

Package: sharpapi (via r-universe)

September 24, 2026

Title Client for the 'SharpAPI' Real-Time Sports Betting Odds API

Version 0.1.1

Description Access real-time sports betting odds, no-vig fair odds, expected value (+EV) and arbitrage opportunities from the 'SharpAPI' REST API <<https://sharpapi.io>>. Covers 45+ sportsbooks normalized into one schema. Requires a 'SharpAPI' key; a free tier is available.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

Imports httr2 (>= 1.2.0), jsonlite

Suggests httpuv, knitr, rmarkdown, testthat (>= 3.0.0), withr

VignetteBuilder knitr

Config/testthat/edition 3

URL <https://sharpapi.io>, <https://github.com/Sharp-API/SharpAPI-R>

BugReports <https://github.com/Sharp-API/SharpAPI-R/issues>

Config/roxygen2/version 8.1.0

Config/pak/sysreqs libssl-dev

Repository <https://sharp-api.r-universe.dev>

Date/Publication 2026-09-24 10:37:28 UTC

RemoteUrl <https://github.com/Sharp-API/SharpAPI-R>

RemoteRef HEAD

RemoteSha 46056a0b81e5dd0730e011e51133da43534a3ad4

Contents

sharpapi_arbitrage	2
sharpapi_ev	3
sharpapi_odds	4
sharpapi_sports	5

Index**6**

sharpapi_arbitrage *Arbitrage opportunities*

Description

Requires a Hobby tier key or higher.

Usage

```
sharpapi_arbitrage(...)
```

Arguments

... Filters passed to the API.

Value

A data frame with one row per arbitrage opportunity. Columns include event_id, sport, market_type and profit_percent. The individual sides live in a nested legs column rather than as top-level fields. A key below the Hobby tier raises an HTTP error rather than returning an empty frame.

Examples

```
# Toy example. A canned response stands in for the API, so this block runs
# offline -- no key, no network -- and returns the same shape a live call does.
if (requireNamespace("withr", quietly = TRUE)) {
  json <- '{"data": [
    {"event_id": "evt-1", "sport": "baseball",
     "market_type": "moneyline", "profit_percent": 1.8}
  ]}'
  mock <- function(req) {
    htr2::response(
      status_code = 200L,
      headers = list(`Content-Type` = "application/json"),
      body = charToRaw(json)
    )
  }
  withr::with_envvar(c(SharpAPI_KEY = "toy-key"), {
    htr2::with_mocked_responses(mock, sharpapi_arbitrage(sport = "baseball"))
  })
}

if (nzchar(Sys.getenv("SHARPAPI_KEY"))) {
  arb <- sharpapi_arbitrage(sport = "baseball")
  head(arb)
}
```

sharpapi_ev

*Positive expected value opportunities***Description**

Requires a Pro tier key or higher.

Usage

```
sharpapi_ev(...)
```

Arguments

... Filters passed to the API.

Value

A data frame with one row per +EV opportunity. Columns include event_id, sportsbook, market_type, selection and ev_percentage. A key below the Pro tier raises an HTTP error rather than returning an empty frame.

Examples

```
# Toy example. A canned response stands in for the API, so this block runs
# offline -- no key, no network -- and returns the same shape a live call does.
if (requireNamespace("withr", quietly = TRUE)) {
  json <- '{"data": [
    {"event_id": "evt-1", "sportsbook": "draftkings",
     "market_type": "moneyline", "selection": "Yankees",
     "ev_percentage": 3.42}
  ]}'
  mock <- function(req) {
    htr2::response(
      status_code = 200L,
      headers = list(`Content-Type` = "application/json"),
      body = charToRaw(json)
    )
  }
  withr::with_envvar(c(SHARPAPI_KEY = "toy-key"), {
    htr2::with_mocked_responses(mock, sharpapi_ev(sport = "baseball"))
  })
}

if (nzchar(Sys.getenv("SHARPAPI_KEY"))) {
  ev <- sharpapi_ev(sport = "baseball")
  head(ev)
}
```

sharpapi_odds	<i>Current odds as a data frame</i>
---------------	-------------------------------------

Description

Current odds as a data frame

Usage

```
sharpapi_odds(...)
```

Arguments

... Filters passed to the API, e.g. sport = "soccer", league = "mlb", sportsbook = "pinnacle", market_type = "moneyline", limit = 500.

Value

A data frame with one row per odds line. Columns include event_id, sportsbook, market_type, selection, odds_american, odds_decimal and odds_probability.

Examples

```
# Toy example. A canned response stands in for the API, so this block runs
# offline -- no key, no network -- and returns the same shape a live call does.
if (requireNamespace("withr", quietly = TRUE)) {
  json <- '{"data": [
    { "event_id": "evt-1", "sportsbook": "pinnacle",
      "market_type": "moneyline", "selection": "Yankees",
      "odds_american": -145, "odds_decimal": 1.69,
      "odds_probability": 0.5918 }
  ]}'
  mock <- function(req) {
    htr2::response(
      status_code = 200L,
      headers = list(`Content-Type` = "application/json"),
      body = charToRaw(json)
    )
  }
  withr::with_envvar(c(SHARPAPI_KEY = "toy-key"), {
    htr2::with_mocked_responses(mock, sharpapi_odds(sport = "baseball", limit = 5))
  })
}

if (nzchar(Sys.getenv("SHARPAPI_KEY"))) {
  odds <- sharpapi_odds(sport = "baseball", limit = 5)
  head(odds)
}
```

sharpapi_sports	<i>List sports with live event counts</i>
-----------------	---

Description

List sports with live event counts

Usage

```
sharpapi_sports()
```

Value

A data frame with one row per sport. Columns include `id`, `name`, `numerical_id`, `event_count` and `live_count`.

Examples

```
# Toy example. A canned response stands in for the API, so this block runs
# offline -- no key, no network -- and returns the same shape a live call does.
if (requireNamespace("withr", quietly = TRUE)) {
  json <- '{"data": [
    {"id": "baseball", "name": "Baseball", "numerical_id": 3,
     "event_count": 51, "live_count": 4}
  ]}'
  mock <- function(req) {
    htr2::response(
      status_code = 200L,
      headers = list(`Content-Type` = "application/json"),
      body = charToRaw(json)
    )
  }
  withr::with_envvar(c(SharpAPI_KEY = "toy-key"), {
    htr2::with_mocked_responses(mock, sharpapi_sports())
  })
}

# Requires SHARPAPI_KEY to be set; get a free key at https://sharpapi.io
if (nzchar(Sys.getenv("SHARPAPI_KEY"))) {
  sports <- sharpapi_sports()
  head(sports)
}
```

Index

sharpapi_arbitrage, [2](#)
sharpapi_ev, [3](#)
sharpapi_odds, [4](#)
sharpapi_sports, [5](#)