Package 'cryptowatchR'

October 12, 2022

Type Package

Title An API Wrapper for 'Cryptowatch'

Version 0.2.0

Description
An API wrapper for 'Cryptowatch' to get prices and other information (e.g., volume, trades, order books, bid and ask prices, live quotes, and more) about cryptocurrencies and crypto exchanges. See <https://docs.cryptowat.ch/rest-api> for a detailed documentation.

Imports httr, jsonlite, lubridate

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.1.1

URL https://github.com/lorenzbr/cryptowatchR

BugReports https://github.com/lorenzbr/cryptowatchR/issues

NeedsCompilation no

Author Lorenz Brachtendorf [aut, cre]
(<https://orcid.org/0000-0001-7066-0838>)

Maintainer Lorenz Brachtendorf <lorenz.brachtendorf@gmx.de>

Repository CRAN

Date/Publication 2021-06-09 21:20:02 UTC

R topics documented:

`get_assets` *Get asset details*

Description

Get asset information on cryptocurrencies.

Usage

```
get_assets(asset = NULL, api_key = NULL, allowance = FALSE)
```

Arguments

`asset`	A string containing an asset symbol, e.g. *btc* (optional argument). Run `get_assets()` to get all available assets.
`api_key`	A string containing the API key. See https://docs.cryptowat.ch/rest-api/rate-limit to learn how to create an account and how to generate an API key.
`allowance`	A logical (default is `FALSE`). If `TRUE` the function returns a list which includes allowance information, i.e. cost of the request, remaining credits and your account name.

Value

A list or data.frame containing data on assets.

References

See https://docs.cryptowat.ch/rest-api for further information

See Also

`get_markets`, `get_exchanges`, `get_pairs`

Examples

```
## Not run:
# Get all assets available on 'Cryptowatch'
df.assets <- get_assets()
# Bitcoin asset details
asset.btc <- get_assets("btc")

## End(Not run)
```

get_current_price *Get current price of cryptocurrencies*

Description

Get current price of cryptocurrencies using the REST API of cryptowat.ch. The route is *price* or *prices* and returns the current price of a given pair or current prices of all pairs. See https://docs.cryptowat.ch/rest-api/markets/price for further information.

Usage

```
get_current_price(pair, exchange = "kraken", api_key = NULL, allowance = FALSE)
```

Arguments

pair	A string containing a pair symbol, e.g. *btcusd* (required argument). Run `get_pairs()` to find other available pairs.
exchange	A string containing the exchange. Default is *kraken*. Run `get_exchanges()` to find other available exchanges.
api_key	A string containing the API key. See https://docs.cryptowat.ch/rest-api/rate-limit to learn how to create an account and how to generate an API key.
allowance	A logical (default is `FALSE`). If `TRUE` the function returns a list which includes allowance information, i.e. cost of the request, remaining credits and your account name.

Value

Current price of a given pair of currencies. If allowance is `TRUE`, `get_current_price()` returns a list.

References

See https://docs.cryptowat.ch/rest-api for further information.

See Also

get_markets, get_ohlc, get_exchanges, get_pairs

Examples

```
## Not run:
# Daily prices of Bitcoin in USD
current.price <- get_current_price("btcusd")
current.prices <- get_current_price()

## End(Not run)
```

get_exchanges *Get exchange details*

Description

Get information on exchanges.

Usage

```
get_exchanges(exchange = NULL, api_key = NULL, allowance = FALSE)
```

Arguments

exchange	A string containing an exchange symbol, e.g. *kraken* (optional argument). Run `get_exchanges()` to get a list of exchanges.
api_key	A string containing the API key. See https://docs.cryptowat.ch/rest-api/rate-limit to learn how to create an account and how to generate an API key.
allowance	A logical (default is FALSE). If TRUE the function returns a list which includes allowance information, i.e. cost of the request, remaining credits and your account name.

Value

A list or data.frame containing data on exchanges.

References

See https://docs.cryptowat.ch/rest-api for further information

See Also

get_markets, get_assets, get_pairs

Examples

```
## Not run:
# Get all available exchanges
df.exchanges <- get_exchanges()
# Get information on the exchange Kraken
exchange.kraken <- get_exchanges("kraken")

## End(Not run)
```

get_markets *Get prices of cryptocurrencies*

Description

Get prices of cryptocurrencies using the REST API of cryptowat.ch.

Usage

```
get_markets(route, pair = NULL, exchange = NULL, params = NULL,
            api_key = NULL, allowance = FALSE)
```

Arguments

route	A character string containing a market endpoint. Possible values: *price, prices, trades, summary, summaries, orderbook, orderbook/liquidity, orderbooks/calculator, ohlc* (required argument). See *https://docs.cryptowat.ch/rest-api/markets* for further information.
pair	A character string containing a pair symbol, e.g. *btcusd* (optional argument). Run `get_pairs()` to find other available pairs.
exchange	A character string containing the exchange (optional argument). Run `get_exchanges()` to find other available exchanges.
params	A list containing query parameters. E.g., for the route *ohlc*, this is `before`, `after` and `periods` (optional). See *https://docs.cryptowat.ch/rest-api/markets* for further information.
api_key	A character string containing the API key. See https://docs.cryptowat.ch/rest-api/rate-limit to learn how to create an account and how to generate an API key.
allowance	A logical (default is `FALSE`). If `TRUE` the function returns a list which includes allowance information, i.e. cost of the request, remaining credits and your account name.

Value

A list containing markets data.

References

See https://docs.cryptowat.ch/rest-api for further information

See Also

markets, get_assets, get_exchanges, get_pairs

Examples

```
## Not run:
# Prices of Bitcoin in USD for all periods
btcusd.ohlc.all <- get_markets(route = "ohlc", pair = "btcusd", exchange = "kraken")
# Hourly prices of Bitcoin in USD for a specific time period
btcusd.ohlc.hourly <- get_markets(route = "ohlc", pair = "btcusd", exchange = "kraken",
                          list(periods = 3600, before = 1609851600, after = 1609506000))

## End(Not run)
```

`get_ohlc` *Get prices of cryptocurrencies*

Description

Get data.frame with prices of cryptocurrencies using the REST API of cryptowat.ch. The route is *ohlc* and returns OHLC candlestick prices. The default is daily prices but can be changed with `periods`. See https://docs.cryptowat.ch/rest-api/markets/ohlc for further information.

Usage

```
get_ohlc(pair, before = NULL, after = NULL, periods = NULL, exchange = "kraken",
         datetime = TRUE, api_key = NULL, allowance = FALSE)
```

Arguments

`pair`	A character string containing a pair symbol, e.g. *btcusd* (required argument). Run `get_pairs()` to find other available pairs.
`before`	An integer if *datetime* is `FALSE` and a character string if it is `TRUE`: Only returns candles opening before this time (optional).
`after`	An integer if *datetime* is `FALSE` and a character string if it is `TRUE`: Only returns candles opening after this time (optional).
`periods`	A integer or integer vector. Only return these periods. Periods are measured in seconds (optional). Examples: 60, 180, 108000.
`exchange`	A character string containing the exchange. Default is *kraken*. Run `get_exchanges()` to find other available exchanges.
`datetime`	A logical. `TRUE` indicates that datetime type is used. `FALSE` indicates *unix timestamp*. Default is `TRUE`.
`api_key`	A character string containing the API key. See https://docs.cryptowat.ch/rest-api/rate-limit to learn how to create an account and how to generate an API key.
`allowance`	A logical (default is `FALSE`). If `TRUE` the function returns a list which includes allowance information, i.e. cost of the request, remaining credits and your account name.

Value

A data.frame containing OHLC candlestick prices of a given pair of currencies. If allowance is TRUE, get_ohlc() returns a list.

References

See https://docs.cryptowat.ch/rest-api for further information.

See Also

get_markets, get_assets, get_exchanges, get_pairs

Examples

```
## Not run:
# Daily prices of Bitcoin in USD
df.ohlc <- get_ohlc("btcusd")
# Hourly prices of Bitcoin in USD for a specific time period
df.ohlc.hourly <- get_ohlc("btcusd", periods = 3600, before = 1609851600,
                                          after = 1609506000, datetime = FALSE)
# Daily prices of Bitcoin in Euro for a specific time period
df.ohlc.daily <- get_ohlc("btceur", periods = 86400, before = "2021-05-12",
                                          after = "2021-01-01", datetime = TRUE)

## End(Not run)
```

---

get_orderbook *Get order book of cryptocurrencies*

Description

Get the order book of cryptocurrencies using the REST API of cryptowat.ch. The route is *orderbook* and returns two arrays, bids and asks. See https://docs.cryptowat.ch/rest-api/markets/order-book for further information.

Usage

```
get_orderbook(pair, depth = NULL, span = NULL, limit = NULL,
              exchange = "kraken", api_key = NULL, allowance = FALSE)
```

Arguments

pair	A string containing a pair symbol, e.g. *btcusd* (required argument). Run get_pairs() to find other available pairs.
depth	A number: Only return orders cumulating up to this size (optional).
span	A number: Only return orders within this percentage of the midpoint (optional). Example: 0.5 (meaning 0.5 percent).

limit	An integer limiting the number of orders on each side of the book (optional).
exchange	A character string containing the exchange. Default is *kraken*. Run `get_exchanges()` to find other available exchanges.
api_key	A character string containing the API key. See https://docs.cryptowat.ch/rest-api/rate-limit to learn how to create an account and how to generate an API key.
allowance	A logical (default is FALSE). If TRUE the function returns a list which includes allowance information, i.e. cost of the request, remaining credits and your account name.

Value

A list containing an order book of a given pair of currencies. It contain two data.frames for bid and ask prices with columns *Price* and *Amount*. The function also returns the sequence number *seqNum*. If allowance is TRUE, `get_orderbook()` returns a list which additionally includes allowance information.

References

See https://docs.cryptowat.ch/rest-api for further information.

See Also

get_markets, get_orderbook_liquidity, get_exchanges, get_pairs

Examples

```
## Not run:
# Entire order book of Bitcoin in USD
orderbook <- get_orderbook("btcusd")
# Order book of Bitcoin in USD: only the best bid and best ask, i.e. the spread
orderbook.limit <- get_orderbook("btcusd", limit = 1)
# Order book of Bitcoin in USD for orders within 0.5% of the top of the book
orderbook.span <- get_orderbook("btcusd", span = 0.5)
# Order book of Bitcoin in Euro for orders adding up to 100 BTC on each side
orderbook.depth <- get_orderbook("btceur", depth = 100)

## End(Not run)
```

---

get_orderbook_calculator

Get live quotes from order book

Description

Get a live quote from the order book for a given buy and sell amount. The route is *orderbook/calculator* and returns two data.frames containing buy and sell projections. See https://docs.cryptowat.ch/rest-api/markets/order-book for further information.

Usage

```
get_orderbook_calculator(pair, amount = NULL, exchange = "kraken",
                         api_key = NULL, allowance = FALSE)
```

Arguments

pair	A string containing a pair symbol, e.g. *btcusd* (required argument). Run `get_pairs()` to find other available pairs.
amount	A number: Amount to buy or sell (required).
exchange	A character string containing the exchange. Default is *kraken*. Run `get_exchanges()` to find other available exchanges.
api_key	A character string containing the API key. See https://docs.cryptowat.ch/rest-api/rate-limit to learn how to create an account and how to generate an API key.
allowance	A logical (default is `FALSE`). If `TRUE` the function returns a list which includes allowance information, i.e. cost of the request, remaining credits and your account name.

Value

A list containing projections for the buy and sell side for a given amount. *Reach* indicates the farthest your reach would fill. *Avg* indicates the volume-weighted average. If allowance is `TRUE`, `get_orderbook_calculator()` returns a list which additionally includes allowance information.

References

See https://docs.cryptowat.ch/rest-api for further information.

See Also

`get_markets`, `get_orderbook`, `get_orderbook_liquidity`, `get_exchanges`, `get_pairs`

Examples

```
## Not run:
# Live quote for 50 Bitcoins
calculator <- get_orderbook_calculator("btcusd", amount = 50)

## End(Not run)
```

`get_orderbook_liquidity` *Get liquidity sums in the order book of cryptocurrencies*

Description

Get liquidity sums in the order book of cryptocurrencies using the REST API of cryptowat.ch. The route is *orderbook/liquidity*. See https://docs.cryptowat.ch/rest-api/markets/order-book for further information.

Usage

```
get_orderbook_liquidity(pair, exchange = "kraken", api_key = NULL, allowance = FALSE)
```

Arguments

`pair`	A character string containing a pair symbol, e.g. *btcusd* (required argument). Run `get_pairs()` to find other available pairs.
`exchange`	A character string containing the exchange. Default is *kraken*. Run `get_exchanges()` to find other available exchanges.
`api_key`	A character string containing the API key. See https://docs.cryptowat.ch/rest-api/rate-limit to learn how to create an account and how to generate an API key.
`allowance`	A logical (default is `FALSE`). If `TRUE` the function returns a list which includes allowance information, i.e. cost of the request, remaining credits and your account name.

Value

A list containing liquidity sums at several basis point levels in the order book. If allowance is `TRUE`, `get_orderbook()` returns a list which additionally includes allowance information.

References

See https://docs.cryptowat.ch/rest-api for further information.

See Also

`get_markets`, `get_orderbook`, `get_exchanges`, `get_pairs`

Examples

```
## Not run:
# Get liquidity sums in the order book of Bitcoin in USD
liquidity <- get_orderbook_liquidity("btcusd")

## End(Not run)
```

get_pairs *Get pair of currencies*

Description

Get details on pairs of (crypto)currencies.

Usage

```
get_pairs(pair = NULL, api_key = NULL, allowance = FALSE)
```

Arguments

pair	A string containing a pair symbol, e.g. *btcusd* (optional argument). Run `get_pairs()` to get all available pairs of currencies.
api_key	A string containing the API key. See https://docs.cryptowat.ch/rest-api/rate-limit to learn how to create an account and how to generate an API key.
allowance	A logical (default is FALSE). If TRUE the function returns a list which includes allowance information, i.e. cost of the request, remaining credits and your account name.

Value

A list or data.frame containing data on pairs.

References

See https://docs.cryptowat.ch/rest-api for further information

See Also

get_markets, get_assets, get_exchanges

Examples

```
## Not run:
# Get all available pairs of currencies
df.pairs <- get_pairs()
# Get details on the pair Bitcoin-USD
pair.btcusd <- get_pairs("btcusd")

## End(Not run)
```

get_summary *Get 24-hour summary of cryptocurrencies*

Description

Get 24-hour summary of cryptocurrencies using the REST API of cryptowat.ch. The route is *summary* or *summaries* and returns the current 24-hour summary of a given pair or a summary of all pairs. See https://docs.cryptowat.ch/rest-api/markets/summary for further information.

Usage

```
get_summary(pair, keyBy = NULL, exchange = "kraken", api_key = NULL, allowance = FALSE)
```

Arguments

pair	A character string containing a pair symbol, e.g. *btcusd* (required argument). Run get_pairs() to find other available pairs.
keyBy	A single character string. Possible values are are either *id* or *symbols*.
exchange	A character string containing the exchange. Default is *kraken*. Run get_exchanges() to find other available exchanges.
api_key	A character string containing the API key. See https://docs.cryptowat.ch/rest-api/rate-limit to learn how to create an account and how to generate an API key.
allowance	A logical (default is FALSE). If TRUE the function returns a list which includes allowance information, i.e. cost of the request, remaining credits and your account name.

Value

Current 24-hour summary of a given pair of currencies. If allowance is TRUE, get_summary() returns a list with additional information on allowances.

References

See https://docs.cryptowat.ch/rest-api for further information.

See Also

get_markets, get_current_price, get_exchanges, get_pairs

Examples

```
## Not run:
# 24h-hour summary of cryptocurrency pairs
summary <- get_summary("btcusd")
summaries <- get_summary()
summaries2 <- get_summary(keyBy = "id")
```

```
summaries3 <- get_summary(keyBy = "symbols")

## End(Not run)
```

`get_trades` *Get trades of cryptocurrencies*

Description

Get data.frame with trades of cryptocurrencies using the REST API of cryptowat.ch. The route is *trades* and returns trades for a specified market. See https://docs.cryptowat.ch/rest-api/markets/trades for further information.

Usage

```
get_trades(pair, since = NULL, limit = NULL, exchange = "kraken",
           datetime = TRUE, api_key = NULL, allowance = FALSE)
```

Arguments

`pair`	A character string containing a pair symbol, e.g. *btcusd* (required argument). Run `get_pairs()` to find other available pairs.
`since`	An integer if *datetime* is `FALSE` and a character string if it is `TRUE`: Limit the response to trades after this date (optional). This can only be used to filter recent trades. Historical trades cannot be retrieved.
`limit`	An integer: Limit the number of trades (optional). Max: 1000.
`exchange`	A character string containing the exchange. Default is *kraken*. Run `get_exchanges()` to find other available exchanges.
`datetime`	A logical. `TRUE` indicates that datetime type is used. `FALSE` indicates *unix timestamp*. Default is `TRUE`.
`api_key`	A character string containing the API key. See https://docs.cryptowat.ch/rest-api/rate-limit to learn how to create an account and how to generate an API key.
`allowance`	A logical (default is `FALSE`). If `TRUE` the function returns a list which includes allowance information, i.e. cost of the request, remaining credits and your account name.

Value

A data.frame containing trades of a given pair of currencies. If allowance is `TRUE`, `get_trades()` returns a list.

References

See https://docs.cryptowat.ch/rest-api for further information.

See Also

get_markets, get_assets, get_exchanges, get_pairs

Examples

```
## Not run:
# Most recent trades (default is 50)
trades <- get_trades(pair)
# 200 trades (maximum is 1000) since 1589571417 (unix timestamp)
trades.unix <- get_trades(pair, since = 1589571417, limit = 200, datetime = FALSE)
# 1000 trades and datetime is TRUE
trades.datetime <- get_trades(pair, since = "2021-06-01", limit = 1000)

## End(Not run)
```

---

markets *Get prices of cryptocurrencies*

Description

Get data.frame with prices of cryptocurrencies using the REST API of cryptowat.ch.

Usage

```
markets(pair, params = NULL, exchange = "kraken", route = "ohlc", datetime = TRUE)
```

Arguments

pair	A string containing a pair symbol, e.g. *btcusd* (required argument).
params	A list containing before, after and periods (optional). See https://docs.cryptowat.ch/rest-api/markets/ohlc for further information.
exchange	A character string containing the exchange. Default is *kraken*.
route	A character string containing the route. Default is *ohlc*.
datetime	A logical. TRUE indicates that datetime type is used. FALSE indicates *unix timestamp*. Default is TRUE.

Value

A data.frame containing OHLC candlestick prices of a given pair of currencies.

References

See https://docs.cryptowat.ch/rest-api for further information.

See Also

get_markets, get_assets, get_exchanges, get_pairs

Examples

```
## Not run:
df.ohlc.hourly <- markets("btcusd")
df.ohlc.hourly2 <- markets("btcusd", list(periods = 3600, before = 1609851600,
                                          after = 1609506000), datetime = FALSE)
df.ohlc.daily <- markets("btceur", list(periods = 86400, before = "2021-05-12",
                                          after = "2021-01-01"), datetime = TRUE)

## End(Not run)
```

Index