Package 'cryptotrackr'

May 8, 2026

Type Package

Title An Interface to Crypto Data Sources

Version 1.3.3

Maintainer Trevor French <FrenchTrevor@outlook.com>

Description Allows you to connect to data sources across the crypto ecosystem. This data can enable a range of activity such as portfolio tracking, programmatic trading, or industry analysis. The package is described in French (2024) <https://github.com/TrevorFrench/cryptotrackr/wiki>.

License MIT + file LICENSE

Encoding UTF-8

Imports httr, jsonlite, stringi, openssl, utils, digest

RoxygenNote 7.2.1

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Trevor French [aut, cre] (ORCID: <https://orcid.org/0000-0002-6246-2249>)

Repository CRAN

Date/Publication 2024-06-09 06:20:05 UTC

Contents

`amberdata_api_call` *amberdata_api_call*

Description

amberdata_api_call

Usage

```
amberdata_api_call(url, api_key, method, timeout_seconds = 60, query = NULL)
```

Arguments

`url`	the url for your Amberdata API call
`api_key`	your Amberdata API key
`method`	"GET" or "POST"
`timeout_seconds`	seconds until the query times out. Default is 60.
`query`	your query parameters. The default value is NULL.

Value

returns data from your Amberdata API call

Examples

```
## Not run:
api_key <- "..."
url <- "https://web3api.io/api/v2/market/exchanges"
method <- "GET"
exchanges <- amberdata_api_call(url, api_key, method)
## End(Not run)
```

amberdata_blockchain_metrics

amberdata_blockchain_metrics

Description

amberdata_blockchain_metrics

Usage

```
amberdata_blockchain_metrics(
  api_key,
  blockchain_id = "ethereum-mainnet",
  timeout_seconds = 60
)
```

Arguments

api_key	your Amberdata API key
blockchain_id	the id for the blockchain you wish to query. The default blockchain_id is "ethereum-mainnet".
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a list containing blockchain metrics for your specified blockchain_id.

Examples

```
## Not run:
api_key <- "..."
metrics <- amberdata_blockchain_metrics(api_key)
## End(Not run)
```

amberdata_historical_exchange_volume

amberdata_historical_exchange_volume

Description

amberdata_historical_exchange_volume

Usage

```
amberdata_historical_exchange_volume(
  api_key,
  exchange = NULL,
  direction = NULL,
  start_date = NULL,
  end_date = NULL,
  time_format = NULL,
  timeout_seconds = 60
)
```

Arguments

api_key	your Amberdata API key
exchange	the exchange you wish to retrieve volume for. By default this function will return all exchange volume.
direction	the direction in which to sort your results (by volume). The default is descending order. This function will accept "desc" or "asc".
start_date	only include data after this date (inclusive). This parameter will accept dates formatted as seconds, milliseconds, or iso8601.
end_date	only include data before this date (exclusive). The maximum time range is 31 days. This parameter will accept dates formatted as seconds, milliseconds, or iso8601.
time_format	the format to return your times in. Choose from: "milliseconds", "ms", "iso", "iso8601", "hr", and "human_readable". Default is "ms".
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a dataframe with your volume data.

Examples

```
## Not run:
api_key <- "..."
volume <- amberdata_historical_exchange_volume(api_key
                                               , "gdax"
                                               , "desc"
                                               , "2020-09-01T01:00:00"
                                               , "2020-09-02T01:00:00"
                                               , "hr")
## End(Not run)
```

amberdata_market_metrics *amberdata_market_metrics*

Description

amberdata_market_metrics

Usage

```
amberdata_market_metrics(api_key, symbol, timeout_seconds = 60)
```

Arguments

api_key	your Amberdata API key
symbol	the asset symbol you wish to receive metrics for
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a list containing market metrics for the specified symbol.

Examples

```
## Not run:
api_key <- "..."
metrics <- amberdata_market_metrics(api_key, "btc")
## End(Not run)
```

amberdata_spot_exchanges *amberdata_spot_exchanges*

Description

amberdata_spot_exchanges

Usage

```
amberdata_spot_exchanges(
  api_key,
  exchange = NULL,
  pair = NULL,
  include_dates = "false",
  time_format = "ms",
  timeout_seconds = 60
)
```

Arguments

api_key	your Amberdata API key
exchange	choose a specific exchange or multiple exchanges (comma-separated) rather than all exchanges
pair	choose a specific pair or multiple pairs (comma-separated) rather than all pairs
include_dates	include a start date and an end date along with your data. Default is "false"
time_format	the format to return your times in. Choose from: "milliseconds", "ms", "iso", "iso8601", "hr", and "human_readable". Default is "ms".
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a list of spot exchanges and pairs supported on Amberdata with the option of including the dates each one was supported.

Examples

```
## Not run:
api_key <- "..."
exchanges <- amberdata_spot_exchanges(api_key)
gdax <- amberdata_spot_exchanges(api_key
                                 , "gdax"
                                 , "1inch_btc,ada_usd"
                                 , "true"
                                 , "hr")
## End(Not run)
```

---

amberdata_spot_pairs *amberdata_spot_pairs*

Description

amberdata_spot_pairs

Usage

```
amberdata_spot_pairs(
  api_key,
  exchange = NULL,
  pair = NULL,
  include_dates = "false",
  time_format = "ms",
  timeout_seconds = 60
)
```

Arguments

api_key	your Amberdata API key
exchange	choose a specific exchange or multiple exchanges (comma-separated) rather than all exchanges
pair	choose a specific pair or multiple pairs (comma-separated) rather than all pairs
include_dates	include a start date and an end date along with your data. Default is "false"
time_format	the format to return your times in. Choose from: "milliseconds", "ms", "iso", "iso8601", "hr", and "human_readable". Default is "ms".
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a list of spot pairs and exchanges supported on Amberdata with the option of including the dates each one was supported.

Examples

```
## Not run:
api_key <- "..."
pairs <- amberdata_spot_pairs(api_key)
btc_usd <- amberdata_spot_pairs(api_key, pair = "btc_usd")
## End(Not run)
```

---

amberdata_spot_reference

amberdata_spot_reference

Description

amberdata_spot_reference

Usage

```
amberdata_spot_reference(
  api_key,
  exchange = NULL,
  pair = NULL,
  include_inactive = "False",
  include_original_reference = "False",
  timeout_seconds = 60
)
```

Arguments

api_key	your Amberdata API key
exchange	choose a specific exchange or multiple exchanges (comma-separated) rather than all exchanges
pair	choose a specific pair or multiple pairs (comma-separated) rather than all pairs
include_inactive	If 'True', endpoint returns all pairs, including delisted ones. Default is 'False'.
include_original_reference	If 'True', endpoint returns originalReference. Default is 'False'.
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a list of reference information for each of the pairs on Amberdata.

Examples

```
## Not run:
api_key <- "..."
reference <- amberdata_spot_reference(api_key)
btc_usd <- amberdata_spot_reference(api_key, pair = "btc_usd")
## End(Not run)
```

---

binance_us_account_info

binance_us_account_info

Description

binance_us_account_info

Usage

```
binance_us_account_info(key, secret, timeout_seconds = 60)
```

Arguments

key	your Binance.US API key
secret	your Binance.US secret key
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a list containing information about your account

Examples

```
## Not run:
key <- "..."
secret <- "..."
account_info <- binance_us_account_info(key, secret, 4.5)
## End(Not run)
```

`binance_us_api_call` *binance_us_api_call*

Description

binance_us_api_call

Usage

```
binance_us_api_call(url, key, data, secret, timeout_seconds = 60)
```

Arguments

`url`	the base url and endpoint followed by '?' for your API call
`key`	your Binance.US API key
`data`	your URL encoded query parameters
`secret`	your Binance.US secret key
`timeout_seconds`	seconds until the query times out. Default is 60.

Value

executes an authenticated API call

Examples

```
## Not run:
key <- "..."
secret <- "..."
time <- binance_us_time()
data <- paste('timestamp=', time, sep = '')
url <- 'https://api.binance.us/api/v3/account'
data <- binance_us_api_call(url, key, data, secret)
## End(Not run)
```

`binance_us_ping` *binance_us_ping*

Description

binance_us_ping

Usage

```
binance_us_ping(timeout_seconds = 60)
```

Arguments

`timeout_seconds`	seconds until the query times out. Default is 60.

Value

returns a response from the Binance.US API server

Examples

```
binance_us_ping(4.5)
```

`binance_us_recent_trades` *binance_us_recent_trades*

Description

binance_us_recent_trades

Usage

```
binance_us_recent_trades(symbol, limit, timeout_seconds = 60)
```

Arguments

`symbol`	the trading pair for which you wish to retrieve data.
`limit`	the number of results to return. The maximum is 1,000.
`timeout_seconds`	seconds until the query times out. Default is 60.

Value

returns a dataframe containing the most recent trades executed for the designated currency pair on Binance US

Examples

```
symbol <- 'LTCBTC'
limit <- '1000'
binance_us_recent_trades(symbol, limit, 4.5)
```

`binance_us_server_time` *binance_us_server_time*

Description

binance_us_server_time

Usage

```
binance_us_server_time(timeout_seconds = 60)
```

Arguments

`timeout_seconds`	seconds until the query times out. Default is 60.

Value

returns the Binance.US server time

Examples

```
binance_us_server_time(4.5)
```

`binance_us_signature` *binance_us_signature*

Description

binance_us_signature

Usage

```
binance_us_signature(data, secret)
```

Arguments

`data`	your URL encoded query parameters
`secret`	your Binance.US secret key

Value

returns your Binance.US signature for use in API calls

Examples

```
## Not run:
time <- binance_us_time()
data <- paste('timestamp=', time, sep = '')
secret <- "..."
signature <- binance_us_signature(data, secret)
## End(Not run)
```

`binance_us_system_status` *binance_us_system_status*

Description

binance_us_system_status

Usage

```
binance_us_system_status(key, secret, timeout_seconds = 60)
```

Arguments

`key`	your Binance.US API key
`secret`	your Binance.US secret key
`timeout_seconds`	seconds until the query times out. Default is 60.

Value

returns the status of the Binance.US API. The response will either be a "0" for normal or a "1" for system maintenance.

Examples

```
## Not run:
key <- "..."
secret <- "..."
system_status <- binance_us_system_status(key, secret)
## End(Not run)
```

`binance_us_time` *binance_us_time*

Description

binance_us_time

Usage

```
binance_us_time()
```

Value

returns a timestamp in the format that Binance.US expects

Examples

```
binance_us_time()
```

`blockchain_dot_com_l2_order_book` *blockchain_dot_com_l2_order_book*

Description

blockchain_dot_com_l2_order_book

Usage

```
blockchain_dot_com_l2_order_book(symbol, timeout_seconds = 60)
```

Arguments

`symbol`	the symbol for which to retrieve data
`timeout_seconds`	seconds until the query times out. Default is 60.

Value

returns a list containing a 'bids' and an 'asks' dataframe along with the specified symbol

Examples

```
symbol <- 'BTC-USD'
l2_order_book <- blockchain_dot_com_l2_order_book(symbol, 4.5)
l2_order_book$bids
l2_order_book$asks
```

blockchain_dot_com_l3_order_book

blockchain_dot_com_l3_order_book

Description

blockchain_dot_com_l3_order_book

Usage

```
blockchain_dot_com_l3_order_book(symbol, timeout_seconds = 60)
```

Arguments

symbol	the symbol for which to retrieve data
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a list containing a 'bids' and an 'asks' dataframe along with the specified symbol

Examples

```
symbol <- 'BTC-USD'
l3_order_book <- blockchain_dot_com_l3_order_book(symbol, 4.5)
l3_order_book$bids
l3_order_book$asks
```

blockchain_dot_com_symbol

blockchain_dot_com_symbol

Description

blockchain_dot_com_symbol

Usage

```
blockchain_dot_com_symbol(symbol, timeout_seconds = 60)
```

Arguments

symbol	the symbol for which to retrieve data
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a list with various data for specified symbol

Examples

```
symbol <- 'BTC-USD'
blockchain_dot_com_symbol(symbol, 4.5)
```

`blockchain_dot_com_symbols` *blockchain_dot_com_symbols*

Description

blockchain_dot_com_symbols

Usage

```
blockchain_dot_com_symbols(timeout_seconds = 60)
```

Arguments

`timeout_seconds` seconds until the query times out. Default is 60.

Value

returns a list with various data for all symbols

Examples

```
blockchain_dot_com_symbols(4.5)
```

`blockchain_dot_com_tickers` *blockchain_dot_com_tickers*

Description

blockchain_dot_com_tickers

Usage

```
blockchain_dot_com_tickers(timeout_seconds = 60)
```

Arguments

`timeout_seconds`
: seconds until the query times out. Default is 60.

Value

returns a dataframe with price and volume data for all symbols

Examples

```
blockchain_dot_com_tickers(4.5)
```

`blockchain_dot_com_ticker_symbol`
blockchain_dot_com_ticker_symbol

Description

blockchain_dot_com_ticker_symbol

Usage

```
blockchain_dot_com_ticker_symbol(symbol, timeout_seconds = 60)
```

Arguments

`symbol`
: the symbol for which to retrieve data

`timeout_seconds`
: seconds until the query times out. Default is 60.

Value

returns a list with price and volume data for specified symbol

Examples

```
symbol <- 'BTC-USD'
blockchain_dot_com_ticker_symbol(symbol, 4.5)
```

cex_io_balance *cex_io_balance*

Description

cex_io_balance

Usage

```
cex_io_balance(username, api_key, api_secret, timeout_seconds = 60)
```

Arguments

username	your cex.io username
api_key	your cex.io api_key
api_secret	your cex.io api_secret
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a list with your balances for each currency

Examples

```
## Not run:
username <- "..."
api_key <- "..."
api_secret <- "..."
balances <- cex_io_balance(username, api_key, api_secret, 4.5)
## End(Not run)
```

cex_io_converter *cex_io_converter*

Description

cex_io_converter

Usage

```
cex_io_converter(symbol_1, symbol_2, amount, timeout_seconds = 60)
```

Arguments

symbol_1	the first currency in your pair
symbol_2	the second currency in your pair
amount	the currency amount to convert denominated in symbol_1
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns the converted amount denominated in symbol_2

Examples

```
symbol_1 <- 'btc'
symbol_2 <- 'usd'
amount <- '2.5'
cex_io_converter(symbol_1, symbol_2, amount, 4.5)
```

---

cex_io_currency_limits

cex_io_currency_limits

Description

cex_io_currency_limits

Usage

```
cex_io_currency_limits(timeout_seconds = 60)
```

Arguments

timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a dataframe with information about currency limits on CEX.io.

Examples

```
cex_io_currency_limits(4.5)
```

cex_io_last_price *cex_io_last_price*

Description

cex_io_last_price

Usage

```
cex_io_last_price(symbol_1, symbol_2, timeout_seconds = 60)
```

Arguments

symbol_1	the first currency in your pair
symbol_2	the second currency in your pair
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a list with the last price of your specified currency pair.

Examples

```
symbol_1 <- 'btc'
symbol_2 <- 'usd'
cex_io_last_price(symbol_1, symbol_2, 4.5)
```

cex_io_nonce *cex_io_nonce*

Description

cex_io_nonce

Usage

```
cex_io_nonce()
```

Value

returns a nonce for use in your signature

Examples

```
cex_io_nonce()
```

cex_io_ohlcv *cex_io_ohlcv*

Description

cex_io_ohlcv

Usage

```
cex_io_ohlcv(date, symbol_1, symbol_2, timeout_seconds = 60)
```

Arguments

date	the date for which to retrieve data
symbol_1	the first currency in your pair
symbol_2	the second currency in your pair
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a list containing open, high, low, close, volume data for the past minute, hour, and day

Examples

```
date <- '20220927'
symbol_1 <- 'btc'
symbol_2 <- 'usd'
cex_io_ohlcv(date, symbol_1, symbol_2, 4.5)
```

cex_io_signature *cex_io_signature*

Description

cex_io_signature

Usage

```
cex_io_signature(username, api_key, api_secret, nonce)
```

Arguments

username	your cex.io username
api_key	your cex.io api_key
api_secret	your cex.io api_secret
nonce	a nonce to use in your signature and request body

Value

returns a signature for use in your API call

Examples

```
## Not run:
nonce <- cex_io_nonce()
username <- "..."
api_key <- "..."
api_secret <- "..."
sig <- cex_io_signature(username, api_key, api_secret, nonce)
## End(Not run)
```

`cex_io_ticker` *cex_io_ticker*

Description

cex_io_ticker

Usage

```
cex_io_ticker(symbol_1, symbol_2, timeout_seconds = 60)
```

Arguments

`symbol_1`	the first currency in your pair
`symbol_2`	the second currency in your pair
`timeout_seconds`	seconds until the query times out. Default is 60.

Value

returns a list with basic trading information about your specified currency pair for the last 24 hours.

Examples

```
symbol_1 <- 'btc'
symbol_2 <- 'usd'
cex_io_ticker(symbol_1, symbol_2, 4.5)
```

coinbase_accounts *coinbase_accounts*

Description

coinbase_accounts

Usage

```
coinbase_accounts(
  api_key,
  api_secret,
  limit = NULL,
  cursor = NULL,
  timeout_seconds = 60
)
```

Arguments

api_key	your Coinbase API key
api_secret	your Coinbase API secret
limit	the maximum number of results to return. The maximum limit is 250 while the default value is 49.
cursor	Cursor used for pagination. When provided, the response returns responses after this cursor.
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a list with a dataframe with information about your Coinbase accounts along with your cursor for use in pagination.

Examples

```
## Not run:
api_key <- "..."
api_secret <- "..."
accounts <- coinbase_accounts(api_key, api_secret)
## End(Not run)
```

coinbase_all_currencies *coinbase_all_currencies*

Description

coinbase_all_currencies

Usage

```
coinbase_all_currencies(timeout_seconds = 60)
```

Arguments

timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a dataframe with information about all currencies known to Coinbase

Examples

```
coinbase_all_currencies(4.5)
```

coinbase_api_call *coinbase_api_call*

Description

coinbase_api_call

Usage

```
coinbase_api_call(
  api_key,
  api_secret,
  method,
  path,
  body,
  query = NULL,
  timeout_seconds = 60
)
```

Arguments

api_key	your Coinbase API key
api_secret	your Coinbase API secret
method	"GET" or "POST"
path	the path of your API call
body	the body of your API call
query	the query for your coinbase API call as a list
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns the response from your Coinbase API call

Examples

```
## Not run:
path <- "/api/v3/brokerage/accounts"
method <- "GET"
api_key <- "..."
api_secret <- "..."
body <- ""
data <- coinbase_api_call(api_key, api_secret, method, path, body)
## End(Not run)
```

---

coinbase_candles *coinbase_candles*

Description

coinbase_candles

Usage

```
coinbase_candles(
  api_key,
  api_secret,
  product_id,
  start,
  end,
  granularity,
  timeout_seconds = 60
)
```

Arguments

api_key	your Coinbase API key
api_secret	your Coinbase API secret
product_id	the trading pair.
start	timestamp for starting range of aggregations, in UNIX time.
end	timestamp for ending range of aggregations, in UNIX time.
granularity	time slice value for each candle. Options: "ONE_MINUTE", "FIVE_MINUTE", "FIFTEEN_MINUTE", "THIRTY_MINUTE", "ONE_HOUR", "TWO_HOUR", "SIX_HOUR", or "ONE_DAY"
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a dataframe with your Coinbase candle data.

Examples

```
## Not run:
api_key <- "..."
api_secret <- "..."
end <- coinbase_time()
end_timestamp <- as.POSIXct(end, origin = "1970-01-01", tz = "UTC")
start_timestamp <- end_timestamp - 20 * 60  # 20 minutes in seconds
start <- as.numeric(start_timestamp)
coinbase_candles(api_key, api_secret, 'BTC-USD', start, end, 'ONE_MINUTE')
## End(Not run)
```

---

coinbase_signature *coinbase_signature*

Description

coinbase_signature

Usage

```
coinbase_signature(api_secret, coinbase_time, method, path, body)
```

Arguments

api_secret	your Coinbase API secret
coinbase_time	a timestamp in the correct format according to Coinbase
method	"GET" or "POST"
path	the path of your API call
body	the body of your API call

Value

returns a signature for use in your Coinbase API calls

Examples

```
## Not run:
api_secret <- "..."
coinbase_time <- coinbase_time()
method <- "GET"
path <- "/api/v3/brokerage/accounts"
body <- ""
coinbase_signature <- coinbase_signature(api_secret, coinbase_time, method, path, body)
## End(Not run)
```

---

coinbase_single_currency

coinbase_single_currency

Description

coinbase_single_currency

Usage

```
coinbase_single_currency(currency, timeout_seconds = 60)
```

Arguments

currency	the currency id for the relevant asset
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a list with details related to the specified currency

Examples

```
currency <- 'btc'
coinbase_single_currency(currency, 4.5)
```

coinbase_time *coinbase_time*

Description

coinbase_time

Usage

```
coinbase_time()
```

Value

returns a timestamp for use in your Coinbase API calls

Examples

```
coinbase_time()
```

coingecko_categories *coingecko_categories*

Description

coingecko_categories

Usage

```
coingecko_categories(timeout_seconds = 60)
```

Arguments

timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a dataframe of all categories on CoinGecko.

Examples

```
coingecko_categories(4.5)
```

coingecko_coins *coingecko_coins*

Description

coingecko_coins

Usage

```
coingecko_coins(include_platform = NULL, timeout_seconds = 60)
```

Arguments

include_platform	optionally select either "true" or "false" to include platform contract tokens.
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a dataframe containing all coins on CoinGecko and their respective ids, symbols, and names

Examples

```
coingecko_coins(timeout_seconds = 4.5)
```

coingecko_global_data *coingecko_global_data*

Description

coingecko_global_data

Usage

```
coingecko_global_data(timeout_seconds = 60)
```

Arguments

timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a list containing high-level statistics about the cryptocurrency ecosystem.

Examples

```
coingecko_global_data(4.5)
```

coingecko_ping *coingecko_ping*

Description

coingecko_ping

Usage

```
coingecko_ping(timeout_seconds = 60)
```

Arguments

timeout_seconds	seconds until the query times out. Default is 60.

Value

returns the Coingecko API server status

Examples

```
coingecko_ping(4.5)
```

coingecko_price *coingecko_price*

Description

coingecko_price

Usage

```
coingecko_price(
  id,
  vs_currency,
  include_market_cap = NULL,
  include_24hr_vol = NULL,
  include_24hr_change = NULL,
  include_last_updated_at = NULL,
  precision = NULL,
  timeout_seconds = 60
)
```

Arguments

id	one or more comma-separated asset ids to query
vs_currency	one or more comma-separated vs_currencies to query
include_market_cap	optionally provide a 'true' or 'false' value to include/exclude market cap. The default is 'false'.
include_24hr_vol	optionally provide a 'true' or 'false' value to include/exclude 24-hour volume. The default is 'false'.
include_24hr_change	optionally provide a 'true' or 'false' value to include/exclude the 24-hour price change. The default is 'false'.
include_last_updated_at	optionally provide a 'true' or 'false' value to include/exclude the last updated information. The default is 'false'.
precision	optionally specify the decimal precision to return. Choose either 'full' or any number between 0 and 18.
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a list of currency prices

Examples

```
coingecko_price(id = 'bitcoin', vs_currency = 'usd', timeout_seconds = 4.5)
```

---

coingecko_price_history

coingecko_price_history

Description

coingecko_price_history

Usage

```
coingecko_price_history(id, date, localization = "false", timeout_seconds = 60)
```

Arguments

id	The asset id you wish to query. IDs can be retrieved with the coingecko_coins function.
date	the date you wish to query formatted as "dd-mm-yyyy"
localization	"true" or "false" to include/exclude localized languages in the response. The default value is "false".
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a list containing data about asset pricing.

Examples

```
price <- coingecko_price_history("bitcoin", "30-12-2017", timeout_seconds = 4.5)
price$market_data$current_price$usd
```

---

coingecko_vs_currencies

coingecko_vs_currencies

Description

coingecko_vs_currencies

Usage

```
coingecko_vs_currencies(timeout_seconds = 60)
```

Arguments

timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a character vector containing all supported currencies on Coingecko.

Examples

```
coingecko_vs_currencies(4.5)
```

coinlist_api_call *coinlist_api_call*

Description

coinlist_api_call

Usage

```
coinlist_api_call(
  api_key,
  api_secret,
  method,
  path,
  body,
  timeout_seconds = 60
)
```

Arguments

api_key	your Coinlist API key
api_secret	your Coinlist API secret
method	"GET" or "POST"
path	the path of your API call
body	the body of your API call
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns the response from your Coinlist API call

Examples

```
## Not run:
path <- "/v1/accounts"
method <- "GET"
api_key <- "..."
api_secret <- "..."
body <- ""
data <- coinlist_api_call(api_key, api_secret, method, path, body)
## End(Not run)
```

coinlist_fees *coinlist_fees*

Description

coinlist_fees

Usage

```
coinlist_fees(api_key, api_secret, timeout_seconds = 60)
```

Arguments

api_key	your Coinlist API key
api_secret	your Coinlist API secret
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a list containing Coinlist fees by symbols.

Examples

```
## Not run:
api_key <- "..."
api_secret <- "..."
fees <- coinlist_fees(api_key, api_secret)
## End(Not run)
```

coinlist_signature *coinlist_signature*

Description

coinlist_signature

Usage

```
coinlist_signature(api_secret, coinlist_time, method, path, body)
```

Arguments

api_secret	your Coinlist API secret
coinlist_time	a timestamp in the correct format according to Coinlist
method	"GET" or "POST"
path	the path of your API call
body	the body of your API call

Value

returns a signature for use in your Coinlist API calls

Examples

```
## Not run:
api_secret <- "..."
coinlist_time <- coinlist_time()
method <- "GET"
path <- "/v1/accounts"
body <- ""
coinlist_signature <- coinlist_signature(api_secret, coinlist_time, method, path, body)
## End(Not run)
```

---

coinlist_symbols *coinlist_symbols*

Description

coinlist_symbols

Usage

```
coinlist_symbols(timeout_seconds = 60)
```

Arguments

timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a dataframe with information about symbols available on Coinlist Pro

Examples

```
coinlist_symbols(4.5)
```

coinlist_time *coinlist_time*

Description

coinlist_time

Usage

```
coinlist_time()
```

Value

returns a timestamp for use in your Coinlist API calls

Examples

```
coinlist_time()
```

coinmarketcap_airdrop *coinmarketcap_airdrop*

Description

coinmarketcap_airdrop

Usage

```
coinmarketcap_airdrop(api_key, id, timeout_seconds = 60)
```

Arguments

api_key	your CoinMarketCap API key
id	the unique airdrop id which can be found through the airdrops api.
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns information about the airdrop for the id you provided.

Examples

```
## Not run:
api_key <- "..."
id <- "10744"
airdrop <- coinmarketcap_airdrop(api_key, id)
## End(Not run)
```

coinmarketcap_api_call *coinmarketcap_api_call*

Description

coinmarketcap_api_call

Usage

```
coinmarketcap_api_call(
  url,
  api_key,
  method,
  query = NULL,
  timeout_seconds = 60
)
```

Arguments

url	the url for your CoinMarketCap API call
api_key	your CoinMarketCap API key
method	"GET" or "POST"
query	your query parameters. The default value is NULL.
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns data from your CoinMarketCap API call

Examples

```
## Not run:
url <- "https://pro-api.coinmarketcap.com/v1/cryptocurrency/map"
api_key <- "..."
query_string <- list(
  listing_status = "active",
  start = "1",
  limit = NULL,
  sort = "id",
  symbol = NULL,
  aux = "platform,first_historical_data,last_historical_data,is_active,status"
)
data <- coinmarketcap_api_call(url, api_key, 'GET', query = query_string)
## End(Not run)
```

coinmarketcap_categories *coinmarketcap_categories*

Description

coinmarketcap_categories

Usage

```
coinmarketcap_categories(
  api_key,
  start = "1",
  limit = NULL,
  id = NULL,
  slug = NULL,
  symbol = NULL,
  timeout_seconds = 60
)
```

Arguments

api_key	your CoinMarketCap API key
start	you can use this parameter to offset your first result. The default value is "1".
limit	an optional string value between 1 and 5000 which tells CoinMarketCap how many results to return. The default value is NULL.
id	filter categories by one or more asset ids. The default value is NULL. Multiple values must be comma-separated.
slug	filter categories by one or more asset slugs. The default value is NULL. Multiple values must be comma-separated.
symbol	filter categories by one or more asset symbols. The default value is NULL. Multiple values must be comma-separated.
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a datafrane with information about CoinMarketCap asset categories.

Examples

```
## Not run:
api_key <- "..."
categories <- coinmarketcap_categories(api_key)
## End(Not run)
```

coinmarketcap_category *coinmarketcap_category*

Description

coinmarketcap_category

Usage

```
coinmarketcap_category(
  api_key,
  id,
  start = "1",
  limit = NULL,
  convert = NULL,
  convert_id = NULL,
  timeout_seconds = 60
)
```

Arguments

api_key	your CoinMarketCap API key
id	the category id you wish to query.
start	you can use this parameter to offset your first result. The default value is "1".
limit	an optional string value between 1 and 5000 which tells CoinMarketCap how many results to return. The default value is NULL.
convert	Optionally calculate market quotes in up to 120 currencies at once by passing a comma-separated list of cryptocurrency or fiat currency symbols.
convert_id	Optionally calculate market quotes by CoinMarketCap id instead of symbol.
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a list with information about the specified category.

Examples

```
## Not run:
api_key <- "..."
id <- "6363a6c9cd197958bb543bf0"
category <- coinmarketcap_category(api_key, id)
## End(Not run)
```

`coinmarketcap_id_map` *coinmarketcap_id_map*

Description

coinmarketcap_id_map

Usage

```
coinmarketcap_id_map(
  api_key,
  listing_status = "active",
  start = "1",
  limit = NULL,
  sort = "id",
  symbol = NULL,
  aux = "platform,first_historical_data,last_historical_data,is_active,status",
  timeout_seconds = 60
)
```

Arguments

`api_key`	your CoinMarketCap API key
`listing_status`	you can choose "active", "inactive", or "untracked". Multiple options can be passed if they are comma-separated. Choosing "active" will return only active cryptocurrencies. Choosing "inactive" will return cryptocurrencies which are inactive. Choosing "untracked" will return a list of cryptocurrencies which are listed by CoinMarketCap but do not yet meet their methodology requirements to have tracked markets available. The default is "active".
`start`	you can use this parameter to offset your first result. The default value is "1".
`limit`	an optional string value between 1 and 5000 which tells CoinMarketCap how many results to return. The default value is NULL.
`sort`	the field used to sort your results. The two acceptable values are "id" and "cmc_rank". The default value is "id".
`symbol`	Optionally pass a comma-separated list of cryptocurrency symbols to return CoinMarketCap IDs for. The default value is NULL.
`aux`	Optionally specify a comma-separated list of supplemental data fields to return. Pass "platform,first_historical_data,last_historical_data, is_active,status" to include all auxiliary fields. This function will include all auxiliary fields by default.
`timeout_seconds`	seconds until the query times out. Default is 60.

Value

returns a dataframe which includes the id mapping for CoinMarketCap cryptocurrencies along with other metadata related to the currencies.

Examples

```
## Not run:
api_key <- "..."
id_map <- coinmarketcap_id_map(api_key)
## End(Not run)
```

`coinmarketcap_metadata` *coinmarketcap_metadata*

Description

coinmarketcap_metadata

Usage

```
coinmarketcap_metadata(
  api_key,
  id = NULL,
  slug = NULL,
  symbol = NULL,
  address = NULL,
  aux = "urls,logo,description,tags,platform,date_added,notice,status",
  timeout_seconds = 60
)
```

Arguments

`api_key`	your CoinMarketCap API key
`id`	the id of the asset you wish to query. The default value is NULL; however, each request must include either an id, slug, symbol, or contract address. You can also pass multiple comma-separated values.
`slug`	the slug of the asset you wish to query. The default value is NULL. You can also pass multiple comma-separated values.
`symbol`	the symbol of the asset you wish to query. The default value is NULL. You can also pass multiple comma-separated values.
`address`	the contract address of the asset you wish to query. The default calue is NULL. You can also pass multiple comma-separated values.
`aux`	Optionally specify a comma-separated list of supplemental data fields to return. Pass "urls,logo,description,tags,platform,date_added, notice,status" to include all auxiliary fields. This function will include all auxiliary fields by default.
`timeout_seconds`	seconds until the query times out. Default is 60.

Value

returns a list which includes a dataframe for each asset you requested. The dataframe will contain CoinMarketCap metadata for the asset.

Examples

```
## Not run:
api_key <- "..."
metadata <- coinmarketcap_metadata(api_key, symbol = "BTC")
## End(Not run)
```

---

covalent_api_call *covalent_api_call*

Description

covalent_api_call

Usage

```
covalent_api_call(url, method, query = NULL, csv = FALSE, timeout_seconds = 60)
```

Arguments

url	the Covalent URL for use in your API call
method	'GET' or 'POST'
query	your query parameters formatted as a named list
csv	'TRUE' will return csv data parsed as a dataframe while 'FALSE' will return json data. The default value is 'FALSE'.
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns your Covalent API data

Examples

```
## Not run:
url <- "https://api.covalenthq.com/v1/1/address/trevorfrench.eth/balances_v2/"
api_key <- "..."
query <- list(key = api_key, format = NULL)
method <- "GET"
balances <- covalent_api_call(url, method, api_key, method, query, csv = FALSE)
## End(Not run)
```

covalent_balances *covalent_balances*

Description

covalent_balances

Usage

```
covalent_balances(
  api_key,
  chain_id,
  address,
  csv = FALSE,
  timeout_seconds = 60
)
```

Arguments

api_key	your Covalent API key
chain_id	the string id of the chain for which you wish to check balances.
address	the address you for which wish to check balances.
csv	'TRUE' will return csv data parsed as a dataframe while 'FALSE' will return json data. The default value is 'FALSE'.
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns either a list or a dataframe with account balances

Examples

```
## Not run:
api_key <- "..."
balances <- covalent_balances(api_key, "1", "trevorfrench.eth", csv = FALSE)
## End(Not run)
```

covalent_portfolio *covalent_portfolio*

Description

covalent_portfolio

Usage

```
covalent_portfolio(
  api_key,
  chain_id,
  address,
  csv = FALSE,
  timeout_seconds = 60
)
```

Arguments

api_key	your Covalent API key
chain_id	the string id of the chain for which you wish to check portfolio history.
address	the address you for which wish to get portfolio history.
csv	'TRUE' will return csv data parsed as a dataframe while 'FALSE' will return json data. The default value is 'FALSE'.
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns either a list or a dataframe with portfolio history

Examples

```
## Not run:
api_key <- "..."
portfolio <- covalent_portfolio(api_key, "1", "trevorfrench.eth", csv = FALSE)
## End(Not run)
```

crypto_dot_com_get_book *crypto_dot_com_get_book*

Description

crypto_dot_com_get_book

Usage

```
crypto_dot_com_get_book(instrument, depth = 50, timeout_seconds = 60)
```

Arguments

instrument	the instrument name which you want to query
depth	the depth of the order book to retrieve. The maximum and default value is 50.
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a list containing the order book for your specified instrument.

Examples

```
crypto_dot_com_get_book("BTC_USDT", timeout_seconds = 4.5)
```

crypto_dot_com_get_candlestick *crypto_dot_com_get_candlestick*

Description

crypto_dot_com_get_candlestick

Usage

```
crypto_dot_com_get_candlestick(
  instrument,
  timeframe = "5m",
  timeout_seconds = 60
)
```

Arguments

instrument	the instrument name which you want to query
timeframe	the timeframe which each candle represents. You can choose from the following options: '1m', '5m', '15m', '30m', '1h', '4h', '6h', '12h', '1D', '7D', '14D', '1M'. The defailt option is '5m'.
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a list which contains metadata about your query along with a dataframe containing your candlestick data.

Examples

```
crypto_dot_com_get_candlestick("BTC_USDT", timeout_seconds = 4.5)
```

---

crypto_dot_com_get_ticker

crypto_dot_com_get_ticker

Description

crypto_dot_com_get_ticker

Usage

```
crypto_dot_com_get_ticker(instrument, timeout_seconds = 60)
```

Arguments

instrument	the instrument name which you want to query
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns ticker data for specified instrument. Refer to Crypto.com for help interpreting response data: https://exchange-docs.crypto.com/spot/index.html#public-get-ticker

Examples

```
crypto_dot_com_get_ticker("BTC_USDT", 4.5)
```

crypto_dot_com_get_trades *crypto_dot_com_get_trades*

Description

crypto_dot_com_get_trades

Usage

```
crypto_dot_com_get_trades(instrument, timeout_seconds = 60)
```

Arguments

instrument	the instrument name which you want to query
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns trade data for specified instrument. Refer to Crypto.com for help interpreting response data: https://exchange-docs.crypto.com/spot/index.html#public-get-trades

Examples

```
crypto_dot_com_get_trades("BTC_USDT", 4.5)
```

crypto_dot_com_instruments *crypto_dot_com_instruments*

Description

crypto_dot_com_instruments

Usage

```
crypto_dot_com_instruments(timeout_seconds = 60)
```

Arguments

timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a dataframe with information about instruments available on Crypto.com

Examples

```
crypto_dot_com_instruments(4.5)
```

`etherscan_account_balance` *etherscan_account_balance*

Description

etherscan_account_balance

Usage

```
etherscan_account_balance(
  address,
  api_key,
  tag = "latest",
  timeout_seconds = 60
)
```

Arguments

`address`	the address for which you wish to retrieve the balance.
`api_key`	your Etherscan API key
`tag`	pre-defined block parameter, either earliest, pending or latest. Default is latest.
`timeout_seconds`	seconds until the query times out. Default is 60.

Value

returns the balance for the specified address

Examples

```
## Not run:
address <- "0xde0b295669a9fd93d5f28d9ec85e40f4cb697bae"
api_key <- "..."
account_balance <- etherscan_account_balance(address, api_key)
## End(Not run)
```

`etherscan_api_call` *etherscan_api_call*

Description

etherscan_api_call

Usage

```
etherscan_api_call(method, query, timeout_seconds = 60)
```

Arguments

`method`	"GET" or "POST"
`query`	your query parameters
`timeout_seconds`	seconds until the query times out. Default is 60.

Value

returns data from your Etherscan API call

Examples

```
## Not run:
address <- "0xde0b295669a9fd93d5f28d9ec85e40f4cb697bae"
api_key <- "..."
tag <- "latest"
query_string <- list(
   module = 'account',
   action = 'balance',
   address = address,
   tag = tag,
   apikey = api_key
)

data <- etherscan_api_call('GET', query_string)
## End(Not run)
```

etherscan_block_reward

etherscan_block_reward

Description

etherscan_block_reward

Usage

```
etherscan_block_reward(block, api_key, timeout_seconds = 60)
```

Arguments

block	the numeric block number
api_key	your Etherscan API key
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns the block and uncle reward for the specified block number as a list.

Examples

```
## Not run:
block <- 12697906
api_key <- "..."
block_reward <- etherscan_block_reward(block, api_key)
## End(Not run)
```

etherscan_contract_abi

etherscan_contract_abi

Description

etherscan_contract_abi

Usage

```
etherscan_contract_abi(address, api_key, timeout_seconds = 60)
```

Arguments

address	the contract address for which you wish to retrieve the ABI.
api_key	your Etherscan API key
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns the contract ABI for the specified address

Examples

```
## Not run:
address <- "0xfb6916095ca1df60bb79ce92ce3ea74c37c5d359"
api_key <- "..."
abi <- etherscan_contract_abi(address, api_key)
## End(Not run)
```

---

etherscan_gas_oracle *etherscan_gas_oracle*

Description

etherscan_gas_oracle

Usage

```
etherscan_gas_oracle(api_key, timeout_seconds = 60)
```

Arguments

api_key	your Etherscan API key
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns current safe, proposed and fast gas prices as determined by Etherscan.

Examples

```
## Not run:
api_key <- "..."
gas_oracle <- etherscan_gas_oracle(api_key)
## End(Not run)
```

gemini_api_call *gemini_api_call*

Description

gemini_api_call

Usage

```
gemini_api_call(key, secret, path, method, timeout_seconds = 60)
```

Arguments

key	your API key for Gemini
secret	your secret key for Gemini
path	your API endpoint
method	"GET" or "POST"
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns data from your Gemini API call

Examples

```
## Not run:
key <- "..."
secret <- "..."
path <- "/v1/mytrades"
method <- "POST"
data <- gemini_api_call(key, secret, path, method)
## End(Not run)
```

gemini_price_feed *gemini_price_feed*

Description

gemini_price_feed

Usage

```
gemini_price_feed(timeout_seconds = 60)
```

Arguments

`timeout_seconds`
: seconds until the query times out. Default is 60.

Value

returns a dataframe containing pairs, their current price, and their 24 hour change in price

Examples

```
gemini_price_feed(4.5)
```

`gemini_symbols` *gemini_symbols*

Description

gemini_symbols

Usage

```
gemini_symbols(timeout_seconds = 60)
```

Arguments

`timeout_seconds`
: seconds until the query times out. Default is 60.

Value

returns a vector containing all symbols available on Gemini

Examples

```
gemini_symbols(4.5)
```

gemini_trades *gemini_trades*

Description

gemini_trades

Usage

```
gemini_trades(key, secret, timeout_seconds = 60)
```

Arguments

key	your API key for Gemini
secret	your secret key for Gemini
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a dataframe containing all of your historical trades.

Examples

```
## Not run:
key <- "..."
secret <- "..."
df <- gemini_trades(key, secret)
## End(Not run)
```

huobi_candles *huobi_candles*

Description

huobi_candles

Usage

```
huobi_candles(period, size, symbol, timeout_seconds = 60)
```

Arguments

period	the period of each candle. The following are acceptable options: "1min", "5min", "15min", "30min", "60min", "4hour", "1day", "1mon", "1week", "1year"
size	the number of datapoints to return. This should fall between 1 and 2000.
symbol	the trading symbol to query.
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a dataframe containing Huobi candle data

Examples

```
period <- '1day'
size <- '200'
symbol <- 'btcusdt'
huobi_candles(period, size, symbol, 4.5)
```

---

kraken_asset_info *kraken_asset_info*

Description

kraken_asset_info

Usage

```
kraken_asset_info(asset = NULL, aclass = NULL, timeout_seconds = 60)
```

Arguments

asset	optionally provide one or more comma-separated ticker symbols.
aclass	optionally provide asset categories to filter by.
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a list containing asset information

Examples

```
all_asset_info <- kraken_asset_info(timeout_seconds = 4.5)
eth_btc_info <- kraken_asset_info("ETH,BTC", timeout_seconds = 4.5)
currency_info <- kraken_asset_info(aclass = "currency", timeout_seconds = 4.5)
```

kraken_asset_pairs *kraken_asset_pairs*

Description

kraken_asset_pairs

Usage

```
kraken_asset_pairs(pair = NULL, info = NULL, timeout_seconds = 60)
```

Arguments

pair	optionally provide one or more comma-separated asset pairs to query.
info	optionally select the information to return. You can choose from: "info" (all info), "leverage" (leverage info), "fees" (fee schedule), or "margin" (margin info).
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a list containing information on Kraken asset pairs.

Examples

```
kraken_asset_pairs(timeout_seconds = 4.5)
```

kraken_server_status *kraken_server_status*

Description

kraken_server_status

Usage

```
kraken_server_status(timeout_seconds = 60)
```

Arguments

timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a list with Kraken's server status along with a timestamp

Examples

```
kraken_server_status(4.5)
```

`kraken_server_time` *kraken_server_time*

Description

kraken_server_time

Usage

```
kraken_server_time(timeout_seconds = 60)
```

Arguments

`timeout_seconds`	seconds until the query times out. Default is 60.

Value

returns a list with Kraken's server time in unix and rfc1123 formats

Examples

```
kraken_server_time(4.5)
```

`kraken_ticker_info` *kraken_ticker_info*

Description

kraken_ticker_info

Usage

```
kraken_ticker_info(pair = NULL, timeout_seconds = 60)
```

Arguments

`pair`	optionally provide one or more comma-separated asset pairs.
`timeout_seconds`	seconds until the query times out. Default is 60.

Value

returns a list containing ticker info for assets on Kraken. Refer to Kraken for help interpreting response data: https://docs.kraken.com/rest/#tag/Market-Data/operation/getTickerInformation

Examples

```
kraken_ticker_info("ETHUSD", 4.5)
```

`kucoin_accounts` *kucoin_accounts*

Description

kucoin_accounts

Usage

```
kucoin_accounts(
  api_key,
  api_secret,
  passphrase,
  version = "2",
  timeout_seconds = 60
)
```

Arguments

`api_key`	your Kucoin API key.
`api_secret`	your Kucoin API secret.
`passphrase`	the passphrase you created when you created your Kucoin API key.
`version`	your API key version. This can be retrieved from your Kucoin API console. The default value is "2".
`timeout_seconds`	seconds until the query times out. Default is 60.

Value

returns a dataframe containing your Kucoin accounts and balances.

Examples

```
## Not run:
api_key <- "..."
api_secret <- "..."
passphrase <- "..."
accounts <- kucoin_accounts(api_key, api_secret, passphrase)
## End(Not run)
```

`kucoin_api_call` *kucoin_api_call*

Description

kucoin_api_call

Usage

```
kucoin_api_call(
  url,
  method,
  api_key,
  sig,
  time,
  passphrase,
  version,
  api_secret,
  query = NULL,
  timeout_seconds = 60
)
```

Arguments

`url`	the full url for your Kucoin API call
`method`	"GET" or "POST"
`api_key`	your Kucoin API key
`sig`	signature for use in your Kucoin API call. This can be generated with the "kucoin_signature" function.
`time`	a timestamp string formatted the way Kucoin requires. This can be created with the "kucoin_time" function.
`passphrase`	the passphrase you created when you created your Kucoin API key.
`version`	your API key version. This can be retrieved from your Kucoin API console.
`api_secret`	your Kucoin API secret.
`query`	a named list containing your query parameters.
`timeout_seconds`	seconds until the query times out. Default is 60.

Value

returns the data from your Kucoin API call.

Examples

```
## Not run:
url <- "https://api.kucoin.com/api/v1/sub/user"
api_key <- "..."
api_secret <- "..."
time <- kucoin_time()
method <- "GET"
path <- "/api/v1/sub/user"
body <- ""
sig <- kucoin_signature(api_secret, time, method, path, body)
passphrase <- "..."
version <- "2"

accounts <- kucoin_api_call(url, method, api_key, sig, time, passphrase,
version, api_secret)
## End(Not run)
```

`kucoin_signature` *kucoin_signature*

Description

kucoin_signature

Usage

```
kucoin_signature(api_secret, time, method, path, body)
```

Arguments

`api_secret`	your Kucoin API secret
`time`	a timestamp string formatted the way Kucoin requires. This can be created with the "kucoin_time" function.
`method`	"GET" or "POST"
`path`	the endpoint you are using to make an API call.
`body`	needs to be a json string which matches url parameters. Use a blank string if not applicable.

Value

returns a signature for use in you Kucoin API calls.

Examples

```
## Not run:
api_secret <- "..."
time <- kucoin_time()
method <- "GET"
path <- "/api/v1/sub/user"
body <- ""
sig <- kucoin_signature(api_secret, time, method, path, body)
## End(Not run)
```

`kucoin_subaccounts` *kucoin_subaccounts*

Description

kucoin_subaccounts

Usage

```
kucoin_subaccounts(
  api_key,
  api_secret,
  passphrase,
  version = "2",
  timeout_seconds = 60
)
```

Arguments

`api_key`	your Kucoin API key.
`api_secret`	your Kucoin API secret.
`passphrase`	the passphrase you created when you created your Kucoin API key.
`version`	your API key version. This can be retrieved from your Kucoin API console. The default value is "2".
`timeout_seconds`	seconds until the query times out. Default is 60.

Value

returns a list containing your Kucoin sub-accounts.

Examples

```
## Not run:
api_key <- "..."
api_secret <- "..."
passphrase <- "..."
accounts <- kucoin_subaccounts(api_key, api_secret, passphrase)
## End(Not run)
```

kucoin_symbols_list *kucoin_symbols_list*

Description

kucoin_symbols_list

Usage

```
kucoin_symbols_list(market = NULL, timeout_seconds = 60)
```

Arguments

market	optionally provide a market to filter on. This function will return all markets by default.
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a dataframe containing information about trading symbols

Examples

```
kucoin_symbols_list('btc', 4.5)
```

kucoin_time *kucoin_time*

Description

kucoin_time

Usage

```
kucoin_time()
```

Value

returns a timestamp formatted in the way it is required in order to make an API call to Kucoin.

Examples

```
kucoin_time()
```

`magic_eden_collection_stats` *magic_eden_collection_stats*

Description

magic_eden_collection_stats

Usage

```
magic_eden_collection_stats(symbol, timeout_seconds = 60)
```

Arguments

`symbol`	the collection symbol you are requesting data for
`timeout_seconds`	seconds until the query times out. Default is 60.

Value

returns a list containing statistics about the specified collection.

Examples

```
symbol <- "gothic_degens"
magic_eden_collection_stats(symbol, timeout_seconds = 4.5)
```

`magic_eden_tokens_owned` *magic_eden_tokens_owned*

Description

magic_eden_tokens_owned

Usage

```
magic_eden_tokens_owned(
  wallet,
  offset = NULL,
  limit = NULL,
  list_status = NULL,
  timeout_seconds = 60
)
```

Arguments

wallet	the address of the wallet you are trying to query
offset	optionally provide a numeric value to specify number of results to skip.
limit	optionally provide a numeric limit to specify maximum number of results.
list_status	either "listed", "unlisted" or "both". The default is "both".
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a dataframe containing all tokens owned by specified wallet.

Examples

```
wallet <- "72tXz6jhGVPFE8ZfAQocJPJU3HgxsdrRqKZoUdWUhs7o"
magic_eden_tokens_owned(wallet, timeout_seconds = 4.5)
```

---

magic_eden_token_listings *magic_eden_token_listings*

Description

magic_eden_token_listings

Usage

```
magic_eden_token_listings(mint_address, timeout_seconds = 60)
```

Arguments

mint_address	the mint address of the token you wish to query
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a dataframe containing the token listings for the specified mint address.

Examples

```
mint_address <- "Hd6sxFEEQQA5aURaWaDesi23AkM19bBkKave1hyWvnfS"
magic_eden_token_listings(mint_address, timeout_seconds = 4.5)
```

magic_eden_token_metadata *magic_eden_token_metadata*

Description

magic_eden_token_metadata

Usage

```
magic_eden_token_metadata(mint_address, timeout_seconds = 60)
```

Arguments

mint_address	the mint address of the token you wish to query
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a list containing the token metadata for the specified mint address.

Examples

```
mint_address <- "Hd6sxFEEQQA5aURaWaDesi23AkM19bBkKave1hyWvnfS"
magic_eden_token_metadata(mint_address, timeout_seconds = 4.5)
```

magic_eden_transactions *magic_eden_transactions*

Description

magic_eden_transactions

Usage

```
magic_eden_transactions(
  wallet,
  offset = NULL,
  limit = NULL,
  timeout_seconds = 60
)
```

Arguments

wallet	the address of the wallet you are trying to query
offset	optionally provide a numeric value to specify number of transactions to skip.
limit	optionally provide a numeric limit to specify maximum number of results.
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a dataframe containing all transactions for the specified wallet.

Examples

```
wallet <- "72tXz6jhGVPFE8ZfAQocJPJU3HgxsdrRqKZoUdWUhs7o"
magic_eden_transactions(wallet, timeout_seconds = 4.5)
```

---

nifty_gateway_creators

nifty_gateway_creators

Description

nifty_gateway_creators

Usage

```
nifty_gateway_creators(
  username,
  limit = NULL,
  offset = NULL,
  timeout_seconds = 60
)
```

Arguments

username	the username you wish to query
limit	optionally provide the maximum number of results to return. This is a numeric parameter.
offset	optionally specify how many results to skip. This is a numeric parameter.
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a dataframe containing ownership information for all nifties created by the given creator and currently stored on Nifty Gateway

Examples

```
creators <- nifty_gateway_creators('beeple')
```

`nifty_gateway_user_nifties` *nifty_gateway_user_nifties*

Description

nifty_gateway_user_nifties

Usage

```
nifty_gateway_user_nifties(
  username,
  limit = NULL,
  offset = NULL,
  contract_address = NULL,
  timeout_seconds = 60
)
```

Arguments

`username`	the username you wish to query
`limit`	optionally provide the maximum number of results to return. This is a numeric parameter.
`offset`	optionally specify how many results to skip. This is a numeric parameter.
`contract_address`	optionally filter results by contract address.
`timeout_seconds`	seconds until the query times out. Default is 60.

Value

returns a dataframe containing information about the nifties owned by the specified user

Examples

```
nifties <- nifty_gateway_user_nifties('tommy')
```

okcoin_api_call *okcoin_api_call*

Description

okcoin_api_call

Usage

```
okcoin_api_call(
  url,
  key,
  signature,
  formatted_time,
  passphrase,
  timeout_seconds = 60
)
```

Arguments

url	the full URL for the API call
key	your API key for Okcoin
signature	your hashed and encoded signature for Okcoin API calls
formatted_time	a string containing the currrent timestamp in ISO 8601 format
passphrase	the passphrase which you created when generating your Okcoin API key
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a dataframe containing the results of your API call

Examples

```
## Not run:
url <- "..."
key <- "..."
path <- "..."
secret <- "..."
formatted_time <- okcoin_time()
method <- "GET"
signature <- okcoin_signature(path, secret, formatted_time, method)
passphrase <- "..."
data <- okcoin_api_call()
## End(Not run)
```

okcoin_orders *okcoin_orders*

Description

okcoin_orders

Usage

```
okcoin_orders(
  secret,
  key,
  passphrase,
  instrument_id,
  state,
  timeout_seconds = 60
)
```

Arguments

secret	your secret key for Okcoin
key	your API key for Okcoin
passphrase	the passphrase which you created when generating your Okcoin API key
instrument_id	the trading pair symbol
state	Order Status: -1: Canceled, 0: Open, 1: Partially Filled, 2: Fully Filled, 3: Submitting, 4: Canceling, 6: Incomplete (open + partially filled), 7: Complete (canceled + fully filled)
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a dataframe containing your orders from the most recent 3 months

Examples

```
## Not run:
secret <- "..."
key <- "..."
passphrase <- "..."
instrument_id <- "BTC-USDT"
state <- '2'
orders <- okcoin_orders(secret, key, passphrase, instrument_id, state)
## End(Not run)
```

okcoin_signature *okcoin_signature*

Description

okcoin_signature

Usage

```
okcoin_signature(path, secret, formatted_time, method)
```

Arguments

path	the API endpoint
secret	your Okcoin secret key
formatted_time	a string containing the currrent timestamp in ISO 8601 format
method	'POST' or 'GET'

Value

returns a base64 encoded SHA256 signature for signing Okcoin API calls

Examples

```
## Not run:
path <- "..."
secret <- "..."
formatted_time <- okcoin_time()
method <- "GET"
signature <- okcoin_signature(path, secret, formatted_time, method)
## End(Not run)
```

---

okcoin_spot_account_info
okcoin_spot_account_info

Description

okcoin_spot_account_info

Usage

```
okcoin_spot_account_info(secret, key, passphrase, timeout_seconds = 60)
```

Arguments

secret	your secret key for Okcoin
key	your API key for Okcoin
passphrase	the passphrase which you created when generating your Okcoin API key
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a dataframe containing your spot account balances

Examples

```
## Not run:
secret <- "..."
key <- "..."
passphrase <- "..."
balances <- okcoin_spot_account_info(secret, key, passphrase)
## End(Not run)
```

---

okcoin_time *okcoin_time*

Description

okcoin_time

Usage

```
okcoin_time()
```

Value

returns a string with the current timestamp in ISO 8601 format

Examples

```
okcoin_time()
```

okcoin_trading_pairs *okcoin_trading_pairs*

Description

okcoin_trading_pairs

Usage

```
okcoin_trading_pairs(timeout_seconds = 60)
```

Arguments

timeout_seconds	seconds until the query times out. Default is 60.

Value

returns a dataframe containing information about all trading pairs on Okcoin

Examples

```
okcoin_trading_pairs(4.5)
```

paxos_bearer_token *paxos_bearer_token*

Description

paxos_bearer_token

Usage

```
paxos_bearer_token(client_id, client_secret, scope, timeout_seconds = 60)
```

Arguments

client_id	the client id you generated when you created your API key
client_secret	the client secret you generated when you created your API key
scope	the scope needed for your specific API call
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns your Paxos bearer token

Examples

```
## Not run:
client_id <- "..."
client_secret <- "..."
scope <- 'funding:read_address'
token <- paxos_bearer_token(client_id, client_secret, scope)
## End(Not run)
```

`paxos_list_profiles` *paxos_list_profiles*

Description

paxos_list_profiles

Usage

```
paxos_list_profiles(client_id, client_secret, timeout_seconds = 60)
```

Arguments

`client_id`	the client id you generated when you created your API key
`client_secret`	the client secret you generated when you created your API key
`timeout_seconds`	seconds until the query times out. Default is 60.

Value

returns a dataframe containing all user profiles

Examples

```
## Not run:
client_id <- "..."
client_secret <- "..."
profiles <- paxos_list_profiles(client_id, client_secret)
## End(Not run)
```

solana_api_call *solana_api_call*

Description

solana_api_call

Usage

```
solana_api_call(url, request_body, timeout_seconds = 60)
```

Arguments

url	the RPC url for your API call
request_body	the request body for your API call
timeout_seconds	seconds until the query times out. Default is 60.

Value

returns data from your Solana API call

Examples

```
url <- "https://api.devnet.solana.com"
request_body <-
  solana_assemble_request_body('"2.0"', 'null', '"getBlockHeight"', NULL)
data <- solana_api_call(url, request_body)
```

solana_assemble_key_pair *solana_assemble_key_pair*

Description

solana_assemble_key_pair

Usage

```
solana_assemble_key_pair(key, pair)
```

Arguments

key	the key for your key pair
pair	the pair for your key pair

Value

Returns your key pair if it exists or a blank string if it doesn't exist

Examples

```
limit <- NULL
limit <- solana_assemble_key_pair('limit', limit)
```

`solana_assemble_list` *solana_assemble_list*

Description

solana_assemble_list

Usage

```
solana_assemble_list(character_vector)
```

Arguments

`character_vector`	the character vector used to create the config object

Value

Returns your config object

Examples

```
limit <- solana_assemble_key_pair('limit', NULL)
character_vector <- c(limit)
config_object <- solana_assemble_list(character_vector)
```

`solana_assemble_request_body` *solana_assemble_request_body*

Description

solana_assemble_request_body

Usage

```
solana_assemble_request_body(jsonrpc, id, method, params)
```

Arguments

jsonrpc	the jsonrpc for your request body
id	the id for your request body
method	the method for your request body
params	the parameters for your request body

Value

Returns the request body for your solana API call

Examples

```
limit <- solana_assemble_key_pair('limit', NULL)
character_vector <- c(limit)
config_object <- solana_assemble_list(character_vector)
address <- "Vote111111111111111111111111111111111111111"
params <- paste('["', address, '", {', config_object, '}]', sep = '')
solana_assemble_request_body('"2.0"', 'null', '"getSignaturesForAddress"', params)
```

---

solana_get_account_info

solana_get_account_info

Description

solana_get_account_info

Usage

```
solana_get_account_info(url, pubkey, timeout_seconds = 60)
```

Arguments

url	the RPC url for your API call
pubkey	the pubkey for which you're retrieving account info
timeout_seconds	seconds until the query times out. Default is 60.

Value

Returns all information associated with the account of provided Pubkey

Examples

```
url <- "https://api.devnet.solana.com"
pubkey <- "vines1vzrYbzLMRdu58ou5XTby4qAqVRLmqo36NKPTg"
data <- solana_get_account_info(url, pubkey)
```

solana_get_block *solana_get_block*

Description

solana_get_block

Usage

```
solana_get_block(url, slot, timeout_seconds = 60)
```

Arguments

url	the RPC url for your API call
slot	slot number, as u64 integer
timeout_seconds	seconds until the query times out. Default is 60.

Value

Returns identity and transaction information about a confirmed block in the ledger.

Examples

```
url <- "https://api.devnet.solana.com"
slot <- solana_get_slot(url)
data <- solana_get_block(url, slot)
```

solana_get_block_height *solana_get_block_height*

Description

solana_get_block_height

Usage

```
solana_get_block_height(url, timeout_seconds = 60)
```

Arguments

url	the RPC url for your API call
timeout_seconds	seconds until the query times out. Default is 60.

Value

Returns the current block height of the node

Examples

```
url <- "https://api.devnet.solana.com"
data <- solana_get_block_height(url)
```

`solana_get_genesis_hash` *solana_get_genesis_hash*

Description

solana_get_genesis_hash

Usage

```
solana_get_genesis_hash(url, timeout_seconds = 60)
```

Arguments

`url`	the RPC url for your API call
`timeout_seconds`	seconds until the query times out. Default is 60.

Value

Returns the genesis hash

Examples

```
url <- "https://api.devnet.solana.com"
data <- solana_get_genesis_hash(url)
```

solana_get_health *solana_get_health*

Description

solana_get_health

Usage

```
solana_get_health(url, timeout_seconds = 60)
```

Arguments

url	the RPC url for your API call
timeout_seconds	seconds until the query times out. Default is 60.

Value

Returns the current health of the node.

Examples

```
url <- "https://api.devnet.solana.com"
data <- solana_get_health(url)
```

solana_get_identity *solana_get_identity*

Description

solana_get_identity

Usage

```
solana_get_identity(url, timeout_seconds = 60)
```

Arguments

url	the RPC url for your API call
timeout_seconds	seconds until the query times out. Default is 60.

Value

Returns the identity pubkey for the current node

Examples

```
url <- "https://api.devnet.solana.com"
data <- solana_get_identity(url)
```

`solana_get_inflation_rate` *solana_get_inflation_rate*

Description

solana_get_inflation_rate

Usage

```
solana_get_inflation_rate(url, timeout_seconds = 60)
```

Arguments

`url`	the RPC url for your API call
`timeout_seconds`	seconds until the query times out. Default is 60.

Value

Returns the specific inflation values for the current epoch

Examples

```
url <- "https://api.devnet.solana.com"
data <- solana_get_inflation_rate(url)
```

`solana_get_recent_prioritization_fees` *solana_get_recent_prioritization_fees*

Description

solana_get_recent_prioritization_fees

Usage

```
solana_get_recent_prioritization_fees(url, timeout_seconds = 60)
```

Arguments

url	the RPC url for your API call
timeout_seconds	seconds until the query times out. Default is 60.

Value

Returns a list of prioritization fees from recent blocks.

Examples

```
url <- "https://api.devnet.solana.com"
data <- solana_get_recent_prioritization_fees(url)
```

---

solana_get_signature_for_address

solana_get_signature_for_address

Description

solana_get_signature_for_address

Usage

```
solana_get_signature_for_address(
  url,
  address,
  limit = NULL,
  timeout_seconds = 60
)
```

Arguments

url	the RPC url for your API call
address	the address for which you're retrieving signatures
limit	maximum transaction signatures to return (between 1 and 1,000). Default is 1,000.
timeout_seconds	seconds until the query times out. Default is 60.

Value

Returns signatures for confirmed transactions that include the given address in their accountKeys list. Returns signatures backwards in time from the provided signature or most recent confirmed block

Examples

```
url <- "https://api.devnet.solana.com"
address <- "Vote111111111111111111111111111111111111111"
data <- solana_get_signature_for_address(url,address)
```

`solana_get_slot` *solana_get_slot*

Description

solana_get_slot

Usage

```
solana_get_slot(url, timeout_seconds = 60)
```

Arguments

`url`	the RPC url for your API call
`timeout_seconds`	seconds until the query times out. Default is 60.

Value

Returns the slot that has reached the given or default commitment level.

Examples

```
url <- "https://api.devnet.solana.com"
data <- solana_get_slot(url)
```

`solana_get_supply` *solana_get_supply*

Description

solana_get_supply

Usage

```
solana_get_supply(url, timeout_seconds = 60)
```

Arguments

`url`	the RPC url for your API call
`timeout_seconds`	seconds until the query times out. Default is 60.

Value

Returns information about the current supply.

Examples

```
url <- "https://api.devnet.solana.com"
data <- solana_get_supply(url)
```

`solana_get_version` *solana_get_version*

Description

solana_get_version

Usage

```
solana_get_version(url, timeout_seconds = 60)
```

Arguments

`url`	the RPC url for your API call
`timeout_seconds`	seconds until the query times out. Default is 60.

Value

Returns the current Solana version running on the node

Examples

```
url <- "https://api.devnet.solana.com"
data <- solana_get_version(url)
```

Index