · `wnba_in_game_win_prob` · `wnba_shot_value` · `wnba_playtype_impact` · `wnba_tracking_value` · `wnba_draft_model` · `wnba_rookie_projection` · `wnba_aging_curve` · `wnba_career_trajectory` · `wnba_availability` · `wnba_clutch` · `wnba_player_props`.

Loaders: `load_wnba_pbp schedule team_boxscore player_boxscore rosters game_rosters officials standings player_core player_season_stats team_season_stats shots draft player_impact` + `load_wnba_stats_*` (pbp possessions lineups shots schedules rosters game_rosters game_lineups player/team boxscores player_game_logs player/team season_stats standings officials coaches draft) + crosswalks.

CROSSWALKS (LEAGUE ↔ COLLEGE)

WNBA↔NBA and WBB↔MBB identity crosswalks: `load_wnba_player_crosswalk` · `load_mbb_player_crosswalk` · `load_wbb_team_crosswalk` · `load_wbb_schedule_crosswalk` · `_common_crosswalk_basketball`. ESPN↔stats.ncaa.org: `mbb_ncaa_espn_crosswalk`.

COLLEGE · SPORTSDATAVERSE.MBB / .WBB

ESPN + STATS.NCAA.ORG

```
from sportsdataverse.mbb import *
espn_mbb_schedule(dates=2025);
espn_mbb_pbp(game_id)
espn_mbb_teams();
espn_mbb_game_rosters(game_id)
# bigballR / wbigballR port — 16 fns per
league
ncaa_mbb_team_schedule(team="Duke",
season="2024-25")
pbp = ncaa_mbb_game_pbp(game_id)
ncaa_mbb_lineups(pbp);
ncaa_mbb_possessions(pbp)
ncaa_mbb_on_off(...);
ncaa_mbb_player_combos(...)
ncaa_mbb_shot_locations(...);
ncaa_mbb_box_scores(...)
```

Also `ncaa_mbb_date_games team_ids team_roster team_stats player_stats player_lineups play_by_play join_pbp_shots`; `ncaa_wbb_*` mirrors (WBB is halves before 2016). Engine: `mbb_ncaa_*` (fetch → pbp → lineups → stints → possessions → strength, with stint self-healing); names via `display_name_to_roster_key`. Torvik: `torvik_ratings(year)`, `torvik_team_factors`; WBB: `bart_wbb_ratings`.

COLLEGE MODELS & PROJECTIONS

```
r = mbb_team_ratings(seasons=2025)        #
league="womens" for WBB
mbb_predict_games(games, r); mbb_season_sim(r,
remaining)
mbb_bracketology(2025); mbb_bracket_sim(field,
r)
mbb_in_game_win_prob(pbp,
pregame_home_prob=0.6)
mbb_box_bpm(2025); mbb_archetypes(2025)
mbb_shot_quality(shots);
mbb_shooter_talent(scored)
mbb_strength_of_schedule([2025])
mbb_transfer_projection(2025);
mbb_recruiting_projection(2025)
mbb_draft_projection(2025)
```

Plus `mbb_rapm` · `solve_rapm_league` (D-I-wide, released as `ncaa_mbb/wbb_rapm`) · `mbb_lineup_stats` · `mbb_luck` · `mbb_positions` · `mbb_shot_selection` · `wbb_*` twins. Bundled: `mbb/wbb_in_game_wp.ubj`, `_box_bpm`, `_archetypes`, `_draft`, `_recruiting`, `_transfer` (.json).

Loaders

`load_mbb_pbp schedule team_boxscore player_boxscore rosters game_rosters officials standings player_core player_season_stats team_season_stats shots ratings player_value` + crosswalks; `load_wbb_*` identical. WP enriched in place in the released pbp.

Hockey · Baseball · Soccer · Cricket · Odds :: and the wider SportsDataverse

R siblings: fastRhockey · baseballr · softballR · oddsapiR · sportsdataversedata

HOCKEY · NHL · PWHL · HOCKEY.*

NHL — FIVE LIVE APIS

```
from sportsdataverse.nhl import *
nhl_web_schedule("2025-01-15")
# api-web.nhle.com
plays = nhl_web_pbp(2024020500)
# parsed by default
nhl_edge_skater_detail(8478402,
season=20242025)  # EDGE tracking
nhl_stats_rest_skater_report(...)
# api.nhle.com/stats/rest
nhl_records_awards()
# records.nhl.com
espn_nhl_pbp(game_id=401559000);
espn_nhl_schedule(dates=2025)
```

Parsers: 19 api-web (`parse_nhl_web_pbp` …), 4 EDGE families, generic stats-REST / records. Loaders: `load_nhl_pbp schedule games rosters player_box team_box goalie_box skater_box shifts scoring penalties shootout linescore officials three_stars scratches shots_by_period`.

NHL analytics

```
nhl_xg(pbp)                          # shot xG
(bundled model)
nhl_expected_assists(pbp);
nhl_faceoff_value(pbp)
nhl_gsax.nhl_goalie_gsax(pbp, shifts)
nhl_rapm.nhl_skater_rapm(pbp, shifts);
nhl_war.nhl_skater_war(...)
nhl_unit_ratings(pbp, shifts);
nhl_team_ratings(seasons=2025)
nhl_in_game_win_prob(pbp,
pregame_home_prob=0.55)
```

Also `nhl_penalty_value` · `nhl_zone_transitions` · `nhl_special_teams` · `nhl_edge_value` · `nhl_player_props`.

PWHL + 19 HOCKEYTECH LEAGUES

```
import sportsdataverse as sdv
sdv.pwhl_schedule(season=2025);
sdv.pwhl_pbp(game_id)
sdv.pwhl_game_shifts(game_id);
sdv.pwhl_game_corsi(game_id)
sdv.echl_schedule(season=2026);
sdv.ushl_standings(season=2025)
```

One registry-driven core (`hockeytech.LEAGUES`): 13 callables per league — `<lg>_schedule _pbp _standings _teams _team_roster _player_stats _leaders _game_summary _game_shifts _player_toi _game_corsi _season_id` + `most_recent_<lg>_season`. Leagues: `pwhl ahl ohl whl qmjhl echl sphl chl ushl bchl ajhl sjhl ojhl cchl gojhl mhl nojhl vijhl kijhl mjhl`. PWHL extras: `pwhl_goalie_gsax` · `pwhl_skater_rapm` · `pwhl_in_game_win_prob` · `pwhl_team_ratings`. College hockey: `espn_mch_*` / `espn_wch_*`, `college_hockey_ratings`.

BASEBALL · MLB · BASEBALL

MLB STATS API + ESPN

```
from sportsdataverse.mlb import *
mlb_schedule(date="2025-07-04")         #
statsapi.mlb.com
mlb_boxscore(game_pk=745000);
mlb_play_by_play(game_pk)
espn_mlb_pbp(game_id=401570000);
espn_mlb_schedule(dates=2025)
```

~80 Stats-API wrappers (`mlb_*`). `load_mlb_*`: `batter_projection catcher_framing command_plus expected_hr expected_stats oaa re24_matrix stuff_plus we_table wpa xera`.

STATCAST — BASEBALL SAVANT (43 ENDPOINTS)

```
mlb_statcast_search("2025-04-01", "2025-04-
07",
                    player_type="pitcher",
pitch_type="FF")
mlb_statcast_leaderboard_expected_stats(year=2025)
# 37 boards
mlb_statcast_gamefeed(game_pk=745000)
mlb_statcast_player(660271,
section="statcast")
```

Families `search` (chunked; `_minors`, `_wbc`) · `leaderboard` · `gamefeed` · `player`; friendly filters (`season pitch_type at_bat_result`) translate to Savant params.

MLB ANALYTICS

```
mlb_stuff_plus(pitches);
mlb_command_plus(pitches)
mlb_expected_stats("2025-04-01", "2025-09-30")
mlb_catcher_framing(pitches);
mlb_fielding_oaa(bip)
mlb_run_expectancy_matrix(seasons=2024);
mlb_win_expectancy(pbp, results)
mlb_pitch_classify(pitches);
mlb_batter_projection(2026)
```

Also `mlb_expected_home_runs` · `mlb_run_values` · `mlb_swing_decision` · `mlb_pitch_sequencing` · `mlb_pitch_fatigue` · `mlb_stolen_base` · `mlb_baserunning` · `mlb_umpire_zone` · `mlb_team_projection` · `mlb_prop_projection`.

COLLEGE BASEBALL & SOFTBALL

```
from sportsdataverse.baseball.college_baseball
import *
espn_college_baseball_scoreboard(dates=2025)
parse_college_baseball_ncaa_pbp(html)   #
stats.ncaa.org
college_baseball_re24(...);
college_baseball_wpa(...)  # + softball
```

SOCCER · CRICKET · ODDS · WIN-EXPECTANCY

SOCCER (ESPN, 12 LEAGUES)

```
from sportsdataverse.soccer.epl import
espn_epl_scoreboard
espn_epl_scoreboard(dates=20250315)
espn_soccer_team_schedule("eng.1",
team_id=359, season=2025)
```

Per-league modules `epl mls nwsl laliga bundesliga seriea ligue1 ligamx ucl uel wc wwc` (full ESPN surface each) + league-parameterized `espn_soccer_*`.

CRICKET (ESPN + BUNDLED WP)

```
from sportsdataverse.cricket import *
espn_cricket_scoreboard(league="icc",
dates=2025)
state = cricket_match_state(pbp)
cricket_win_probability(state);
cricket_wpa(state_wp)
```

ODDS — THE ODDS API (`ODDS_API_KEY`)

```
from sportsdataverse.odds import *
toa_sports(); toa_usage()
toa_sports_odds("basketball_nba",
regions="us", markets="h2h,spreads")
toa_event_odds(sport, event_id,
markets="player_points")
toa_sports_odds_history(sport, date="2024-11-
29T22:45:00Z")
toa_sports_scores(...); toa_sports_events(...)
```

Historical lines also ride the release loaders (`load_cfb_betting_lines`, `load_nfl_schedule`).

WIN-EXPECTANCY LAB (`WEXP`)

```
from sportsdataverse import wexp
oracle = wexp.build_cfb_market_oracle(...)  #
or _nfl_
pred =
wexp.build_predictor(wexp.VariantConfig(...))
wexp.backtest.run_backtest(oracle, pred,
model_id="ridge")
```

Ridge-margin, Elo, Glickman–Stern predictors vs market closing lines; Brier + calibration scoring. Football-only today.

THE WIDER SPORTSDATAVERSE

R PACKAGES (SPORTSDATAVERSE.ORG)

Package	Domain	sdv-py mirror
cfbfastR	college football	`cfb` · cfbfastR-data loaders
hoopR	NBA · men's college	`nba mbb`
wehoop	WNBA · women's college	`wnba wbb`
baseballr	MLB · Statcast · college	`mlb baseball`
fastRhockey	NHL · PWHL · HockeyTech	`nhl pwhl hockey`
softballR	college softball	`baseball.college_softball`
recruitR	CFB recruiting	`cfb.sports247_* on3_*`
oddsapiR	The Odds API	`odds`
nflseedR · cfbseedR	season sims / seeding	`nfl_simulations cfb_simulations`
sportsdataversedata	release publishing	`release`

Related: nflfastR / nflreadr (nflverse) · cfbplotR · sportyR / sportypy · sportsdataverse-js.

DATA REPOS BEHIND THE LOADERS

Each sport has a `*-raw` (scraped JSON, committed per game) and a `*-data` (tidy parquet on GitHub releases) repo under `github.com/sportsdataverse`. Loaders read the release asset; `sportsdataverse.release` writes them.

```
sportsdataverse_save(df, "pbp_2025", "cfb",
release_tag="cfb_pbp")
```

CONVENTIONS TO REMEMBER

- polars 1.x API everywhere; `return_as_pandas=True` converts at the edge.
- IDs are join keys — one dtype per id; never paper over with a float→str cast.
- Live tests gate on `SDV_PY_LIVE_TESTS=1`; stats.nba.com on `SDV_PY_NBA_STATS_LIVE=1`.
- Reference docs + returns tables are generated from endpoint YAML (`tools/codegen`).

Models :: the cross-sport modeling grid, bundled artifacts, validation gates

Every row below is code that ships in the wheel — no notebooks, no downloads (except NFL xYAC).

ONE SPINE, EVERY SPORT

Each league implements the same five layers. Names are the public entry points; `·` separates alternatives, — means not built for that league.

League	Team ratings	Game predict	Season / bracket sim	In-game WP	Player value
CFB	`cfb_ratings`	`cfb_predict_games`	`cfb_simulations` · `cfb_season_odds` · `cfb_playoff_seeds`	`wp_spread` / `wp_naive` via `CFBPlayProcess`	`cfb_adjusted_epa` · `cfb_roster_talent` · `cfb_draft_projection`
NFL	`nfl_ratings`	`nfl_market` · `nfl_gamescript`	`nfl_simulations` (+ `nfl_standings_calc`)	`enrich_nfl_pbp` → `wp`, `vegas_wp`	`nfl_projection` · `nfl_kicker_rating` · `nfl_line_grades`
NBA / G-Lg	`nba_team_ratings`	`nba_game_predict.nba_predict_games`	— (schedule-driven via predict)	`nba_in_game_win_prob`	`nba_rapm` · `nba_adj_rapm` · `nba_bpm` · `nba_spm` · `nba_war` · `nba_darko` · `nba_shot_value`
WNBA	`wnba_team_ratings`	`wnba_game_predict.wnba_predict_games`	—	`wnba_in_game_win_prob`	`wnba_engine` RAPM · `wnba_shot_value` · `wnba_playtype_impact`
MBB / WBB	`mbb_team_ratings` · `torvik_ratings`	`mbb_predict_games`	`mbb_season_sim` · `mbb_bracketology` · `mbb_bracket_sim`	`mbb_in_game_win_prob`	`mbb_box_bpm` · `mbb_rapm` · `mbb_shot_quality` · `mbb_archetypes` (+ `wbb_*`)
NHL	`nhl_team_ratings`	`nhl_market.nhl_predict_games`	—	`nhl_in_game_win_prob`	`nhl_xg` · `nhl_skater_rapm` · `nhl_skater_war` · `nhl_goalie_gsax` · `nhl_unit_ratings`
PWHL	`pwhl_team_ratings`	`pwhl_market.pwhl_predict_games`	—	`pwhl_in_game_win_prob`	`pwhl_skater_rapm` · `pwhl_skater_war` · `pwhl_goalie_gsax`
MLB	`mlb_team_projection`	— (projection-driven)	—	`mlb_win_expectancy`	`mlb_stuff_plus` · `mlb_command_plus` · `mlb_expected_stats` · `mlb_catcher_framing` · `mlb_fielding_oaa`
Others	college hockey `college_hockey_ratings` · cricket `cricket_win_probability` / `cricket_wpa` · spring football reuses the NFL engine (`build_spring_football_pbp` → `enrich_spring_football_pbp`)				

Shared shape: ratings take `seasons=` + `as_of_date=` (leak-safe, exclusive cut); predictors take `(games, ratings)`; sims take `(ratings | games, …, n_sims / simulations)`; in-game WP takes `(pbp, pregame_home_prob)`.

DRAFT, PROJECTION & ROSTER MODELS

Family	Entry points
Draft value / rookies	`nba_draft_model` · `nba_rookie_projection` · `wnba_draft_model` · `nfl_draft_model` · `mbb_draft_projection` · `cfb_draft_projection`
Aging & availability	`nba_aging_curve` · `nba_availability` · `wnba_career_trajectory` · `nfl_availability`
Recruiting / transfers	`cfb_recruiting_projection` · `cfb_transfer_impact` · `mbb_recruiting_projection` · `mbb_transfer_projection`
Props & markets	`nba_player_props` · `wnba_player_props` · `nfl_player_props` · `nhl_player_props` · `pwhl_player_props` · `mlb_prop_projection` · `nfl_market` · `nhl_market`

BUNDLED ARTIFACT REGISTRY

Shipped as package data (`importlib.resources`); XGBoost `.ubj` unless noted. Feature counts from the model cards.

Dir	Artifacts
`cfb/models`	`ep_model` 8f · `wp_naive` 12f · `wp_spread` 13f · `cfb_cp_model` · `qbr_model` 10f · `fg_model` 5f · `fd_model` · `two_pt_model` 4f · `xpass_model` 7f · `punt_distribution.parquet` · `cfb_field_position_ep.parquet`
`nfl/models`	`ep_model` 18f · `wp_spread` 12f · `wp_naive` 11f · `cp_model` 18f · `fg_model` · `qbr_model` · `two_pt_model` · `xpass_model` · `nfl_playcall` · `punt_data.parquet`; xYAC (76-class) fetched on first use
`nba/models`	`nba_in_game_wp` · `wnba_in_game_wp` · `{nba,wnba}_aging_curve` · `_draft_value` · `_rookie_projection` · `_availability` (.json)
`mbb/models`	`{mbb,wbb}_in_game_wp` · `_box_bpm` · `_archetypes` · `_draft` · `_recruiting` · `_transfer` (.json)
`nhl/models`	`nhl_in_game_wp.json` (+ xG coefficients in `nhl_xg`)
`mlb/models`	`mlb_stuff_plus` · `mlb_command_plus`
`cricket/models`	`cricket_resource_surface.parquet` · `cricket_winprob_calibration.parquet`

Every model's fitting script and era cuts live in the `*-data` producer repo that trains it (cfbfastR-cfb-data, nfl-data, hoopR-nba-data …); constants in `*_constants.py` / `model_vars.py` cite them.

RULE-ERA & FEATURE CONVENTIONS

- NFL era cuts `ERA_SEASON_CUTS` = 2001 / 2005 / 2013 / 2017; CFB models are per rule era.
- Kickoff / PAT feature substitution: touchback yardline 80 pre-2016, 75 from 2016; `down→1`, `ydstogo→10`.
- Spread WP: `spread_time` decays with exponent −4.0 (`SPREAD_TIME_DECAY_EXPONENT`).
- Outputs are `Float64`; models emit float32 — cast at the boundary.

VALIDATION & ORACLE GATES

Every model ships with a gate against an external oracle captured as a committed fixture. Gates are measured from observed values and never lowered.

Metric	Used for
Brier · log-loss · calibration table	win probabilities, in-game WP, sims (`brier_score`, `calibration_table` in `*_prediction_constants` / `wexp`)
MAE vs market	spreads / totals (`nfl_market`, `cfb_season_odds`, wexp market oracles)
Spearman ρ	ratings vs Torvik / KenPom / ESPN FPI / MoneyPuck; player value vs EPM, DARKO, RAPM, LEBRON
Correlation floors	parity with the R producers (nflfastR EP 0.996, CFB ratings ρ 0.97)

Oracle loaders

`load_darko_dpm` · `load_epm` · `load_lebron_daily` · `load_rapm_ryan_davis` · `load_dunks_threes_stats` · `load_draft_outcomes` · `torvik_ratings` · `bart_wbb_ratings` · `load_cfb_fpi_weekly` · `load_cfb_power_index` · `wexp.build_cfb_market_oracle` / `_nfl_`.

LEAK-SAFETY CONTRACT

- `as_of_date` / `through_week` are **exclusive**: the cut game is never in the training window.
- Every lag, cumulative sum or shift is `.over("game_id")` (or season) — no cross-game bleed when frames are concatenated.
- Ratings for a date use only games with `game_date < as_of_date`; sims draw from those ratings.
- Fitted constants (HFA, σ, λ) name the fitting script; assert the **output** changed, not that the code ran.

WIN-EXPECTANCY LAB (WEXP)

```
from sportsdataverse import wexp
o  = wexp.build_cfb_market_oracle(...)
p  = wexp.build_predictor(wexp.VariantConfig(...))
wexp.backtest.run_backtest(o, p, model_id="ridge")
```

Ridge-margin, Elo, Glickman–Stern vs closing lines; vintage store for reproducible walk-forward runs.

Recipes :: end-to-end pipelines, polars idioms, gotchas

Copy-paste starting points — each runs as written given the named inputs.

CFB · RAW JSON → ENRICHED PBP (OFFLINE)

```
from sportsdataverse.cfb import CFBPlayProcess
proc = CFBPlayProcess(gameId=401628334,
    path_to_json="raw/2024/",
    odds_override={"gameSpread": -6.5,
"overUnder": 55.5,
                   "homeFavorite": True,
                   "gameSpreadAvailable":
True})
proc.cfb_pbp_disk()
pbp = proc.run_processing_pipeline()   #
EPA/WPA, no network
box = proc.create_box_score()
```

The override injects persisted odds so the rebuild never falls back to the live odds endpoint (provenance: `pbp_txt["odds_source"] == "injected"`).

NFL · LOAD → ENRICH → 4TH-DOWN TABLE

```
import sportsdataverse.nfl as nfl
from sportsdataverse.nfl.ep_wp import
enrich_nfl_pbp
from sportsdataverse.nfl.nfl_fourth_down
import get_4th_down_probs
pbp = nfl.load_pbp([2024])
df  = enrich_nfl_pbp(pbp)          #
ep→epa→wp→wpa→cp→xyac
fourth = get_4th_down_probs(
    df.filter(pl.col("down") == 4))
```

`get_4th_down_probs` adds go / FG / punt WP + the recommendation columns (nfl4th-style, fully offline).

SPRING FOOTBALL ON THE NFL ENGINE

```
from sportsdataverse.football.ufl import
espn_ufl_summary
from
sportsdataverse.football.spring_football_ep_wp
import *
raw = espn_ufl_summary(game_id)
pbp = build_spring_football_pbp(raw,
league="ufl")
pbp = enrich_spring_football_pbp(pbp)
```

NBA · POSSESSIONS → RAPM → WAR

```
from sportsdataverse.nba.nba_season_compile
import compile_nba_season
from sportsdataverse.nba.nba_rapm import
nba_rapm_from_games
from sportsdataverse.nba import nba_war
poss  = compile_nba_season(2025)   # end-year:
2024-25
rapm  =
nba_rapm_from_games(poss["game_id"].unique())
value = nba_war(rapm, poss,
                replacement_level=-2.0,
pts_per_win=30.0)
```

Set `SDV_PY_NBA_RAW_JSON_DIR` first and the compile reads disk before network; add `_READONLY=1` for a fully offline, reproducible build (a miss raises `RawStoreMissError`).

MBB · STATS.NCAA.ORG PBP → LINEUPS → ON/OFF

```
from sportsdataverse.mbb import
(ncaa_mbb_game_pbp,
    ncaa_mbb_lineups, ncaa_mbb_possessions,
ncaa_mbb_on_off)
pbp   = ncaa_mbb_game_pbp(6305900)
lu    = ncaa_mbb_lineups(pbp)
poss  = ncaa_mbb_possessions(pbp)
oo    = ncaa_mbb_on_off("PlayerName", lu)
```

WBB mirrors every call (`ncaa_wbb_*`); pre-2016 WBB pbp is halves, not quarters — 0 rows for a quarters filter is data, not an error.

MLB · STATCAST SEARCH → STUFF+

```
from sportsdataverse.mlb import
mlb_statcast_search, mlb_stuff_plus
pitches = mlb_statcast_search("2025-04-01",
"2025-04-30",
    player_type="pitcher", pitch_type="FF")
stuff = mlb_stuff_plus(pitches, level="pitch")
```

POLARS IDIOMS FOR PBP WORK

```
# EPA per drive, success rate — group, don't
loop
pbp.group_by("game_id", "drive_id",
maintain_order=True) \
   .agg(pl.col("epa").sum(), (pl.col("epa") >
0).mean()
        .alias("success_rate"))

# lags NEVER cross a game boundary
pbp.with_columns(pl.col("score_diff").shift(1)

.over("game_id").alias("prev_diff"))

# explicit bool masks (house style)
pbp.filter((pl.col("pass") == True) &
           (pl.col("garbage_time") == False))

# no lookaround in Rust regex — case-toggle
instead
pl.col("text").str.extract(
    r"(?i)sacked by(?-i: ([A-Z][\w'\.\-]+))",
1)

# pandas only at the very edge
df = any_loader(..., return_as_pandas=True)
```

CROSS-LEAGUE JOINS

```
# pin ONE dtype per id, assert before joining
a =
a.with_columns(pl.col("athlete_id").cast(pl.Int64))
assert a.schema["athlete_id"] ==
b.schema["athlete_id"]
a.join(b, on="athlete_id", how="left")
# ESPN ↔ stats.nba.com identity
xwalk = sdv.load_nba_player_crosswalk(seasons=
[2025])
```

Never stringify a float id (`"123.0"` ≠ `"123"`); if you must go Utf8, cast through Int64 first.

GOTCHA STRIP

Trap	Rule
Season year	NBA/WNBA-stats dirs use the **end year** (2024 = 2023-24); ESPN loaders use kickoff/tip year; NBA span params want `"2023-24"`
stats.nba.com	TLS-fingerprint block → silent hang, not an error; the runtime uses `curl_cffi impersonate="chrome"`; run live only from residential IPs
Game ids	stats.nba.com `game_id` is a zero-padded **string** (`"0022400001"`) — never int-cast it
WBB history	halves before 2016 — quarter-keyed logic silently empties six seasons
Cache	`@cached_loader` keys on (name, args) not URL — `clear_cache()` after changing loader targets
Empty ≠ error	parsers return zero-row frames with the documented schema; check `df.is_empty()`, don't try/except
ESPN rate	Core v2 403s under aggressive parallelism — keep scrapes low-concurrency

TEST GATES

```
uv run pytest                          # offline
suite
SDV_PY_LIVE_TESTS=1 uv run pytest      # live
APIs
SDV_PY_NBA_STATS_LIVE=1 ...            #
stats.nba.com only,
                                       #
residential IP
```

CONTRIBUTING A NEW ENDPOINT

- Wrapper metadata lives in `tools/codegen/endpoints/*.yaml`; run `uv run python tools/codegen/generate.py`, never hand-edit generated files.
- Returns-table descriptions go in `manual_column_descriptions.yaml` (schemas/** is clobbered on re-capture).
- Validate parsers against **real captured fixtures**, never synthetic ones.
- New modules are fully typed and join the mypy ratchet in `pyproject.toml`.
- `generate.py --check` is the CI drift gate — regenerate before every push.