oddsapiR :: CHEAT SHEET

Clean, tidy sports odds from **The Odds API** (the-odds-api.com): live and upcoming lines from dozens of bookmakers across US / UK / EU / AU regions, scores, player props, and historical odds snapshots. Part of the SportsDataverse; Python mirror: `sportsdataverse.odds`. Page 1 of 2: setup, the 15-function surface, arguments, quota.

v1.0.0

R ≥ 4.0.0 · httr2 · tidyr
oddsapiR.sportsdataverse.org

INSTALL

```
install.packages("oddsapiR")
# or the dev version:
pak::pak("sportsdataverse/oddsapiR")
library(oddsapiR)
```

API KEY SETUP

Every endpoint needs a (free-tier) key from **the-odds-api.com**, read from the `ODDS_API_KEY` env var.

Persistent (recommended)

```
usethis::edit_r_environ()
# paste in .Renviron, NO quotes:
ODDS_API_KEY = XXXX-YOUR-KEY-XXXX
# save, then restart R
```

Per session

```
Sys.setenv(
  ODDS_API_KEY = "XXXX-YOUR-KEY-XXXX")
```

Key helpers

`toa_key()`	the key, or `NA` if unset
`has_toa_key()`	`TRUE` / `FALSE`
`check_toa_key()`	errors loudly if unset

Docs: `?register_toa`.

QUOTA & CREDITS

Every response carries `x-requests-remaining / -used / -last` headers. Two ways to watch them:

```
toa_requests() # live check, FREE call
#   requests_remaining  requests_used
toa_quota()    # cached from the last
               # call - no network hit;
               # adds requests_last
```

Quota resets monthly. Cost of a call: **markets × regions** (odds), 10× that for historical odds. See the cost column → and page 2 for budgeting.

FUNCTION INDEX

Account & usage

Function	Cost
`toa_key` · `has_toa_key` · `check_toa_key`	—
`toa_requests()` — usage from live headers	free
`toa_quota()` — cached usage, offline	free

Sports & events

`toa_sports()` — coverage; the `key` col IS `sport_key`	free
`toa_sports_events()` — upcoming events + `event_id`	free
`toa_sports_participants()` — teams/players + ids	1
`toa_sports_scores()` — live + last-3-days scores	1–2

Odds & markets

`toa_sports_odds()` — featured markets, all games	mkt×reg
`toa_event_odds()` — one event; props + alternates	mkt×reg
`toa_event_markets()` — market keys live for an event	1

Historical (paid plans)

`toa_sports_odds_history()` — featured odds snapshot	10×mkt×reg
`toa_event_odds_history()` — event snapshot, props ok	mkt×reg
`toa_sports_events_history()` — events at a snapshot	1

All 15 exports return tibbles (class `oddsapiR_data`) and fail soft: on error you get an empty frame + a `cli` alert, never an exception.

ARGUMENT REFERENCE

Arg	Meaning
`sport_key`	from `toa_sports()$key`, e.g. `basketball_nba`
`event_id`	from `toa_sports_events()$id`
`regions`	`us us2 uk eu au`; comma-sep; each = 1 quota unit
`markets`	`h2h spreads totals outrights` (+ props on event endpoints); comma-sep
`odds_format`	`decimal` (default) | `american`
`date_format`	`iso` (default) | `unix`
`bookmakers`	comma-sep slugs (`draftkings,fanduel`); overrides `regions`; 10 books = 1 region of quota
`days_from`	scores: completed games 1–3 days back
`event_ids`	filter events/odds to listed ids
`commence_time_from` / `_to`	ISO 8601 window, e.g. `2023-09-09T00:00:00Z`
`include_rotation_numbers`	events: add `home/away_rotation`
`all_sports`	`toa_sports()`: include out-of-season
`date`	history only: ISO 8601 snapshot timestamp (required)

TIDY OUTPUT SHAPE

Odds come back **long**: one row per bookmaker × market × outcome, already unnested.

```
id  sport_key  commence_time
home_team  away_team
bookmaker_key  bookmaker
market_key  market_last_update
outcomes_name   # team/Over/Under
outcomes_price  # the odds
outcomes_point  # line/handicap
outcomes_description # player (props)
```

History adds `timestamp`, `previous_timestamp`, `next_timestamp`.

QUICK START

1 · Discover (free)

```
library(oddsapiR)
sports <- toa_sports(all_sports = FALSE)
# key: americanfootball_nfl,
# basketball_nba, soccer_epl, ...
```

2 · Featured lines for a sport

```
nba <- toa_sports_odds(
  sport_key   = "basketball_nba",
  regions     = "us",
  markets     = "h2h,spreads,totals",
  odds_format = "decimal",
  date_format = "iso")
```

3 · Player props for one event

```
ev <- toa_sports_events(
  sport_key = "basketball_nba")
toa_event_odds(
  sport_key = "basketball_nba",
  event_id  = ev$id[1],
  regions   = "us",
  markets   = "player_points")
```

SCORES

```
toa_sports_scores(
  sport_key = "basketball_nba",
  days_from = 3)  # adds finished games
```

Live scores refresh ~every 30 s; `completed` flags finals.

SPORTSDATAVERSE LINKS

Python mirror: `sportsdataverse.odds` in sportsdataverse-py — same `toa_*` names (`toa_requests` → `toa_usage`).

Private capture repo `sportsdataverse/odds-data` archives raw snapshots (7 sports, 2020+) with an ESPN event crosswalk.

Join odds to PBP from **cfbfastR**, **nflreadr**, **hoopR**, **wehoop** — recipes on page 2.

Keys, history & recipes :: markets · snapshots · joins

Join targets: **cfbfastR** · **nflreadr** · **hoopR** · **wehoop** · Python: `sportsdataverse.odds`

SPORT_KEY VALUES

COMMON SPORT KEYS

sport_key	Title
`americanfootball_nfl`	NFL
`americanfootball_ncaaf`	NCAAF
`basketball_nba`	NBA
`basketball_wnba`	WNBA
`basketball_ncaab`	NCAAB (M)
`baseball_mlb`	MLB
`icehockey_nhl`	NHL
`soccer_epl`	EPL
`soccer_usa_mls`	MLS
`mma_mixed_martial_arts`	MMA

The authoritative, current list is always one free call away:

```
toa_sports(all_sports = TRUE) |>
  dplyr::select(key, group,
                title, active)
```

Outright-only competitions (golf majors, futures) show `has_outrights = TRUE`; query them with `markets = "outrights"`.

MARKET KEYS

FEATURED MARKETS

Key	Market
`h2h`	moneyline / match winner
`spreads`	points handicap
`totals`	over / under
`outrights`	futures (select sports)

These four work on `toa_sports_odds()`. Everything below needs the per-event endpoints.

Player props (examples)

`player_points`	NBA/WNBA points O/U
`player_rebounds`	rebounds O/U
`player_pass_tds`	NFL passing TDs
`player_shots_on_goal`	NHL shots on goal

Props return the player in `outcomes_description`, the line in `outcomes_point`. Alternate lines (e.g. alt spreads / totals) are also event-endpoint markets. Full catalog: the-odds-api.com betting-markets docs.

What is live right now?

```
toa_event_markets(
  sport_key = "basketball_nba",
  event_id  = ev$id[1],
  regions   = "us")  # 1 credit
# one row per bookmaker x market_key
```

HISTORICAL ODDS

SNAPSHOT MODEL

Paid-plan endpoints. Supply an ISO 8601 `date`; you get the state at the closest snapshot **at or before** that time, plus `timestamp` / `previous_timestamp` / `next_timestamp` for paging.

```
hist <- toa_sports_odds_history(
  sport_key = "basketball_nba",
  date    = "2025-01-14T12:00:00Z",
  regions = "us", markets = "h2h")
# cost: 10 x markets x regions
```

Closing-line capture loop

```
# walk snapshots up to tip-off
d <- "2025-01-14T00:00:00Z"
snaps <- list()
for (i in 1:6) {
  s <- toa_event_odds_history(
    sport_key = "basketball_nba",
    event_id  = ev$id[1],
    date = d, regions = "us",
    markets = "h2h")
  snaps[[i]] <- s
  d <- s$next_timestamp[1]
}
line_move <- dplyr::bind_rows(snaps)
# last snap before commence_time
# = the closing line
```

`toa_event_odds_history()` bills at the standard rate (not 10x); empty snapshots cost 0.
`toa_sports_events_history()`: 1 credit.

RECIPES

IMPLIED PROB & DE-VIG

```
h2h |>
 dplyr::group_by(id, bookmaker_key) |>
 dplyr::mutate(
  p_raw  = 1 / outcomes_price,
  vig    = sum(p_raw),
  p_fair = p_raw / vig)
```

`vig` of 1.05 = a 5% bookmaker margin. Line-shop with `slice_max(outcomes_price)` per outcome.

JOIN TO SCHEDULES

```
sched <- cfbfastR::cfbd_game_info(2024) |>
  dplyr::mutate(
    kick = as.Date(start_date))
# or nflreadr::load_schedules()
odds |> dplyr::mutate(
  kick = as.Date(commence_time)) |>
 dplyr::left_join(sched,
  by = dplyr::join_by(home_team, kick))
```

Team names differ by source — normalize first (e.g. a name crosswalk); match on team + date, not name alone.

CREDIT BUDGETING

- Discovery is free: `toa_sports()`, `toa_sports_events()`, `toa_requests()`.
- Cost = markets × regions — `"h2h,spreads,totals"` × `"us,uk"` = 6 credits per call.
- Prefer `bookmakers=` to whole regions: 10 books count as 1 region.
- Check `toa_quota()$requests_last` after a new call shape to learn its real cost.
- Gate scheduled pulls: skip when `toa_requests()$requests_remaining` runs low.

oddsapiR :: CHEAT SHEET

Clean, tidy sports odds from **The Odds API** (the-odds-api.com): live and upcoming lines from dozens of bookmakers across US / UK / EU / AU regions, scores, player props, and historical odds snapshots. Part of the **SportsDataverse**; Python mirror: `sportsdataverse.odds`. Page 1 of 2: setup, the 15-function surface, arguments, quota.

v1.0.0

R ≥ 4.0.0 · httr2 · tidyr
oddsapiR.sportsdataverse.org

INSTALL

```
install.packages("oddsapiR")
# or the dev version:
pak::pak("sportsdataverse/oddsapiR")
library(oddsapiR)
```

API KEY SETUP

Every endpoint needs a (free-tier) key from **the-odds-api.com**, read from the `ODDS_API_KEY` env var.

Persistent (recommended)

```
usethis::edit_r_environ()
# paste in .Renviron, NO quotes:
ODDS_API_KEY = XXXX-YOUR-KEY-XXXX
# save, then restart R
```

Per session

```
Sys.setenv(
  ODDS_API_KEY = "XXXX-YOUR-KEY-XXXX")
```

Key helpers

`toa_key()`	the key, or `NA` if unset
`has_toa_key()`	`TRUE` / `FALSE`
`check_toa_key()`	errors loudly if unset

Docs: `?register_toa`.

QUOTA & CREDITS

Every response carries `x-requests-remaining` / `-used` / `-last` headers. Two ways to watch them:

```
toa_requests() # live check, FREE call
#   requests_remaining  requests_used
toa_quota()     # cached from the last
                # call - no network hit;
                # adds requests_last
```

Quota resets monthly. Cost of a call: **markets × regions** (odds), 10× that for historical odds. See the cost column → and page 2 for budgeting.

FUNCTION INDEX

Account & usage

Function	Cost
`toa_key` · `has_toa_key` · `check_toa_key`	—
`toa_requests()` — usage from live headers	free
`toa_quota()` — cached usage, offline	free

Sports & events

`toa_sports()` — coverage; the `key` col IS `sport_key`	free
`toa_sports_events()` — upcoming events + `event_id`	free
`toa_sports_participants()` — teams/players + ids	1
`toa_sports_scores()` — live + last-3-days scores	1–2

Odds & markets

`toa_sports_odds()` — featured markets, all games	mkt×reg
`toa_event_odds()` — one event; props + alternates	mkt×reg
`toa_event_markets()` — market keys live for an event	1

Historical (paid plans)

`toa_sports_odds_history()` — featured odds snapshot	10×mkt×reg
`toa_event_odds_history()` — event snapshot, props ok	mkt×reg
`toa_sports_events_history()` — events at a snapshot	1

All 15 exports return tibbles (class `oddsapiR_data`) and fail soft: on error you get an empty frame + a `cli` alert, never an exception.

ARGUMENT REFERENCE

Arg	Meaning
`sport_key`	from `toa_sports()$key`, e.g. `basketball_nba`
`event_id`	from `toa_sports_events()$id`
`regions`	`us us2 uk eu au`; comma-sep; each = 1 quota unit
`markets`	`h2h spreads totals outrights` (+ props on event endpoints); comma-sep
`odds_format`	`decimal` (default) | `american`
`date_format`	`iso` (default) | `unix`
`bookmakers`	comma-sep slugs (`draftkings`, `fanduel`); overrides `regions`; 10 books = 1 region of quota
`days_from`	scores: completed games 1–3 days back
`event_ids`	filter events/odds to listed ids
`commence_time_from` / `_to`	ISO 8601 window, e.g. `2023-09-09T00:00:00Z`
`include_rotation_numbers`	events: add `home/away_rotation`
`all_sports`	`toa_sports()`: include out-of-season
`date`	history only: ISO 8601 snapshot timestamp (required)

TIDY OUTPUT SHAPE

Odds come back **long**: one row per bookmaker × market × outcome, already unnested.

```
id  sport_key  commence_time
home_team  away_team
bookmaker_key  bookmaker
market_key  market_last_update
outcomes_name    # team/Over/Under
outcomes_price   # the odds
outcomes_point   # line/handicap
outcomes_description # player (props)
```

History adds `timestamp`, `previous_timestamp`, `next_timestamp`.

QUICK START

1 · Discover (free)

```
library(oddsapiR)
sports <- toa_sports(all_sports = FALSE)
# key: americanfootball_nfl,
# basketball_nba, soccer_epl, ...
```

2 · Featured lines for a sport

```
nba <- toa_sports_odds(
  sport_key   = "basketball_nba",
  regions     = "us",
  markets     = "h2h,spreads,totals",
  odds_format = "decimal",
  date_format = "iso")
```

3 · Player props for one event

```
ev <- toa_sports_events(
  sport_key = "basketball_nba")
toa_event_odds(
  sport_key = "basketball_nba",
  event_id  = ev$id[1],
  regions   = "us",
  markets   = "player_points")
```

SCORES

```
toa_sports_scores(
  sport_key = "basketball_nba",
  days_from = 3)  # adds finished games
```

Live scores refresh ~every 30 s; `completed` flags finals.

SPORTSDATAVERSE LINKS

Python mirror: `sportsdataverse.odds` in **sportsdataverse-py** — same `toa_*` names (`toa_requests` → `toa_usage`).

Private capture repo `sportsdataverse/odds-data` archives raw snapshots (7 sports, 2020+) with an ESPN event crosswalk.

Join odds to PBP from **cfbfastR**, **nflreadr**, **hoopR**, **wehoop** — recipes on page 2.

Keys, history & recipes :: markets · snapshots · joins

Join targets: **cfbfastR** · **nflreadr** · **hoopR** · **wehoop** · Python: `sportsdataverse.odds`

SPORT_KEY VALUES

COMMON SPORT KEYS

sport_key	Title
`americanfootball_nfl`	NFL
`americanfootball_ncaaf`	NCAAF
`basketball_nba`	NBA
`basketball_wnba`	WNBA
`basketball_ncaab`	NCAAB (M)
`baseball_mlb`	MLB
`icehockey_nhl`	NHL
`soccer_epl`	EPL
`soccer_usa_mls`	MLS
`mma_mixed_martial_arts`	MMA

The authoritative, current list is always one free call away:

```
toa_sports(all_sports = TRUE) |>
  dplyr::select(key, group,
                title, active)
```

Outright-only competitions (golf majors, futures) show `has_outrights = TRUE`; query them with `markets = "outrights"`.

MARKET KEYS

FEATURED MARKETS

Key	Market
`h2h`	moneyline / match winner
`spreads`	points handicap
`totals`	over / under
`outrights`	futures (select sports)

These four work on `toa_sports_odds()`. Everything below needs the per-event endpoints.

Player props (examples)

`player_points`	NBA/WNBA points O/U
`player_rebounds`	rebounds O/U
`player_pass_tds`	NFL passing TDs
`player_shots_on_goal`	NHL shots on goal

Props return the player in `outcomes_description`, the line in `outcomes_point`. Alternate lines (e.g. alt spreads / totals) are also event-endpoint markets. Full catalog: the-odds-api.com betting-markets docs.

What is live right now?

```
toa_event_markets(
  sport_key = "basketball_nba",
  event_id  = ev$id[1],
  regions   = "us")  # 1 credit
# one row per bookmaker x market_key
```

HISTORICAL ODDS

SNAPSHOT MODEL

Paid-plan endpoints. Supply an ISO 8601 `date`; you get the state at the closest snapshot **at or before** that time, plus `timestamp` / `previous_timestamp` / `next_timestamp` for paging.

```
hist <- toa_sports_odds_history(
  sport_key = "basketball_nba",
  date    = "2025-01-14T12:00:00Z",
  regions = "us", markets = "h2h")
# cost: 10 x markets x regions
```

Closing-line capture loop

```
# walk snapshots up to tip-off
d <- "2025-01-14T00:00:00Z"
snaps <- list()
for (i in 1:6) {
  s <- toa_event_odds_history(
    sport_key = "basketball_nba",
    event_id  = ev$id[1],
    date = d, regions = "us",
    markets = "h2h")
  snaps[[i]] <- s
  d <- s$next_timestamp[1]
}
line_move <- dplyr::bind_rows(snaps)
# last snap before commence_time
# = the closing line
```

`toa_event_odds_history()` bills at the standard rate (not 10x); empty snapshots cost 0.
`toa_sports_events_history()`: 1 credit.

RECIPES

IMPLIED PROB & DE-VIG

```
h2h |>
 dplyr::group_by(id, bookmaker_key) |>
 dplyr::mutate(
  p_raw  = 1 / outcomes_price,
  vig    = sum(p_raw),
  p_fair = p_raw / vig)
```

`vig` of 1.05 = a 5% bookmaker margin. Line-shop with `slice_max(outcomes_price)` per outcome.

JOIN TO SCHEDULES

```
sched <- cfbfastR::cfbd_game_info(2024) |>
  dplyr::mutate(
    kick = as.Date(start_date))
# or nflreadr::load_schedules()
odds |> dplyr::mutate(
  kick = as.Date(commence_time)) |>
 dplyr::left_join(sched,
  by = dplyr::join_by(home_team, kick))
```

Team names differ by source — normalize first (e.g. a name crosswalk); match on team + date, not name alone.

CREDIT BUDGETING

- Discovery is free: `toa_sports()`, `toa_sports_events()`, `toa_requests()`.
- Cost = markets × regions — `"h2h,spreads,totals"` × `"us,uk"` = 6 credits per call.
- Prefer `bookmakers=` to whole regions: 10 books count as 1 region.
- Check `toa_quota()$requests_last` after a new call shape to learn its real cost.
- Gate scheduled pulls: skip when `toa_requests()$requests_remaining` runs low.