Package 'cfbseedR'

September 9, 2026

Title Simulate and Evaluate College Football Seasons

Version 0.2.0

Description Simulate college football seasons and compute standings, conference ranks, conference champions, and College Football Playoff (CFP) seeds. Implements the official season-scoped championship-game tiebreaker procedure of each Football Bowl Subdivision (FBS) conference and the CFP's season-keyed automatic-qualifier policies, including the 2026 formats. Seasons are simulated week by week with a pluggable results generator, and external tiebreaker inputs such as committee rankings or analytics composites can be supplied. Adapts the approach of the 'nflseedR' package by Sebastian Carl and Lee Sharpe to college football semantics.

License MIT + file LICENSE

URL https://cfbseedR.sportsdataverse.org, https://github.com/sportsdataverse/cfbseedR

BugReports https://github.com/sportsdataverse/cfbseedR/issues

Depends R (>= 4.1.0)

Imports cli, dplyr (>= 1.1.0), furrr, future, progressr, purrr, rlang, stats, tibble, tidyr, utils

Suggests cfbfastR, gt (>= 0.9.0), knitr, rmarkdown, scales (>= 1.2.0), testthat (>= 3.0.0)

Encoding UTF-8

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

LazyData true

NeedsCompilation no

Author Saiem Gilani [cre, aut, cph] (ORCID: <https://orcid.org/0000-0002-7194-9067>), Sebastian Carl [aut] (Author of nflseedR, from which cfbseedR is adapted), Lee Sharpe [aut] (Author of nflseedR, from which cfbseedR is adapted)

Maintainer Saiem Gilani <saiem.gilani@gmail.com>

Repository CRAN

Date/Publication 2026-09-09 14:50:11 UTC

Contents

`cfbseedR_compute_results`

Compute CFB Game Results in Season Simulations

Description

The default `compute_results` function used by `cfb_simulations()`. It is a faithful adaptation of nflseedR's `nflseedR_compute_results()` (a variant of 538's ELO model, originally coded by Lee Sharpe and rewritten by Sebastian Carl): a simple dynamic ELO carried through the `teams` table, updated each week from (real or simulated) results.

Usage

```
cfbseedR_compute_results(teams, games, week_num, ...)
```

Arguments

`teams`	A data frame of teams by simulation with at least the columns `sim` and `team`. May carry an `elo` column between weeks.
`games`	A data frame of games with at least `sim`, `game_type`, `week`, `home_team`, `away_team`, `result`, and optionally `neutral` (0/1; a home-field ELO bonus of 20 applies when not neutral).
`week_num`	The (numeric) week to simulate. Only games with `week == week_num & is.na(result)` receive a simulated result.
`...`	Optionally pass `elo`, a named vector of initial ELO ratings (names = team names). If absent, teams start at random ratings drawn from `rnorm(1500, 150)`.

Details

Mechanics (constants identical to nflseedR): ELO difference is `home - away` plus 20 for true home games, multiplied by 1.2 in the postseason (`game_type != "REG"`). Win probability is `1 / (10^(-elo_diff / 400) + 1)`, the margin estimate is `elo_diff / 25`, and simulated margins are drawn from `rnorm(mean = estimate, sd = 13)` and rounded away from zero. Ties are possible in the regular season only; a simulated postseason tie is re-decided by the win probability with a 3-point margin. nflseedR's rest-day adjustment is dropped (no rest data in the CFB schema).

Value

A list with two elements, as required by the `compute_results` contract (see `simulations_verify_fct()`):

Element	Type	Description
`teams`	data.frame	The input `teams` table with the `elo` column added/updated from this week's results.
`games`	data.frame	The input `games` table with `result` filled for this week's previously missing results (home margin, inte

See Also

`cfb_simulations()`, `simulations_verify_fct()`

Examples

```
games <- read.csv(system.file("extdata", "toy_games.csv", package = "cfbseedR"))
teams <- read.csv(system.file("extdata", "toy_teams.csv", package = "cfbseedR"))
games$result[games$week >= 3] <- NA
teams$sim <- 1
games$sim <- 1
out <- cfbseedR_compute_results(teams, games, week_num = 3)
out$games[out$games$week == 3, ]
```

`cfb_games_example` *Example Games of a Toy College Football Season*

Description

A nine-team, two-conference toy season (plus one independent) used throughout the examples, the vignettes, and as the default input of `simulations_verify_fct()`. It is the same data shipped in `inst/extdata/toy_games.csv`, so it needs no network access.

Usage

```
cfb_games_example
```

Format

A data frame with the columns `cfb_standings()` requires:

Column	Type	Description
`season`	integer	Season identifier.
`game_type`	character	`"REG"` or `"CONF_CHAMP"`.
`week`	integer	Week number of the game.
`home_team`, `away_team`	character	Team names matching `cfb_teams_example$team`.
`home_points`, `away_points`	integer	Final scores (feed the SEC capped-margin rung).
`result`	integer	Home margin, i.e. home score minus away score.

See Also

cfb_teams_example, `cfb_standings()`, `cfb_simulations()`

Examples

```
head(cfb_games_example)
```

`cfb_games_from_schedule`

Build an Engine Games Table from a cfbfastR Schedule

Description

Maps the schedule shape returned by `cfbfastR::load_cfb_schedules()` to the games schema used by `cfb_standings()` and `cfb_simulations()`. cfbfastR is not required - any data frame with the expected columns works.

Usage

```
cfb_games_from_schedule(schedule)
```

Arguments

`schedule`	A data frame with at least `season`, `week`, `season_type`, `home_team`, `away_team`. Optional columns used when present: `home_points`/`away_points` (to compute `result`), `neutral_site`, and `notes` (games whose notes mention "championship" are classified as `"CONF_CHAMP"`).

Details

Game type mapping: `season_type == "postseason"` becomes `"POST"`, games whose `notes` contain "championship" (case-insensitive) become `"CONF_CHAMP"`, everything else `"REG"`. This is a documented heuristic - CFBD marks conference championship games as regular-season games with a championship note.

Value

A tibble in the engine games schema, ready for `cfb_standings()` / `cfb_simulations()`:

Column	Type	Description
`season`	integer	Season taken from the schedule.
`week`	integer	Week number of the game.
`game_type`	character	`"REG"`, `"CONF_CHAMP"` (notes mention "championship"), or `"POST"` (`season_type == "postseason`
`home_team`	character	Home team name.
`away_team`	character	Away team name.
`result`	numeric	Home margin (home minus away points); `NA` for unplayed games.
`neutral`	integer	Neutral-site flag (0/1).

See Also

`cfb_standings()`, `cfb_simulations()`, `cfbfastR::load_cfb_schedules()` from cfbfastR, the nflseedR original: `https://nflseedr.com`

Examples

```
schedule <- data.frame(
  season = 2024, week = c(1, 1, 15, 16),
  season_type = c("regular", "regular", "regular", "postseason"),
  home_team = c("A1", "B1", "A1", "A1"),
  away_team = c("A2", "B2", "A2", "B1"),
  home_points = c(21, 24, 17, NA), away_points = c(14, 20, 14, NA),
  neutral_site = c(FALSE, FALSE, TRUE, TRUE),
  notes = c(NA, NA, "Alpha Championship Game", NA)
)
cfb_games_from_schedule(schedule)
```

---

cfb_playoff_seeds *Compute College Football Playoff Seeds*

Description

Implements 12-team CFP **straight seeding** with a season-keyed automatic-qualifier policy:

- `autobid = "2026"` (default, current rule): the ACC, Big 12, Big Ten and SEC champions are in **regardless of ranking**; the highest-ranked team from the Group of 6 (American, Conference USA, MAC, Mountain West, Pac-12, Sun Belt) is in whether or not it won its conference; and Notre Dame is in if ranked inside the top `playoff_seeds`.
- `autobid = "2025"`: the 5 highest-ranked conference champions are guaranteed inclusion (the 2024-2025 rule).

Under both policies seeds are assigned strictly in ranking order (champions are not bumped up; straight seeding, 2025+).

Usage

```
cfb_playoff_seeds(
  standings,
  rankings = NULL,
  playoff_seeds = 12L,
  autobid = c("2026", "2025")
)
```

Arguments

standings	A standings table as returned by cfb_standings() (requires at least team, conference, conf_champ, win_pct, sov, sos, pd, and a sim or season id column).
rankings	Optional data frame with columns team and rank (1 = best), e.g. the CFP committee rankings. Teams missing from rankings are treated as unranked and ordered behind all ranked teams. When NULL, a documented fallback ordering is used instead: teams are ordered by win percentage, then strength of victory, strength of schedule, point differential, and team name.
playoff_seeds	Number of playoff spots (default 12).
autobid	Automatic-qualifier policy, "2026" (default) or "2025" - see Description.

Details

Under autobid = "2025", if there are fewer than 5 conference champions all champions are guaranteed (capped at playoff_seeds). Under either policy a guaranteed team ranked outside the top playoff_seeds displaces the lowest-ranked at-large team and is seeded by its rank order within the field. Conference names are matched against both cfbd and short spellings (e.g. "American Athletic" / "American").

Value

The standings input (all its columns unchanged; see cfb_standings() for the column table) with one column added:

Column	Type	Description
seed	integer	CFP seed in straight-seeding order; NA for teams outside the playoff field.

See Also

cfb_standings(), cfb_simulations(), the nflseedR original: https://nflseedr.com

Examples

```
games <- read.csv(system.file("extdata", "toy_games.csv", package = "cfbseedR"))
teams <- read.csv(system.file("extdata", "toy_teams.csv", package = "cfbseedR"))
standings <- cfb_standings(games, teams, tiebreaker_depth = "POINTS",
                           verbosity = "NONE")
```

```
rankings <- data.frame(team = c("B1", "I1", "A1", "A3"), rank = 1:4)
seeded <- cfb_playoff_seeds(standings, rankings = rankings, playoff_seeds = 4)
seeded[!is.na(seeded$seed), c("team", "seed")]
```

`cfb_simulations` *Simulate a College Football Season*

Description

Simulates college football seasons based on a games/schedule table that holds matchups with and without results. Missing results are computed week by week using the pluggable `compute_results` function (default: `cfbseedR_compute_results()`, an ELO-based generator adapted from nflseedR). After the scheduled games, standings, conference champions, and CFP seeds are computed, and - with `sim_include = "POST"` - the playoff bracket is simulated round by round.

Usage

```
cfb_simulations(
  games,
  teams,
  compute_results = cfbseedR_compute_results,
  ...,
  simulations = 10000L,
  playoff_seeds = 12L,
  tiebreaker_depth = c("SOS", "PRE-SOV", "POINTS", "RANDOM"),
  sim_include = c("POST", "REG"),
  rankings = NULL,
  autobid = c("2026", "2025"),
  tiebreaker_data = NULL,
  chunks = 8L,
  verbosity = c("MIN", "MAX", "NONE")
)
```

Arguments

`games`	A data frame of games of a **single** season, in the schema of `cfb_standings()` plus an optional `neutral` column (0/1). Games with `result = NA` are simulated. Must not contain `game_type == "POST"` rows - the playoff bracket is generated from the computed seeds.
`teams`	A data frame with columns `team` and `conference`. Teams with conference `"FBS Independents"` or `NA` are treated as independents: they appear in overall standings but receive no conference rank. An optional `division` column (e.g. `"FBS"/"FCS"`) feeds the Big 12 `total_wins` FCS cap; absent -> that cap degrades to uncapped win totals (noted, see `tiebreak_notes` below). An optional

conf_division column (e.g. "East"/"West") turns on division-format ranking for a conference: each division is ranked separately (with that conference's registered procedure) and the division champions take conf_rank 1-2. Supplying divisions for a conference IS the opt-in - correct for the 2026 Sun Belt and equally for historical divisional seasons of any conference. An optional logical postseason_eligible column (explicit FALSE = ineligible) keeps a team out of the championship-game ranks while its games still count in every comparison (the ACC policy makes this explicit); with fewer than two eligible teams in a conference the remaining conf_rank 1-2 slots are necessarily filled by ineligible teams. teams need not list every team that appears in games - an unlisted opponent (e.g. an FCS-or-lower team) gets no standings row of its own, but its games still count toward its opponents' records and toward the Big 12 total_wins FCS cap (an unknown opponent counts as FCS-or-lower).

compute_results
: A function computing results of games, with the required arguments teams, games, and week_num. See simulations_verify_fct() for the contract and cfbseedR_compute_results() for the default.

...
: Additional parameters passed on to compute_results.

simulations
: The number of times the season shall be simulated.

playoff_seeds
: Number of CFP spots (default 12), passed to cfb_playoff_seeds().

tiebreaker_depth
: One of "SOS" (default), "PRE-SOV", "POINTS", or "RANDOM". Controls how deep the tiebreaker cascade goes before falling back to a coin flip:

  - "RANDOM": coin flip immediately.
  - "PRE-SOV": head-to-head and common opponents only.
  - "SOS": adds strength of victory, then strength of schedule.
  - "POINTS": adds conference point differential.

  This depth ladder gates ONLY the generic fallback cascade used by unregistered conferences; the registered official procedures below always run in full.

sim_include
: One of "POST" (default) or "REG":

  - "REG": simulate the remaining schedule and compute standings, conference champions, and playoff seeds.
  - "POST": "REG" + simulate the playoff bracket.

rankings
: Optional committee rankings (team, rank) used for CFP seeding, held static across simulations. When NULL, seeding falls back to the per-simulation standings ordering (see cfb_playoff_seeds()).

autobid
: CFP automatic-qualifier policy passed to cfb_playoff_seeds(): "2026" (default, current rule) or "2025".

tiebreaker_data
: Optional named list of external tiebreaker inputs (analytics_ratings, cfp_rankings, apr), held static across simulations - see cfb_standings(). Missing inputs skip their rungs (noted).

chunks
: The number of chunks the simulations are split into (default 8, capped at simulations). Chunks are dispatched with furrr::future_map(), so they run in parallel as

soon as you set a parallel plan, e.g. `future::plan("multisession")`; with the default sequential plan they simply run one after another. There is no universally best number - too many chunks can be as slow as too few.

`verbosity` One of `"MIN"` (default), `"MAX"`, or `"NONE"`, as in `cfb_standings()`. `"NONE"` silences the progress messages entirely.

Details

The playoff bracket is a standard single-elimination bracket of size `2^ceiling(log2(playoff_seeds))` with byes for the top seeds - for 12 seeds this reproduces the CFP bracket (quarterfinals 1 vs 8/9 winner, 4 vs 5/12, 3 vs 6/11, 2 vs 7/10). First-round games are hosted by the higher seed; later rounds are neutral-site. There is no reseeding (fixed bracket, per the CFP format). Conference championship matchups are simulated as scheduled, not re-derived from simulated standings.

Simulations are split into `chunks` and dispatched with furrr, so a parallel `future::plan()` spreads them across cores. Progress can be reported by turning on `progressr::handlers()` before the call, or by piping the call into `progressr::with_progress()`.

Set a seed with `set.seed()` for reproducibility: furrr generates parallel-safe RNG streams per chunk, so a given seed reproduces exactly - but the stream differs from a single sequential pass, so results for the same seed change if you change `chunks`.

Value

A list of class `cfbseedR_simulation` with these elements:

Element	Type	Description
`standings`	tibble	Per-simulation standings in the `cfb_standings()` schema (see its column table; `sov`/`sos` are confer
`games`	tibble	All games of all simulations (`sim`, `game_type`, `week`, `home_team`, `away_team`, `result`, `neutral`), in
`overall`	tibble	Per-team means across simulations: `wins` (average), and probabilities `conf_champ`, `playoff`, `seed1`
`team_wins`	tibble	Per-team probability of clearing each half-win threshold (`team`, `wins`, `over_prob`, `under_prob`).
`game_summary`	tibble	Per-matchup aggregates: `away_wins`, `home_wins`, `ties`, mean `result`, `games_played`, `away_perce`
`sim_params`	list	The simulation parameters (number of simulations, seeds, depth, etc.).

See Also

`cfb_standings()`, `cfb_playoff_seeds()`, `cfbseedR_compute_results()`, `simulations_verify_fct()`, the nflseedR original: https://nflseedr.com

Examples

```
games <- read.csv(system.file("extdata", "toy_games.csv", package = "cfbseedR"))
teams <- read.csv(system.file("extdata", "toy_teams.csv", package = "cfbseedR"))
games$result[games$week >= 3] <- NA
set.seed(4)
sim <- cfb_simulations(games, teams, simulations = 4, playoff_seeds = 4)
sim$overall
```

`cfb_standings` *Compute College Football Standings*

Description

Computes overall and conference standings from a table of game results, including conference ranks (via a documented tiebreaker cascade), conference champions, and - optionally - College Football Playoff seeds.

Adapted from nflseedR's `nfl_standings()` with college football semantics: conferences instead of divisions, independents excluded from conference ranks, and conference championship games counting toward the overall record (and deciding the champion) but not the conference record.

Usage

```
cfb_standings(
  games,
  teams,
  ...,
  ranks = c("CONF", "NONE"),
  tiebreaker_depth = c("SOS", "PRE-SOV", "POINTS", "RANDOM"),
  playoff_seeds = NULL,
  rankings = NULL,
  tiebreaker_data = NULL,
  verbosity = c("MIN", "MAX", "NONE")
)
```

Arguments

`games` A data frame of games. Required columns:

sim or season A season or simulation ID.

game_type One of `"REG"`, `"CONF_CHAMP"`, `"POST"`.

week Week number of the game.

home_team, away_team Team names matching `teams$team`.

result Home margin, i.e. home score minus away score. Must not be `NA` (play or simulate the games first).

Optional `home_points`/`away_points` columns (per-game scores) feed the official SEC `capped_scoring_margin` tiebreaker rung (see "Official per-conference tiebreakers" below); `cfb_games_from_schedule()` emits both. Absent -> that rung is skipped, not an error.

`teams` A data frame with columns `team` and `conference`. Teams with conference `"FBS Independents"` or `NA` are treated as independents: they appear in overall standings but receive no conference rank. An optional `division` column (e.g. `"FBS"`/`"FCS"`) feeds the Big 12 `total_wins` FCS cap; absent -> that cap degrades to uncapped win totals (noted, see `tiebreak_notes` below). An optional

conf_division column (e.g. "East"/"West") turns on division-format ranking for a conference: each division is ranked separately (with that conference's registered procedure) and the division champions take conf_rank 1-2. Supplying divisions for a conference IS the opt-in - correct for the 2026 Sun Belt and equally for historical divisional seasons of any conference. An optional logical postseason_eligible column (explicit FALSE = ineligible) keeps a team out of the championship-game ranks while its games still count in every comparison (the ACC policy makes this explicit); with fewer than two eligible teams in a conference the remaining conf_rank 1-2 slots are necessarily filled by ineligible teams. teams need not list every team that appears in games - an unlisted opponent (e.g. an FCS-or-lower team) gets no standings row of its own, but its games still count toward its opponents' records and toward the Big 12 total_wins FCS cap (an unknown opponent counts as FCS-or-lower).

`...` Currently unused.

`ranks` One of "CONF" (default) or "NONE". "NONE" returns the raw standings and skips the conference-rank tiebreaker walk entirely - and therefore also conf_champ and seed, which are derived from it. Useful when you only need records and the cascade would be wasted work.

`tiebreaker_depth`

One of "SOS" (default), "PRE-SOV", "POINTS", or "RANDOM". Controls how deep the tiebreaker cascade goes before falling back to a coin flip:

- "RANDOM": coin flip immediately.
- "PRE-SOV": head-to-head and common opponents only.
- "SOS": adds strength of victory, then strength of schedule.
- "POINTS": adds conference point differential.

This depth ladder gates ONLY the generic fallback cascade used by unregistered conferences; the registered official procedures below always run in full.

`playoff_seeds` If not NULL, a seed column is added via cfb_playoff_seeds() with this number of playoff spots.

`rankings` Optional committee-style rankings data frame with columns team and rank, passed to cfb_playoff_seeds(). Ignored when playoff_seeds is NULL.

`tiebreaker_data`

Optional named list of external inputs for the official registry rungs. Supported: analytics_ratings, a data frame with columns team and rating (feeds the SportSource-style rating / metric-composite rungs used by the Big Ten, Big 12, ACC, MAC, American, CUSA, Mountain West, and Sun Belt - supply your own composite of the conference's published metrics), and cfp_rankings, a data frame with columns team and rank (feeds the cfp_ranked_final_week clause used by the American, CUSA, and Sun Belt), and apr, a data frame with columns team and apr (multi-year Academic Progress Rate, the CUSA policy's late fallback). A missing input -> that rung is skipped (noted).

`verbosity` One of "MIN" (default), "MAX", or "NONE". "MAX" logs every tied group as it's broken.

Details

Conference ranks are seeded by conference win percentage; ties within a tier are broken by a documented cascade. **Registered conferences** (SEC, Big Ten, Big 12, ACC, MAC, American, Conference USA, Mountain West, Sun Belt) use their **official procedures** (see "Official per-conference tiebreakers" below); every other conference - including the re-formed Pac-12, which has not published a procedure - uses the **generic fallback**: head-to-head record among the tied teams, record vs. common conference opponents (minimum one), conference-scoped strength of victory, conference-scoped strength of schedule, conference point differential, and finally a coin flip, gated by `tiebreaker_depth`. All cascade quantities are computed over regular-season conference games so conference ranks depend only on conference play.

Official per-conference tiebreakers:

`CONFERENCE_TIEBREAKERS` (internal) registers each conference's official procedure as a list of season-scoped **epochs**, so a policy change applies from its first season: with a `season` id column, the ACC's all-new 2026 policy (head-to-head, then SportSource Team Success Ranking, then a draw, with the alternate-game-count candidate-pool rule) applies from 2026 while earlier seasons keep the 2024 cascade; plain `sim` ids resolve to the current rules. The SEC/Big Ten/Big 12 2024 procedures are ported verbatim from sdv-py's `cfb_standings.py` so both engines produce identical output on the shared cross-language parity fixture. Rung primitives: `h2h` (multi-team combined head-to-head, applied only when every tied pair played; otherwise only "defeated-all" elimination - the symmetric "lost-to-all" elimination is intentionally not modeled, a documented simplification, see `R/tiebreakers.R`), `record_vs_common`, `record_vs_common_desc` (descend the standings from best to worst, comparing a tied GROUP of common opponents collectively - the Big 12 rule, adopted for every registry descent rung), `opp_conf_win_pct` (pooled opponents' conference win pct - this reuses the existing `sos` column, which already computes the pooled sum-of-wins/sum-of-games formula), `capped_scoring_margin` (SEC: points scored capped at 42 / allowed capped at 48, per game, summed over conference games; needs `home_points`/`away_points`), `total_wins` (Big 12: overall wins with at most one win vs an FCS-or-lower opponent counted; needs `teams$division`), `analytics_rating` (external, via `tiebreaker_data$analytics_ratings` - stands in for each conference's published metric composite, e.g. Connelly SP+ / ESPN SOR / KPI / SportSource for the American, CUSA, and Mountain West), `cfp_ranked_final_week` (American/CUSA/Sun Belt: the best-ranked tied team advances if it won its final conference game, else the clause falls through; needs `tiebreaker_data$cfp_rankings`), `div_pct` (Sun Belt: win pct vs same-division opponents; needs `teams$conf_division`), and `coin_toss`. After each team is seeded/eliminated the procedure restarts from the first rung with the remaining tied set; when a rung's required input is unavailable it is skipped deterministically and the skip is recorded once (per conference) in `attr(result, "tiebreak_notes")`. Under registry conferences, `conf_rank` 1-2 are the two teams that reach the conference championship game (the cascade only orders them - see the design brief).

Value

A tibble of standings, one row per (`sim`, `team`) (the id column is named `season` if the input used `season`), sorted by sim, conference, conference rank, and team. Note that `sov` and `sos` are **conference-REG-scoped**: they are computed over regular-season conference games only, `sov` over conference victories and `sos` over conference opponents; independents get `0.0` for both. The result also carries a character vector `attr(result, "tiebreak_notes")` recording any registry rungs skipped for lack of their optional input (see "Official per-conference tiebreakers" above); empty

when nothing was skipped.

Column	Type	Description
sim / season	integer	Season or simulation ID (name follows the input).
team	character	Team name.
conference	character	Conference name ("FBS Independents" / NA = independent).
games	integer	Games played (REG + CONF_CHAMP).
wins	integer	True win count (ties not counted).
losses	integer	True loss count.
ties	integer	Tie count.
win_pct	numeric	Overall win percentage; a tie counts as half a win.
pd	integer	Overall point differential.
conf_games	numeric	Regular-season conference games played (0 for independents).
conf_wins	numeric	Wins over regular-season conference games.
conf_losses	numeric	Losses over regular-season conference games.
conf_ties	numeric	Ties over regular-season conference games.
conf_pct	numeric	Conference win percentage (CONF_CHAMP games excluded).
conf_pd	numeric	Point differential over regular-season conference games.
sov	numeric	Strength of victory, conference-REG-scoped: beaten conference opponents' conference wins divi
sos	numeric	Strength of schedule, conference-REG-scoped: all conference opponents' conference wins divide
conf_rank	integer	Rank within the conference via the tiebreaker cascade (NA for independents).
conf_champ	logical	Conference champion flag (decided by the CONF_CHAMP game).
seed	integer	CFP seed, only when playoff_seeds is not NULL (NA outside the field).

See Also

cfb_playoff_seeds(), cfb_simulations(), cfb_games_from_schedule(), the nflseedR original: https://nflseedr.com, and cfbfastR for real schedules

Examples

```
games <- read.csv(system.file("extdata", "toy_games.csv", package = "cfbseedR"))
teams <- read.csv(system.file("extdata", "toy_teams.csv", package = "cfbseedR"))
standings <- cfb_standings(games, teams, tiebreaker_depth = "POINTS",
                           verbosity = "NONE")
standings[, c("team", "conference", "conf_rank", "conf_champ")]

# An official-registry analytics rating input (used by Big Ten/Big
# 12/ACC/MAC when their cascade reaches the `analytics_rating` rung)
ratings <- data.frame(team = teams$team, rating = seq(90, 70, length.out = nrow(teams)))
standings2 <- cfb_standings(games, teams,
                            tiebreaker_data = list(analytics_ratings = ratings),
                            verbosity = "NONE")
attr(standings2, "tiebreak_notes")
```

cfb_teams_example *Example Teams of a Toy College Football Season*

Description

The team table that pairs with cfb_games_example: nine teams across two conferences plus one independent. Same data as `inst/extdata/toy_teams.csv`.

Usage

```
cfb_teams_example
```

Format

A data frame with the columns `cfb_standings()` requires:

Column	Type	Description
`team`	character	Team name, matching the game table.
`conference`	character	Conference name; `"FBS Independents"` marks an independent.

See Also

cfb_games_example, `cfb_standings()`

Examples

```
cfb_teams_example
```

fmt_pct_special *Format Probabilities for Display*

Description

Formats a numeric vector of probabilities as percentage strings the way simulation summaries want them: values below 1% render as `"<1%"` and values above 99.9% as `">99.9%"`, so a long-shot never reads as a flat `0%` and a near-lock never reads as a certain `100%`. Exact 0 and exact 1 are left as `"0%"` and `"100%"`.

Adapted from nflseedR's `fmt_pct_special()`.

Usage

```
fmt_pct_special(x)
```

Arguments

x	A numeric vector of probabilities, all between 0 and 1 (NA is allowed and returned as NA).

Value

A character vector the same length as x.

See Also

summary.cfbseedR_simulation()

Examples

```
fmt_pct_special(c(0, 0.0004, 0.123, 0.5, 0.9994, 1))
```

---

simulations_verify_fct

Verify a Custom CFB Result Simulation Function

Description

cfbseedR supports custom functions to compute results in season simulations through the compute_results argument of cfb_simulations(). This function verifies that a custom function behaves as the simulator expects, mirroring nflseedR's simulations_verify_fct(). It checks the output structure and whether game results are changed as expected, and errors with a hint at the first problem found.

Usage

```
simulations_verify_fct(compute_results, ..., games = NULL, teams = NULL)
```

Arguments

compute_results	A function to compute results of games. Required arguments: teams, games, week_num. It must return list(teams = teams, games = games) without removing rows or columns from games, must fill result for exactly the games with week == week_num & is.na(result), must not modify any other result, and must not produce ties (result == 0) outside the regular season.
...	Further arguments passed on to compute_results.
games	A schedule where some results are missing. Defaults to the bundled toy season with all results from week 3 onwards blanked.
teams	A teams table (team, conference, optionally sim). Defaults to the bundled toy teams.

Value

Returns TRUE invisibly if no problems are found.

See Also

cfb_simulations(), cfbseedR_compute_results()

Examples

```
simulations_verify_fct(cfbseedR_compute_results)
```

---

summary.cfbseedR_simulation

Compute a Pretty Simulation Summary Table

Description

Renders the overall table of a cfb_simulations() result as a gt table, one row group per conference, sorted by average wins. Probabilities are formatted with fmt_pct_special() so long-shots and near-locks stay readable.

The CFB counterpart of nflseedR's summary.nflseedR_simulation(); the layout is conference-grouped rather than nflseedR's AFC/NFC pair of columns, because college conferences are neither two nor equally sized.

Usage

```
## S3 method for class 'cfbseedR_simulation'
summary(object, ...)
```

Arguments

object	A cfbseedR_simulation object, as returned by cfb_simulations().
...	Additional arguments (currently unused).

Value

A gt_tbl object.

See Also

cfb_simulations(), fmt_pct_special()

Examples

```
games <- cfb_games_example
games$result[games$week >= 3] <- NA
set.seed(4)
sim <- cfb_simulations(games, cfb_teams_example,
                       simulations = 4, playoff_seeds = 4, chunks = 1)
summary(sim)
```

Index