Package ‘crypto2’

October 8, 2026

Title Download Crypto Currency Data from 'CoinMarketCap' and 'CoinGecko'

Version 3.0.0

Date 2026-10-07

Maintainer Sebastian Stoeckl <sebastian.stoeckl@uni.li>

URL https://github.com/sstoeckl/crypto2, https://www.sebastianstoeckl.com/crypto2/

Description

Retrieves crypto currency information and historical prices as well as information on the exchanges they are listed on. Historical data contains daily open, high, low and close values for all crypto currencies. The package draws on two complementary sources: 'CoinMarketCap' <https://coinmarketcap.com> (primary, via the 'crypto_*' functions) and 'CoinGecko' <https://www.coingecko.com> (secondary, via the 'cg_*' functions). Both sources are queried without an 'API' key; the two function families return tibbles with identical column conventions so downstream pipelines work on either source.

License MIT + file LICENSE

Depends R (>= 4.1.0)

Imports dplyr, tibble, tidyr, purrr, progress, stats, lubridate, jsonlite, cli, plyr, base64enc, janitor, httr

Encoding UTF-8

RoxygenNote 7.3.3

Suggests spelling, testthat (>= 3.0.0), digest, knitr, rmarkdown, arrow, withr

VignetteBuilder knitr

Language en-US

Config/testthat/edition 3

NeedsCompilation no

Author Sebastian Stoeckl [aut, cre] (ORCID: <https://orcid.org/0000-0002-4196-6093>, Package commissioner and maintainer.),
Jesse Vent [ctb] (Creator of the crypto package that provided the idea and basis for this package.)

Repository CRAN

Date/Publication 2026-10-08 17:20:02 UTC

Contents

cg_history . . . 2
cg_history_by_id . . . 4
cg_id_mapping . . . 7
cg_info . . . 8
cg_list . . . 9
cg_listings . . . 10
crypto_crosswalk . . . 13
crypto_global_quotes . . . 14
crypto_history . . . 16
crypto_info . . . 18
crypto_list . . . 20
crypto_listings . . . 21
exchange_info . . . 24
exchange_list . . . 26
fiat_list . . . 27

Index **28**

`cg_history` *Get historic crypto currency market data from CoinGecko*

Description

Companion to `crypto_history()` but for CoinGecko. Returns daily OHLC, volume, and market-cap timeseries in a tibble whose column names match the crypto2 CMC output.

Usage

```
cg_history(
  coin_list = NULL,
  convert = "USD",
  limit = NULL,
  start_date = NULL,
  end_date = NULL,
  interval = NULL,
  requestLimit = 400,
  sleep = 0,
  wait = 60,
  finalWait = FALSE,
  single_id = TRUE,
  date_convention = c("end_of_day", "raw")
)
```

Arguments

- `coin_list` — string if NULL retrieve all currently existing coins (`cg_list()`), or provide list of crypto currencies in the `cg_list()` / `cg_listings()` format.
- `convert` — (default: `"USD"`). Any CoinGecko quote currency, e.g. `"BTC"`, `"ETH"` or `"EUR"`.
- `limit` — integer Return the top n records, default is all tokens.
- `start_date, end_date` — date Filter the returned timeseries to this date window after fetching.
- `interval` — string Always coerced to `"daily"` – CoinGecko website endpoints return daily granularity for full-history pulls. Hourly is not available without an API key.
- `requestLimit` — Kept for parity with `crypto_history()` – ignored (CoinGecko returns full history per coin in one call).
- `sleep` — integer (default `0`) Seconds to sleep between API requests. The internal client enforces a polite floor to stay within CoinGecko's per-minute budget.
- `wait` — waiting time before retry in case of fail (default `60`).
- `finalWait` — Sleep 60s after the last call (mirrors `crypto_history()`).
- `single_id` — Kept for parity with `crypto_history()` – ignored; CoinGecko endpoints are always single-coin per call.
- `date_convention` — Either `"end_of_day"` (the default) or `"raw"`. CoinGecko's native daily series timestamps each point at 00:00:00 UTC of date X, which is the same physical instant as 23:59:59 UTC of date X-1 – i.e. CG labels it as the start-of-day rather than the close-of-day. CMC (and `crypto_history()`) label that instant as date X-1 (the day that just ended), which is also the standard asset-pricing convention used by CRSP/Compustat and major academic datasets. With `"end_of_day"` (default) `cg_history()` shifts CG's midnight ticks by -1 day so `close[X] / close[X-1] - 1` is the return earned during date X, matching CMC. Pass `"raw"` to keep CG's native start-of-day labelling.

Details

No API key is required. When the requested coin's numeric id is missing in `coin_list`, `cg_id_mapping()` is consulted to fill the `id` column.

All data come from CoinGecko's website chart endpoints; no API and no key are used. Coverage:

- **Close, volume and market cap for the full lifetime of each coin**, in any quote currency. In USD they come from CoinGecko's daily CSV export (one request per coin), otherwise from the website's chart data (two requests per coin).
- **OHLC** (`open` / `high` / `low`) is built from 4-hour candles (one extra request per coin) and covers the **most recent 30 days**; older rows have `NA` there. Longer windows are only offered as 4-day candles, which are not daily bars. For a one-shot complete OHLC backfill see `vignette("coingecko-pro-backfill")`.

Only completed days are returned. If no close/volume/market-cap series can be retrieved for any requested coin, the function stops with an error (the source has most likely changed); if it fails for some coins only, a warning names them.

Fallback (opt-in). With `options(crypto2.cg_api_fallback = TRUE)`, coins the website cannot serve are fetched from CoinGecko's public API instead, still without a key. Its daily history covers only the **last 365 days**, so this is off by default: a call that falls back warns and lists the affected coins in `attr(result, "cg_api_fallback")`.

Value

Crypto currency historic OHLC market data in a tibble:

`id`	CoinGecko internal numeric id (NA if unknown).
`slug`, `name`, `symbol`	Coin identifiers.
`timestamp`	POSIXct (UTC), midnight of the trading day.
`ref_cur_id`	Quote currency code (e.g. `"usd"`).
`ref_cur_name`	Upper-cased quote currency.
`open`, `high`, `low`	Daily OHLC from 4-hour candles (last 30 days).
`close`	Daily close from the price series; back-filled from the OHLC candles where the price series has no value.
`volume`	Daily total volume.
`market_cap`	Daily market cap.
`time_open`, `time_high`, `time_low`, `time_close`	`NA` – CoinGecko does not expose intra-day OHLC timestamps in these endpoints.

Examples

```
## Not run:
# Top 50 by market cap, full available history
top50 <- cg_list()[1:50, ]
hist  <- cg_history(top50)

# Bitcoin only, last year
btc <- cg_history(cg_list()[1, ],
                  start_date = Sys.Date() - 365,
                  end_date   = Sys.Date())

## End(Not run)
```

`cg_history_by_id` *Fetch CoinGecko history by numeric ID (incl. partial survivorship-bias correction)*

Description

Companion to `cg_history()` that addresses coins by their **numeric CoinGecko ID** instead of their slug. Useful in two scenarios:

Usage

```
cg_history_by_id(
  ids = NULL,
  what = c("price", "market_cap", "ohlc"),
  vs_currency = "usd",
  start_date = NULL,
  end_date = NULL,
  coin_list = NULL,
  sleep = 0.6,
  wait = 60,
  max_retries = 3,
  quiet = FALSE,
  finalWait = FALSE,
  date_convention = c("end_of_day", "raw")
)
```

Arguments

`ids` Integer vector of numeric IDs to fetch. Default `NULL` -> uses `cg_list()$id` (active universe). To extend coverage to delisted coins, supply the union of historically-observed IDs from your accumulated snapshots.

`what` Subset of streams to fetch. Any combination of `"price"` (close + volume), `"market_cap"`, and `"ohlc"`. Default all three. Coverage is the same as for `cg_history()`: full history for close, volume and market cap, OHLC for the last 30 days.

`vs_currency` Quote currency, default `"usd"`. The opt-in API fallback of `cg_history()` (`options(crypto2.cg_api_fa = TRUE)`) applies here too, for ids whose slug is known from `coin_list`.

`start_date, end_date` Client-side date filter applied after fetch. `NULL` returns full history.

`coin_list` Optional `cg_list()` output used to join `slug` / `name` / `symbol` onto recovered rows for coins still in the active universe. If `NULL`, calls `cg_list()` automatically. Set to `FALSE` to skip the join (rows then have only `id`).

`sleep, wait, max_retries` Rate-limit knobs. Defaults `0.6` / `60` / `3` match `cg_history()`.

`quiet` If `FALSE`, prints a progress bar.

`finalWait` Sleep 60 s after the last call (mirrors `crypto_history()`).

`date_convention` Either `"end_of_day"` (the default) or `"raw"`. See `cg_history()` for the explanation – this argument applies the same -1 day shift to midnight-UTC ticks so dates align with the CMC / CRSP / Compustat end-of-day convention.

Details

1. Coins that have been **delisted** and whose slug no longer resolves.

2. Cronjob-style accumulation: when you persist `cg_list()` snapshots over time, the **union of all numeric IDs ever observed** is the survivorship-bias-corrected universe. `cg_history_by_id()` lets you refetch each historical ID directly without needing its current slug to still resolve.

Important caveats – please read:

- **The numeric-ID space is sparse, not dense.** Blind iteration over `1:N` does NOT recover the full universe – most numeric IDs in that range have no data. The default `ids = NULL` therefore uses the active universe from `cg_list()`, not a numeric range. To recover delisted coins you must supply the IDs explicitly (e.g., the union of accumulated `cg_list()` snapshots, or the historic mapping from `cg_id_mapping()`).
- **Slug recovery for delisted coins is not generally available on the free tier.** Active coins get their slug/name joined back in from `cg_list()`; rows whose numeric ID is no longer in the active universe come back with `slug = NA` and `name = NA`. Use the `id` column as the join key in downstream code. For a one-shot complete recovery of the full historic universe see `vignette("coingecko-pro-backfill")`.

Value

Tibble with one row per (id, date) using crypto2-compatible column names. Columns:

`id` CoinGecko numeric id (always populated).

`slug`, `name`, `symbol`
: Coin identifiers – `NA` for ids no longer in the active universe (i.e. delisted on the slug side).

`timestamp` POSIXct UTC midnight of the trading day.

`ref_cur_id`, `ref_cur_name`
: Quote currency.

`open`, `high`, `low`, `close`, `volume`, `market_cap`
: Daily values.

Examples

```
## Not run:
# Scan first 200 numeric IDs (will include both active and delisted coins)
h <- cg_history_by_id(ids = 1:200, what = c("price", "market_cap"))

# Full sweep -- survivorship-bias-free price history of the entire
# CoinGecko universe. Slow (10+ hours). Run via cronjob package.
h_all <- cg_history_by_id()

## End(Not run)
```

cg_id_mapping — *Historic CoinGecko id/slug mapping (incl. delisted coins)*

Description

The free CoinGecko API only exposes coins currently tracked, so coins that get delisted after their listing day disappear from `/coins/list`. A separate, periodically updated archive of `(numeric_id, slug, symbol, name, l` rows is hosted at a stable companion location. This function downloads and caches it so callers (e.g. `cg_list()`, `cg_history()`) can transparently fall back to historic identifiers without ceremony.

Usage

```
cg_id_mapping(refresh = FALSE, quiet = FALSE)
```

Arguments

`refresh`	Force re-download even if a cached file exists in `tempdir()`. Default `FALSE`.
`quiet`	Suppress the one-line "historic data current until ..." message. Default `FALSE`.

Details

The mapping is fetched once per session and cached in `tempdir()`. If the network is unavailable, a small bundled sample of reference coins is used as a fallback. When `quiet = FALSE` (default), a single one-line message is emitted on first successful download stating the harvest date.

Value

Tibble with columns `id` (integer numeric CoinGecko id), `slug` (character), `symbol` (character), `name` (character), `harvested_at` (Date) – one row per historic coin. Returns an empty tibble with the correct schema if neither the network mapping nor the bundled sample can be loaded.

Examples

```
## Not run:
mapping <- cg_id_mapping()
delisted <- dplyr::anti_join(mapping,
  cg_list() %>% dplyr::select(slug), by = "slug")

## End(Not run)
```

`cg_info` *Retrieve per-coin metadata from CoinGecko (CMC-compatible columns)*

Description

Companion to `crypto_info()` but for CoinGecko. Pulls descriptive metadata (description, categories, logos, contract addresses across chains, links) for each coin in `coin_list`. Uses the documented public API endpoint `api.coingecko.com/api/v3/coins/{slug}` (no key required; ~30 req/min).

Usage

```
cg_info(
  coin_list = NULL,
  limit = NULL,
  requestLimit = 1,
  sleep = 0,
  finalWait = FALSE
)
```

Arguments

`coin_list`	string if NULL retrieve all currently existing coins (`cg_list()`), or provide list of cryptocurrencies in the `cg_list()` / `cg_listings()` format.
`limit`	integer Return the top n records, default is all tokens.
`requestLimit`	Kept for parity with `crypto_info()` – ignored (CoinGecko endpoint is one coin per call).
`sleep`	integer (default `0`) Seconds to sleep between API requests. The internal client enforces a polite floor (default `2.5s`) to stay under the public API's rate limit.
`finalWait`	Sleep 60s after the last call (mirrors `crypto_info()`).

Details

Column names mirror `crypto_info()` where there is a direct equivalent, and add a few CG-specific fields (`platforms`, `categories`).

Value

Tibble with one row per coin:

`id`	CoinGecko internal numeric id (from `image$thumb`).
`name`, `symbol`, `slug`	Coin identifiers.
`category`	`"coin"` if `asset_platform_id` is `NA`, else `"token"`.
`description`	English description (HTML stripped to text).

logo	URL of the large coin logo.
status	Status notice from CoinGecko (public_notice field).
notice	Additional notices (additional_notices joined).
date_added	Date the coin was added to CoinGecko (genesis_date when set, else NA).
date_launched	Same as date_added for CG (no separate launch field exposed).
categories	List-column of coin categories.
platforms	List-column: per-chain contract addresses.
web_slug	CoinGecko canonical web slug (sometimes differs from API slug).
country_origin	Country of project origin (often empty).
sentiment_votes_up_percentage, sentiment_votes_down_percentage	Community sentiment percentages.
watchlist_portfolio_users	Number of CG users watching this coin.
url	List-column of resource URLs (homepage, whitepaper, blockchain explorers, forums, repos, chats, announcements, subreddit, twitter).

Examples

```
## Not run:
info <- cg_info(cg_list(top_n = 50))

## End(Not run)
```

---

cg_list — *Retrieves name, CG id, symbol, slug, rank, an activity flag as well as activity dates on CoinGecko for all coins*

Description

Companion to crypto_list() but for CoinGecko. Returns the active universe as a tibble using the same column conventions as crypto_list(), so downstream code that consumes a CMC coin list also consumes this one.

Usage

```
cg_list(only_active = TRUE, add_untracked = FALSE)
```

Arguments

only_active	Shall the code only retrieve active coins (TRUE = default) or include historically-known but currently-inactive coins (FALSE)? When FALSE, cg_id_mapping() is consulted for the extra slugs.
add_untracked	Kept for API parity with crypto_list() – CoinGecko does not have an "untracked" listing status, so the argument is silently ignored.

Details

Because CoinGecko prunes delisted coins from its public database, the free-tier API alone only returns coins currently active on the platform. When `only_active = FALSE`, the function transparently merges in historically-known coins via `cg_id_mapping()`; a single one-line message is emitted indicating how current that mapping is.

Value

List of (active and historically existing) cryptocurrencies in a tibble:

- `id` — CoinGecko internal numeric id (unique identifier).
- `name` — Coin name.
- `symbol` — Coin symbol (not-unique).
- `slug` — CoinGecko URL slug (unique).
- `rank` — Current market-cap rank on CoinGecko (`NA` for delisted or unranked coins).
- `is_active` — Flag showing whether the coin is currently active (`1`) or only historically present (`0`).
- `first_historical_data` — First time listed on CoinGecko (currently only populated for coins present in the historic mapping).
- `last_historical_data` — Last time listed on CoinGecko, today's date if still active.

Rate-limiting and retry parameters can be overridden globally via the package options `crypto2.cg_sleep`, `crypto2.cg_wait`, `crypto2.cg_max_retries` (defaults: 2.5s between calls, 60s wait before retry, 3 retries on rate-limit / network failure).

Examples

```
## Not run:
# all coins currently tracked by CoinGecko
active_list <- cg_list(only_active = TRUE)

# active + historically-listed coins (uses cg_id_mapping())
full_list <- cg_list(only_active = FALSE)

## End(Not run)
```

`cg_listings` *Retrieves name, CG id, symbol, slug, rank, and quote data for current listings (CoinGecko)*

Description

Companion to crypto_listings() but for CoinGecko. Returns one row per coin with every field of the /coins/markets endpoint: price, volume, 24h range, price and market-cap changes, all-time high/low and ROI. Column names mirror those of crypto_listings() where a CMC counterpart exists, so downstream code that already consumes a CMC listings tibble works on this tibble too.

Usage

```
cg_listings(
  which = "latest",
  convert = "USD",
  limit = NULL,
  start_date = NULL,
  end_date = NULL,
  interval = "day",
  quote = TRUE,
  sort = "cmc_rank",
  sort_dir = "asc",
  sleep = 0,
  wait = 60,
  finalWait = FALSE
)
```

Arguments

which	Always "latest" for CoinGecko free-tier. Other values produce a warning and are coerced to "latest".
convert	string (default: "USD"). The value is lower-cased and passed to CoinGecko as vs_currency. Common values: "USD", "BTC", "ETH", "EUR", "GBP".
limit	integer Return the top n records. NULL (default) returns all coins.
start_date, end_date, interval	Kept for API parity with crypto_listings() – ignored for CoinGecko (no historical-listings endpoint on the free tier).
quote	logical (default TRUE). The /coins/markets endpoint always returns prices at no extra cost, so they are included by default. Set to FALSE to keep only the identifier, rank, market-cap and supply columns (same default as crypto_listings()).
sort, sort_dir	Kept for parity. CoinGecko sorts by market_cap_desc on the underlying endpoint; the arguments are ignored.
sleep	integer (default 0) Seconds to sleep between API requests. Will be raised to at least getOption("crypto2.cg_sleep", 2.5) internally to stay under the public API's rate limit.
wait	Seconds to wait before retrying after a 429 (default 60).
finalWait	Sleep 60s after the last call (mirrors crypto_listings()).

Details

CoinGecko free-tier limitations: only `which = "latest"` is supported. `which = "historical"` and `which = "new"` produce a warning and are coerced to `"latest"`, because CoinGecko's free tier does not expose the historical cross-section. Snapshot this function periodically (daily / weekly via a cron job) to accumulate a survivorship-bias-corrected archive over time.

Coverage. `/coins/markets` no longer lists wrapped, staked or bridged tokens (e.g. stETH, wstETH, WBTC, JitoSOL, bridged USDT). Their history is still available through `cg_history()`, but they will not appear in a `cg_listings()` snapshot.

Rate limits. No API key is used; the public endpoint allows only a handful of calls per minute. HTTP 429 and transient 408/502/503/504 responses are retried with exponential backoff (see `wait` and the `crypto2.cg_max_retries` option). If a page still fails after the last retry, paging stops and a warning names the failed page, so an incomplete result is never returned silently.

Value

Tibble with one row per coin. Always present: `id`, `name`, `symbol`, `slug`, `date_added` (always `NA`), `last_updated` (POSIXct, UTC), `rank`, `market_cap`, `fully_diluted_market_cap`, `circulating_supply`, `total_supply`, `max_supply`. With `quote = TRUE` additionally (CoinGecko field name in brackets):

`price` [`current_price`]

`volume_24h` [`total_volume`]

`high_24h`, `low_24h`
: 24h price range

`price_change_24h`
: absolute 24h price change

`percent_change_1h`, `_24h`, `_7d`, `_14d`, `_30d`, `_200d`, `_1y`
: [`price_change_percentage_<window>_in_currency`]

`market_cap_change_24h`, `market_cap_change_percentage_24h`
: 24h market-cap change, absolute and in percent

`ath`, `ath_change_percentage`, `ath_date`
: all-time high; date as POSIXct (UTC)

`atl`, `atl_change_percentage`, `atl_date`
: all-time low; date as POSIXct (UTC)

`roi_times`, `roi_currency`, `roi_percentage`
: flattened `roi` object (`NA` for most coins)

`ref_currency` upper-cased `convert`

Examples

```
## Not run:
# Full current snapshot (all coins with a market cap), including prices
latest <- cg_listings()

# Top 1000 in BTC
latest_btc <- cg_listings(convert = "BTC", limit = 1000)

## End(Not run)
```

crypto_crosswalk *Crosswalk between CoinMarketCap and CoinGecko coin ids*

Description

Downloads the Open Crypto Pricing crosswalk, which links every CoinMarketCap id (as used by the `crypto_*` functions) to its CoinGecko coin (as used by the `cg_*` functions), including dead coins. Each pair is matched on contract addresses, project links, name/symbol and slug, cross-checked against DefiLlama, and confirmed on the price series of both providers. The file is updated weekly.

Usage

```
crypto_crosswalk(
  min_confidence = c("high", "medium", "low"),
  include_unmatched = FALSE,
  refresh = FALSE,
  quiet = FALSE
)
```

Arguments

`min_confidence` Lowest match confidence to keep, one of `"high"` (default), `"medium"` or `"low"`. Each level includes the ones above it, so `"low"` returns all published matches.

`include_unmatched` If `TRUE`, also return coins listed by only one provider (`match_confidence` `"cmc_only"` or `"cg_only"`), which have `NA` for the other provider's ids. Default `FALSE`.

`refresh` Download again even if the file is cached this session.

`quiet` Suppress the one-line source message.

Details

The file is downloaded once per session and cached in `tempdir()`. Parquet is used when the `arrow` package is installed, CSV otherwise.

Value

Tibble with one row per coin:

`uid` Open Crypto Pricing id, never reused (e.g. `"OCP004106"` for Bitcoin).

`cmc_id` CoinMarketCap id (the `id` column of `crypto_list()`).

`cg_id` CoinGecko slug (the `slug` column of `cg_list()`).

`cg_numeric_id` CoinGecko numeric id (the `id` column of `cg_list()`).

`name, symbol` Coin name and symbol.

`match_confidence` `"high"`, `"medium"` or `"low"`, or `"cmc_only"` / `"cg_only"` for unmatched coins.

`match_basis` The evidence that matched, joined by + (e.g. "contract+name_sym+slug+price").

`defillama_check`
"agrees", "disagrees", "suggests_pair_we_lack" or NA when DefiLlama has no entry.

`is_stablecoin`, `is_derived_token`, `is_tokenized_tradfi`
Logical flags.

Source

Stoeckl & Pukrop (2026), Open Crypto Pricing, https://opencryptopricing.com. Licensed under CC BY 4.0: cite the source when you use or redistribute the crosswalk.

Examples

```
## Not run:
cw <- crypto_crosswalk()

# CoinGecko history for the CMC top 100, via the crosswalk
top <- crypto_listings(limit = 100)
ids <- dplyr::inner_join(top, cw, by = c("id" = "cmc_id"))
hist <- cg_history(dplyr::select(ids, slug = cg_id, id = cg_numeric_id))

## End(Not run)
```

---

crypto_global_quotes *Retrieves historical quotes for the global aggregate market*

Description

This code retrieves global quote data (latest/historic) from coinmarketcap.com.

Usage

```
crypto_global_quotes(
  which = "latest",
  convert = "USD",
  start_date = NULL,
  end_date = NULL,
  interval = "daily",
  quote = FALSE,
  requestLimit = 2200,
  sleep = 0,
  wait = 60,
  finalWait = FALSE
)
```

Arguments

- `which` string Shall the code retrieve the latest listing or a historic listing?
- `convert` string (default: USD) to one or more of available fiat or precious metals prices (`fiat_list()`). If more than one are selected please separate by comma (e.g. "USD,BTC"), only necessary if 'quote=TRUE'
- `start_date` string Start date to retrieve data from, format 'yyyymmdd'
- `end_date` string End date to retrieve data from, format 'yyyymmdd', if not provided, today will be assumed
- `interval` string Interval with which to sample data, default 'daily'. Must be one of `"1d" "2d" "3d" "15d" "30d" "`
- `quote` logical set to TRUE if you want to include price data (FALSE=default)
- `requestLimit` integer (default 2200) Maximum number of requests one API call can handle
- `sleep` integer (default 0) Seconds to sleep between API requests
- `wait` waiting time before retry in case of fail (needs to be larger than 60s in case the server blocks too many attempts, default=60)
- `finalWait` to avoid calling the web-api again with another command before 60s are over (TRUE=default)

Value

List of latest/new/historic listings of crypto currencies in a tibble (depending on the "which"-switch and whether "quote" is requested, the result may only contain some of the following variables):

- `btc_dominance` number Bitcoin's market dominance percentage by market cap.
- `eth_dominance` number Ethereum's market dominance percentage by market cap.
- `active_cryptocurrencies` number Count of active crypto currencies tracked by CMC This includes all crypto currencies with a listing_status of "active" or "listed".
- `total_cryptocurrencies` number Count of all crypto currencies tracked by CMC This includes "inactive" listing_status crypto currencies.
- `active_market_pairs` number Count of active market pairs tracked by CoinMarketCap across all exchanges.

#'

- `active_exchanges` number Count of active exchanges tracked by CMC This includes all exchanges with a listing_status of "active" or "listed".
- `total_exchanges` number Count of all exchanges tracked by CMC This includes "inactive" listing_status exchanges.
- `last_updated` Timestamp of when this record was last updated.
- `total_market_cap` number The sum of all individual cryptocurrency market capitalizations in the requested currency.

`total_volume_24h`
: number The sum of rolling 24 hour adjusted volume (as outlined in our methodology) for all crypto currencies in the requested currency.

`total_volume_24h_reported`
: number The sum of rolling 24 hour reported volume for all crypto currencies in the requested currency.

#'

`altcoin_volume_24h`
: number The sum of rolling 24 hour adjusted volume (as outlined in our methodology) for all crypto currencies excluding Bitcoin in the requested currency.

`altcoin_volume_24h_reported`
: number The sum of rolling 24 hour reported volume for all crypto currencies excluding Bitcoin in the requested currency.

`altcoin_market_cap`
: number The sum of all individual cryptocurrency market capitalizations excluding Bitcoin in the requested currency.

Examples

```
## Not run:
# return new listings from the last 30 days
new_quotes <- crypto_global_quotes(which="latest", quote=TRUE, convert="BTC")
# return all global quotes in the first week of January 2014
quotes_2014w1 <- crypto_global_quotes(which="historical", quote=TRUE,
  start_date = "20140101", end_date="20140107", interval="daily")

# report in two different currencies
listings_2014w1_USDBTC <- crypto_global_quotes(which="historical", quote=TRUE,
  start_date = "20200101", end_date="20240530", interval="daily", convert="BTC")

## End(Not run)
```

`crypto_history` *Get historic crypto currency market data*

Description

Scrape the crypto currency historic market tables from 'CoinMarketCap' https://coinmarketcap.com and display the results in a dataframe/tibble. This can be used to conduct analysis on the crypto financial markets or to attempt to predict future market movements or trends.

Usage

```
crypto_history(
  coin_list = NULL,
  convert = "USD",
  limit = NULL,
  start_date = NULL,
  end_date = NULL,
  interval = NULL,
  requestLimit = 400,
  sleep = 0,
  wait = 60,
  finalWait = FALSE,
  single_id = TRUE
)
```

Arguments

`coin_list`	string if NULL retrieve all currently existing coins (`crypto_list()`), or provide list of crypto currencies in the `crypto_list()` or `cryptoi_listings()` format (e.g. current and/or dead coins since 2015)
`convert`	(default: USD) to one of available fiat prices (`fiat_list()`) or bitcoin 'BTC'. Be aware, that since 2024 only USD and BTC are available here!
`limit`	integer Return the top n records, default is all tokens
`start_date`	date Start date to retrieve data from
`end_date`	date End date to retrieve data from, if not provided, today will be assumed
`interval`	string Interval with which to sample data according to what `seq()` needs
`requestLimit`	limiting the length of request URLs when bundling the api calls
`sleep`	integer (default 60) Seconds to sleep between API requests
`wait`	waiting time before retry in case of fail (needs to be larger than 60s in case the server blocks too many attempts, default=60)
`finalWait`	to avoid calling the web-api again with another command before 60s are over (TRUE=default)
`single_id`	Download data coin by coin (as of May 2024 this is necessary)

Value

Crypto currency historic OHLC market data in a dataframe and additional information via attribute "info":

`timestamp`	Timestamp of entry in database
`id`	Coin market cap unique id
`name`	Coin name
`symbol`	Coin symbol
`ref_cur_id`	reference Currency id

ref_cur_name	reference Currency name
open	Market open
high	Market high
low	Market low
close	Market close
volume	Volume 24 hours
market_cap	Market cap - close x circulating supply
time_open	Timestamp of open
time_close	Timestamp of close
time_high	Timestamp of high
time_low	Timestamp of low

This is the main function of the crypto package. If you want to retrieve ALL active coins then do not pass an argument to `crypto_history()`, alternatively pass the coin name.

Examples

```
## Not run:

# Retrieving market history for ALL crypto currencies
all_coins <- crypto_history(limit = 2)
one_coin <- crypto_history(limit = 1, convert="BTC")

# Retrieving market history since 2020 for ALL crypto currencies
all_coins <- crypto_history(start_date = '2020-01-01',limit=10)

# Retrieve 2015 history for all 2015 crypto currencies
coin_list_2015 <- crypto_list(only_active=TRUE) %>%
              dplyr::filter(first_historical_data<="2015-12-31",
              last_historical_data>="2015-01-01")
coins_2015 <- crypto_history(coin_list = coin_list_2015,
             start_date = "2015-01-01", end_date="2015-12-31", limit=20, interval="30d")
# retrieve hourly bitcoin data for 2 days
btc_hourly <- crypto_history(coin_list = coin_list_2015,
              start_date = "2015-01-01", end_date="2015-01-03", limit=1, interval="1h")


## End(Not run)
```

`crypto_info` *Retrieves info (urls, logo, description, tags, platform, date_added, notice, status,...) on CMC for given id*

Description

This code retrieves data for all specified coins!

Usage

```
crypto_info(
  coin_list = NULL,
  limit = NULL,
  requestLimit = 1,
  sleep = 0,
  finalWait = FALSE
)
```

Arguments

`coin_list`	string if NULL retrieve all currently active coins (`crypto_list()`), or provide list of cryptocurrencies in the `crypto_list()` or `cryptoi_listings()` format (e.g. current and/or dead coins since 2015)
`limit`	integer Return the top n records, default is all tokens
`requestLimit`	(default: 1) limiting the length of request URLs when bundling the api calls (currently needs to be 1)
`sleep`	integer (default: 0) Seconds to sleep between API requests
`finalWait`	to avoid calling the web-api again with another command before 60s are over (FALSE=default)

Value

List of (active and historically existing) cryptocurrencies in a tibble:

`id`	CMC id (unique identifier)
`name`	Coin name
`symbol`	Coin symbol (not-unique)
`slug`	Coin URL slug (unique)
`category`	Coin category: "token" or "coin"
`description`	Coin description according to CMC
`logo`	CMC url of CC logo
`status`	Status message from CMC
`notice`	Markdown formatted notices from CMC
`alert_type`	Type of alert on CMC
`alert_link`	Message link to alert
`date_added`	Date CC was added to the CMC database
`date_launched`	Date CC was launched
`is_audited`	Boolean if CC is audited
`flags`	Boolean flags for various topics
`self_reported_circulating_supply`	Self reported circulating supply
`tags`	Tibble of tags and tag categories

`faq_description`
: FAQ description from CMC

`url`
: Tibble of various resource urls. Gives website, technical_doc (whitepaper), source_code, message_board, chat, announcement, reddit, twitter, (block) explorer urls

`platform`
: Metadata about the parent coin if available. Gives id, name, symbol, slug, and token address according to CMC

Examples

```
## Not run:
# return info for bitcoin
coin_info <- crypto_info(limit=10)

## End(Not run)
```

`crypto_list` *Retrieves name, CMC id, symbol, slug, rank, an activity flag as well as activity dates on CMC for all coins*

Description

This code uses the web api. It retrieves data for all historic and all active coins and does not require an 'API' key.

Usage

```
crypto_list(only_active = TRUE, add_untracked = FALSE)
```

Arguments

`only_active`
: Shall the code only retrieve active coins (TRUE=default) or include inactive coins (FALSE)

`add_untracked`
: Shall the code additionally retrieve untracked coins (FALSE=default)

Value

List of (active and historically existing) cryptocurrencies in a tibble:

`id`
: CMC id (unique identifier)

`name`
: Coin name

`symbol`
: Coin symbol (not-unique)

`slug`
: Coin URL slug (unique)

`rank`
: Current rank on CMC (if still active)

`is_active`
: Flag showing whether coin is active (1), inactive(0) or untracked (-1)

`first_historical_data`
: First time listed on CMC

`last_historical_data`
: Last time listed on CMC, *today's date* if still listed

Examples

```
## Not run:
# return all coins
active_list <- crypto_list(only_active=TRUE)
all_but_untracked_list <- crypto_list(only_active=FALSE)
full_list <- crypto_list(only_active=FALSE,add_untracked=TRUE)

# return all coins active in 2015
coin_list_2015 <- active_list %>%
dplyr::filter(first_historical_data<="2015-12-31",
              last_historical_data>="2015-01-01")

## End(Not run)
```

`crypto_listings` *Retrieves name, CMC id, symbol, slug, rank, an activity flag as well as activity dates on CMC for all coins*

Description

This code retrieves listing data (latest/new/historic).

Usage

```
crypto_listings(
  which = "latest",
  convert = "USD",
  limit = NULL,
  start_date = NULL,
  end_date = NULL,
  interval = "day",
  quote = TRUE,
  sort = "cmc_rank",
  sort_dir = "asc",
  sleep = 0,
  wait = 60,
  finalWait = FALSE
)
```

Arguments

`which`	string Shall the code retrieve the latest listing, the new listings or a historic listing?
`convert`	string (default: USD) to one of available fiat prices (`fiat_list()`). If more than one are selected please separate by comma (e.g. "USD,BTC"), only necessary if 'quote=TRUE'
`limit`	integer Return the top n records per listing (per day for `which = "historical"`). `NULL` (default) returns all coins. For `which = "historical"`, a day that reaches `limit` exactly is flagged as possibly truncated in a warning.
`start_date`	string Start date to retrieve data from, format 'yyyymmdd'
`end_date`	string End date to retrieve data from, format 'yyyymmdd', if not provided, today will be assumed
`interval`	string Interval with which to sample data according to what `seq()` needs
`quote`	logical (default `TRUE`): include price, volume, market-cap and percent-change columns. They come with the same API response at no extra cost. With `FALSE` only identifiers, ranks, dates and supply are returned, and there is no `price` column.
`sort`	(May 2024: currently not available) string use to sort results, possible values: "name", "symbol", "market_cap", "price", "circulating_supply", "total_supply", "max_supply", "num_market_pairs", "volume_24h", "volume_7d", "volume_30d", "percent_change_1h", "percent_change_24h", "percent_change_7d". Especially useful if you only want to download the top x entries using "limit" (deprecated for "new")
`sort_dir`	(May 2024: currently not available) string used to specify the direction of the sort in "sort". Possible values are "asc" (DEFAULT) and "desc"
`sleep`	integer (default 0) Seconds to sleep between API requests
`wait`	waiting time before retry in case of fail (needs to be larger than 60s in case the server blocks too many attempts, default=60)
`finalWait`	to avoid calling the web-api again with another command before 60s are over (TRUE=default)

Details

For `which = "historical"` each day is fetched in pages of 5,000 coins and retried as a whole if a page fails. A day is returned complete or not at all: a day that still fails after the retries is dropped and named in a warning, never returned truncated.

CMC itself has been observed to return an incomplete day without any error: for 2024-01-07, requests from GitHub Actions runners received 2,552 coins while the same request from a desktop received all 9,002. When you run large downloads from a server, compare daily row counts with a run from another machine.

Value

List of latest/new/historic listings of cryptocurrencies in a tibble (depending on the "which"-switch and whether "quote" is requested, the result may only contain some of the following variables):

`id` CMC id (unique identifier)

`name` Coin name

`symbol` Coin symbol (not-unique)

`slug` Coin URL slug (unique)

`date_added` Date when the coin was added to the dataset

`last_updated` Last update of the data in the database

`rank` Current rank on CMC (if still active)

`market_cap` market cap - close x circulating supply

`market_cap_by_total_supply` market cap - close x total supply

`market_cap_dominance` market cap dominance

`fully_diluted_market_cap` fully diluted market cap

`self_reported_market_cap` is the source of the market cap self-reported

`self_reported_circulating_supply` is the source of the circulating supply self-reported

`tvl_ratio` percentage of total value locked

`price` latest average price

`circulating_supply` approx. number of coins in circulation

`total_supply` approx. total amount of coins in existence right now (minus any coins that have been verifiably burned)

`max_supply` CMC approx. of max amount of coins that will ever exist in the lifetime of the currency

`num_market_pairs` number of market pairs across all exchanges this coin

`tvl` total value locked

`volume_24h` Volume 24 hours

`volume_change_24h` Volume change in 24 hours

`percent_change_1h` 1 hour return

`percent_change_24h` 24 hour return

`percent_change_7d` 7 day return

`percent_change_30d` 30 day return

`percent_change_60d` 60 day return

`percent_change_90d` 90 day return

Examples

```
## Not run:
# return new listings from the last 30 days
new_listings <- crypto_listings(which="new", quote=FALSE, limit=50000)
new_listings2 <- crypto_listings(which="new", quote=TRUE, convert="BTC")

# return latest listing (last available data of all CC including quotes)
latest_listings <- crypto_listings(which="latest", quote=FALSE, limit=50000)
latest_listings2 <- crypto_listings(which="latest", quote=TRUE, convert="BTC")

# return the first 10 listings in the first week of January 2024
listings_2024w1 <- crypto_listings(which="historical", quote=TRUE,
      start_date = "20240101", end_date="20240102", interval="day", limit=10)

# only download the top 10 crypto currencies based on their market capitalization
# DOES NOT WORK ANY MORE

# for historically accurate snapshots (e.g. for backtesting crypto investments)
# you need to download the entire history on one day including price information:
listings_20200202 <- crypto_listings(which="historical", quote=TRUE,
      start_date="20200202", end_date="20200202")
listings_20240202 <- crypto_listings(which="historical", quote=TRUE,
      start_date="20240202", end_date="20240202", limit=50000)
# note the much larger amount in CCs in 2024, as well as the existence of many
more variables in the dataset

## End(Not run)
```

`exchange_info` *Retrieves info (urls,logo,description,tags,platform,date_added,notice,status) on CMC for given exchange slug*

Description

This code uses the web api. It retrieves data for all active, delisted and untracked exchanges! It does not require an 'API' key.

Usage

```
exchange_info(
  exchange_list = NULL,
  limit = NULL,
  requestLimit = 1,
  sleep = 0,
  finalWait = FALSE
)
```

Arguments

`exchange_list`	string if NULL retrieve all currently active exchanges (`exchange_list()`), or provide list of exchanges in the `exchange_list()` format (e.g. current and/or delisted)
`limit`	integer Return the top n records, default is all exchanges
`requestLimit`	limiting the length of request URLs when bundling the api calls
`sleep`	integer (default 60) Seconds to sleep between API requests
`finalWait`	to avoid calling the web-api again with another command before 60s are over (TRUE=default)

Value

List of (active and historically existing) exchanges in a tibble:

`id`	CMC exchange id (unique identifier)
`name`	Exchange name
`slug`	Exchange URL slug (unique)
`description`	Exchange description according to CMC
`notice`	Exchange notice (markdown formatted) according to CMC
`logo`	CMC url of CC logo
`type`	Type of exchange
`date_launched`	Launch date of this exchange
`is_hidden`	TBD
`is_redistributable`	TBD
`maker_fee`	Exchanges maker fee
`taker_fee`	Exchanges maker fee
`platform_id`	Platform id on CMC
`dex_status`	Decentralized exchange status
`wallet_source_status`	Wallet source status
`status`	Activity status on CMC
`tags`	Tibble of tags and tag categories
`urls`	Tibble of various resource urls. Gives website, blog, fee, twitter.
`countries`	Tibble of countries the exchange is active in
`fiats`	Tibble of fiat currencies the exchange trades in

Examples

```
## Not run:
# return info for the first three exchanges
exchange_info <- exchange_info(limit=10)

## End(Not run)
```

`exchange_list` *Retrieves name, CMC id, symbol, slug, rank, an activity flag as well as activity dates on CMC for all coins*

Description

This code uses the web api. It retrieves data for all historic and all active exchanges and does not require an 'API' key.

Usage

```
exchange_list(only_active = TRUE, add_untracked = FALSE)
```

Arguments

`only_active`	Shall the code only retrieve active exchanges (TRUE=default) or include inactive coins (FALSE)
`add_untracked`	Shall the code additionally retrieve untracked exchanges (FALSE=default)

Value

List of (active and historically existing) exchanges in a tibble:

`id`	CMC exchange id (unique identifier)
`name`	Exchange name
`slug`	Exchange URL slug (unique)
`is_active`	Flag showing whether exchange is active (1), inactive(0) or untracked (-1)
`first_historical_data`	First time listed on CMC
`last_historical_data`	Last time listed on CMC, *today's date* if still listed

Examples

```
## Not run:
# return all exchanges
ex_active_list <- exchange_list(only_active=TRUE)
ex_all_but_untracked_list <- exchange_list(only_active=FALSE)
ex_full_list <- exchange_list(only_active=FALSE,add_untracked=TRUE)

## End(Not run)
```

`fiat_list` *Retrieves list of all CMC supported fiat currencies available to convert cryptocurrencies*

Description

This code retrieves data for all available fiat currencies that are available on the website.

Usage

```
fiat_list(include_metals = FALSE)
```

Arguments

`include_metals`	Shall the results include precious metals (TRUE) or not (FALSE=default). Update: As of May 2024 no more metals are included in this file

Value

List of (active and historically existing) cryptocurrencies in a tibble:

`id`	CMC id (unique identifier)
`symbol`	Coin symbol (not-unique)
`name`	Coin name
`sign`	Fiat currency sign

Examples

```
## Not run:
# return fiat currencies available through the CMC api
fiat_list <- fiat_list()

## End(Not run)
```

Index

cg_history, 2
cg_history(), *4, 5*, *7*, *12*
cg_history_by_id, 4
cg_id_mapping, 7
cg_id_mapping(), *3*, *6*, *9, 10*
cg_info, 8
cg_list, 9
cg_list(), *3*, *7, 8*, *13*
cg_listings, 10
cg_listings(), *3*, *8*
crypto_crosswalk, 13
crypto_global_quotes, 14
crypto_history, 16
crypto_history(), *2, 3*
crypto_info, 18
crypto_info(), *8*
crypto_list, 20
crypto_list(), *9*, *13*
crypto_listings, 21
crypto_listings(), *11*

exchange_info, 24
exchange_list, 26

fiat_list, 27