coinmetrics

Release 0.2.5

Robert Rice

Apr 25, 2020

CONTENTS:

Note: This module only impliments the Community API (i.e. anything without an API key).

This API for Python 3 provides a convenient interface for developers wanting to use Coin Metrics data.

- The code supporting this can be on Github.
- Batteries (i.e. Python *API Reference* and *Usage* examples) are included.
- Setup instructions are provided on the *Installation* page.

CHAPTER

ONE

INSTALLATION

1.1 Install with *pip*

```
pip install coinmetrics
```

1.2 Manual Install

```
git clone https://github.com/h4110w33n/coinmetrics
cd coinmetrics
python setup.py install
```

Now that you have everything installed, you can refer to the *Usage* page for some code snippets.

CHAPTER

TWO

API REFERENCE

The Coin Metrics API provides discovery mechanisms, network, exchange, and asset data via a direct JSON REST API. This module acts as a client interface.

The *Base* and *Community* APIs are available to all users without an API key. The *Pro* API does require an API key that is only obtainable though CM Network Data Pro.

Note: The following arguments can consist of a single element or a comma delimited list within a string. A Python `list` is not valid at this time.

- Asset: `btc` or `btc,ltc,eth`
- Metric: `PriceUSD` or `NVTAdj,NVTAdj90,PriceUSD`
- Exchange: `coinbase` or `bitfinex,coinbase,kraken`
- Market: `bitfinex-btc-usd-spot` or `bitfinex-btc-usd-spot, bitfinex-eth-usd-spot, kraken-btc-usd-spot`

2.1 Base

This is the core definition for all v2+ Coinmetrics APIs, both Community and Pro. This object includes the primary query device, all **_checker* functions and the necessary discovery methods to enable the them.

2.1.1 Primary Methods

class coinmetrics.base.**Base**(*api_key=''*)

Coin Metrics API Base Object

__init__(*api_key=''*)

Initialize API to use the Base API endpoints by default. An optional `api_key` can be supplied.

Parameters **api_key** (*str, optional*) – API key to be used for the Pro API.

_api_query(*endpoint*, *options=None*)

Execute the raw API query and return the raw JSON output.

Parameters

- **endpoint** (*string*) – URL Path the query will be sent to. This includes any URL based parameters.

- **options** (*dict, optional*) – Query parameters, including asset(s), metric(s), exchanges(s), and time range.

Returns Raw JSON response as dict.

Return type dict

get_assets()
Fetch list of available assets.

Returns List of supported assets.

Return type list

asset_checker(*assets*)
Helper function to determine if the requested asset(s) is(are) valid.

Parameters **asset** (*str*) – Unique ID corresponding to the asset's ticker.

Raises InvalidAssetError

get_metrics()
Fetch list of available metrics.

Returns List of supported metrics.

Return type list

metric_checker(*metrics*)
Helper function to determine if the requested metric(s) is(are) valid.

Parameters **metrics** (*str*) – Unique ID corresponding to metric.

Raises InvalidMetricError

get_exchanges()
Fetch list of available exchanges.

Returns List of supported exchanges.

Return type list

exchange_checker(*exchanges*)
Helper function to determine if the requested exchange(s) is(are) valid.

Parameters **exchanges** (*str*) – Unique ID corresponding to the exchange.

Raises InvalidExchangeError

get_markets()
Fetch list of available markets.

Returns List of supported markets.

Return type list

market_checker(*markets*)
Helper function to determine if the requested market(s) is(are) valid.

Parameters **market** (*str*) – Unique ID corresponding to the market.

Raises InvalidMarketError

timestamp_checker(*begin_timestamp*, *end_timestamp*)
Helper function to determine if the provided timerange is valid.

Parameters

- **begin_timestamp** (*str or datetime*) – Start of time inverval.
- **end_timestamp** (*str or datetime*) – End of time inverval.

Raises InvalidTimeRangeError

2.1.2 Alias Methods

A group of alternative methods that function identically to the references functions above. This is to support any legacy API method names.

class coinmetrics.base.**Base**(*api_key=''*)
Coin Metrics API Base Object

get_supported_assets()
An alias for *get_assets()*

assets()
An alias for *get_assets()*

getmetrics()
An alias for *get_metrics()*

metrics()
An alias for *get_metrics()*

getexchanges()
An alias for *get_exchanges()*

exchange()
An alias for *get_exchanges()*

getmarkets()
An alias for *get_markets()*

markets()
An alias for *get_markets()*

2.2 Community

The `Community` class is an extension of the the *Base* class and in inherits all of the `Base` class' functionality.

2.2.1 Primary Methods

class coinmetrics.community.**Community**(*api_key=''*)
Coin Metrics API Community Object

__init__(*api_key=''*)
Initialize the Community API exactly the same way as the same way as *coinmetrics.base.Base.__init__()*. An optional `api_key` can be supplied.

Parameters **api_key** (*str, optional*) – API key to be used for the Pro API.

get_asset_info(*assets=''*)
Fetch asset(s) information. Including exchanges the asset is in, friendly name, markets, metrics, and most recent and oldest data timestamps. No specified asset will return all available asset info.

Parameters **assets** (*str, optional*) – Unique ID corresponding to the asset's ticker.

Returns Asset information.

Return type list of dict

get_exchange_info(*exchanges=''*)
Fetch exchange(s) information. Including assets, quotes, markets, and most recent and oldest data timestamps for each. No specified exchange will return all available exchange info.

Parameters **exchanges** (*str, optional*) – Unique ID corresponding to the exchange.

Returns Exchange information.

Return type list of dict

get_metric_info(*metrics=''*)
Fetch metric(s) information. Including friendly name, dscription, category.

Parameters **metrics** (*str, optional*) – Unique ID corresponding to the metrics.

Returns Metric information.

Return type list of dict

get_market_info(*markets=''*)
Fetch market(s) information. Includes assets, quotes, and most recent and oldest data timestamps for each.

Parameters **markets** (*str, optional*) – Unique ID corresponding to the market.

Returns Market information.

Return type list of dict

get_asset_metric_data(*asset*, *metrics*, *start*, *end*, *time_agg='day'*)
Fetch metric(s) data given a specified asset, and timeframe. See: Data Dictionary.

Parameters

- **asset** (*str*) – Unique ID corresponding to the asset's ticker.
- **metrics** (*str*) – Unique ID corresponding to the metrics.
- **begin_timestamp** (*str or datetime*) – Start of time inverval.
- **end_timestamp** (*str or datetime*) – End of time inverval.
- **time_agg** (*str*) – Interval the time is descritized into: day, hour.

Returns Coin Metrics API data object. See the API reference for details

Return type dict

2.2.2 Conveniance Methods

The methods are designed to provide a simple way to get a single metric, and essentially extend the *coinmetrics.community.Community.get_asset_metric_data()* method.

class coinmetrics.community.**Community**(*api_key=''*)
Coin Metrics API Community Object

get_available_data_types_for_asset(*asset*)
An alias for `get_asset_metrics()` (backwards compatibility)

get_asset_data_for_time_range(*asset*, *metrics*, *start*, *end*, *time_agg='day'*)
An alias for *get_asset_metric_data()* (backwards compatibility)

get_active_addresses (*assets*, *start*, *end*)
The sum count of unique addresses that were active in the network (either as a recipient or originator of a ledger change) that day. All parties in a ledger change action (recipients and originators) are counted. Individual addresses are not double-counted if previously active. See: Data Dictionary.

Parameters See see *get_asset_metric_data()* for parameter and return details.

get_block_count (*assets*, *start*, *end*)
The sum count of unique addresses that were active in the network (either as a recipient or originator of a ledger change) that day. All parties in a ledger change action (recipients and originators) are counted. Individual addresses are not double-counted if previously active. See: Data Dictionary.

Parameters See see *get_asset_metric_data()* for parameter and return details.

get_mean_block_size (*assets*, *start*, *end*)
The mean size (in bytes) of all blocks created that day. See: Data Dictionary.

Parameters See see *get_asset_metric_data()* for parameter and return details.

get_mvrv_cur (*assets*, *start*, *end*)
The ratio of the sum USD value of the current supply to the sum "realized" USD value of the current supply. See: Data Dictionary.

Parameters See see *get_asset_metric_data()* for parameter and return details.

get_real_cap (*assets*, *start*, *end*)
The sum USD value based on the USD closing price on the day that a native unit last moved (i.e., last transacted) for all native units. See: Data Dictionary.

Parameters See see *get_asset_metric_data()* for parameter and return details.

get_diff_mean (*assets*, *start*, *end*)
The mean difficulty of finding a hash that meets the protocol designated requirement (i.e., the difficulty of finding a new block) that day. The requirement is unique to each applicable cryptocurrency protocol. Difficulty is adjusted periodically by the protocol as a function of how much hashing power is being deployed by miners. See: Data Dictionary.

Parameters See see *get_asset_metric_data()* for parameter and return details.

get_fee_mean (*assets*, *start*, *end*)
The USD value of the mean fee per transaction that day. See: Data Dictionary.

Parameters See see *get_asset_metric_data()* for parameter and return details.

get_median_fee (*assets*, *start*, *end*)
The USD value of the median fee per transaction that day. See: Data Dictionary.

Parameters See see *get_asset_metric_data()* for parameter and return details.

get_fee_total (*assets*, *start*, *end*)
The sum USD value of all fees paid to miners that day. Fees do not include new issuance. See: Data Dictionary.

Parameters See see *get_asset_metric_data()* for parameter and return details.

get_units_issued (*assets*, *start*, *end*)
The sum of new native units issued that day. Only those native units that are issued by a protocol-mandated continuous emission schedule are included. See: Data Dictionary.

Parameters See see *get_asset_metric_data()* for parameter and return details.

get_units_issued_ann_pct (*assets*, *start*, *end*)
The percentage of new native units (continuous) issued on that day, extrapolated to one year (i.e., multiplied

by 365), and divided by the current supply on that day. Also referred to as the annual inflation rate. See: Data Dictionary.

Parameters See see *get_asset_metric_data()* for parameter and return details.

get_units_issued_usd(*assets*, *start*, *end*)
The sum USD value of all new native units issued that day. See: Data Dictionary.

Parameters See see *get_asset_metric_data()* for parameter and return details.

get_nvt_adj(*assets*, *start*, *end*)
The ratio of the network value (or market capitalization, current supply) divided by the adjusted transfer value. Also referred to as NVT. See: Data Dictionary.

Parameters See see *get_asset_metric_data()* for parameter and return details.

get_nvt_adj_90(*assets*, *start*, *end*)
The ratio of the network value (or market capitalization, current supply) to the 90-day moving average of the adjusted transfer value. Also referred to as NVT. See: Data Dictionary.

Parameters See see *get_asset_metric_data()* for parameter and return details.

get_price_btc(*assets*, *start*, *end*)
The fixed closing price of the asset as of 00:00 UTC the following day (i.e., midnight UTC of the current day) denominated in USD. This price is generated by Coin Metrics' fixing/reference rate service. See: Data Dictionary.

Parameters See see *get_asset_metric_data()* for parameter and return details.

get_price_usd(*assets*, *start*, *end*)
The fixed closing price of the asset as of 00:00 UTC the following day (i.e., midnight UTC of the current day) denominated in BTC. See: Data Dictionary.

Parameters See see *get_asset_metric_data()* for parameter and return details.

get_cur_sply(*assets*, *start*, *end*)
The sum of all native units ever created and visible on the ledger (i.e., issued) as of that day. For account-based protocols, only accounts with positive balances are counted. See: Data Dictionary.

Parameters See see *get_asset_metric_data()* for parameter and return details.

get_tx_count(*assets*, *start*, *end*)
The sum count of transactions that day. Transactions represent a bundle of intended actions to alter the ledger initiated by a user (human or machine). See: Data Dictionary.

Parameters See see *get_asset_metric_data()* for parameter and return details.

get_txtfr_count(*assets*, *start*, *end*)
The sum count of transfers that day. Transfers represent movements of native units from one ledger entity to another distinct ledger entity. See: Data Dictionary.

Parameters See see *get_asset_metric_data()* for parameter and return details.

get_txtfr_val_adj(*assets*, *start*, *end*)
The sum of native units transferred that day removing noise and certain artifacts. See: Data Dictionary.

Parameters See see *get_asset_metric_data()* for parameter and return details.

get_txtfr_val_adj_usd(*assets*, *start*, *end*)
The USD value of the sum of native units transferred that day removing noise and certain artifacts. See: Data Dictionary.

Parameters See see *get_asset_metric_data()* for parameter and return details.

get_txtfr_val_mean(*assets*, *start*, *end*)
The mean count of native units transferred per transaction (i.e., the mean "size" of a transaction) that day. See: Data Dictionary.

Parameters See see *get_asset_metric_data()* for parameter and return details.

get_txtfr_val_mean_usd(*assets*, *start*, *end*)
The sum USD value of native units transferred divided by the count of transfers (i.e., the mean "size" in USD of a transfer) that day. See: Data Dictionary.

Parameters See see *get_asset_metric_data()* for parameter and return details.

get_txtfr_val_med(*assets*, *start*, *end*)
The median count of native units transferred per transfer (i.e., the median "size" of a transfer) that day. See: Data Dictionary.

Parameters See see *get_asset_metric_data()* for parameter and return details.

get_txtfr_val_med_usd(*assets*, *start*, *end*)
The median USD value transferred per transfer (i.e., the median "size" in USD of a transfer) that day. See: Data Dictionary.

Parameters See see *get_asset_metric_data()* for parameter and return details.

get_txtfr_val(*assets*, *start*, *end*)
The sum of native units transferred (i.e., the aggregate "size" of all transfers) that day. See: Data Dictionary.

Parameters See see *get_asset_metric_data()* for parameter and return details.

get_txtfr_val_usd(*assets*, *start*, *end*)
The sum USD value of all native units transferred (i.e., the aggregate size in USD of all transfers) that day. See: Data Dictionary.

Parameters See see *get_asset_metric_data()* for parameter and return details.

get_vty_ret_180d(*assets*, *start*, *end*)
The 180D volatility, measured as the deviation of log returns See: Data Dictionary.

Parameters See see *get_asset_metric_data()* for parameter and return details.

get_vty_ret_30d(*assets*, *start*, *end*)
The 30D volatility, measured as the deviation of log returns See: Data Dictionary.

Parameters See see *get_asset_metric_data()* for parameter and return details.

get_vty_ret_60d(*assets*, *start*, *end*)
The 60D volatility, measured as the deviation of log returns. See: Data Dictionary.

Parameters See see *get_asset_metric_data()* for parameter and return details.

Note: The conveniance methods are not explicitly included in the coverage testing at this time.

2.2.3 Alias Methods

There are a very large number of aliases for the *Conveniance Methods* above, that can be found in the code under each method definition. Below is an example of the aliases for `get_active_addresses()`.

```
# Method definition:
def get_active_addresses(self, assets, start, end):
    return self.get_asset_metric_data(assets, "AdrActCnt", start, end)

# Alias definition:
active_addresses, activeaddresses, AdrActCnt = [get_active_addresses] * 3
```

The complete list of available aliases can be found using the following lines of code.

```
# Import the library
import coinmetrics

# Write out the complete method/atribute listing for the library
print(dir(coinmetrics))
```

2.3 Pro

Note: The `Pro` class of APIs are planned but not currently available through this API library. For information regarding the development of the ProAPIs, please see the related GitHub Issue.

2.4 Utilities

Coin Metrics API Conversion Utilities

The functions below are not internally used by the `coinmetrics` library. They are transformations to allow data that was fetched to be more easily processed by another process.

Usage Examples: *Usage*

`coinmetrics.utils.`**`cm_to_pandas`**(*data*)

Convert an object output from *coinmetrics.community.Community.get_asset_metric_data()* to a Pandas object for further processing.

Parameters **`object`** (*dict*) – Raw data object to convert to Pandas datagram.

Returns Pandas dataframe form of original object.

Return type pandas dataframe

`coinmetrics.utils.`**`normalize`**(*data*)

Convert an object output from *coinmetrics.community.Community.get_asset_metric_data()* to a standard list of dictionaries for further processing.

Parameters **`object`** (*dict*) – Raw data object to convert to list of dict.

Returns A normalized list of dictionaries

Return type list

`coinmetrics.utils.`**`csv`**(*data*, *path*)

Convert an object output from *coinmetrics.community.Community.get_asset_metric_data()* to a standard list of dictionaries for further processing.

Parameters

- **object** (*dict*) – Raw data object to convert to Pandas datagram.
- **path** (*str, optional*) – Location to save the CSV file to.

CHAPTER

THREE

USAGE

3.1 Fetching Data

```
# API Setup ###################################################################

# Import the API
import coinmetrics

# Initialize a reference object, in this case `cm` for the Community API
cm = coinmetrics.Community()

# Usage Examples ##############################################################

# List the assets Coin Metrics has data for.
supported_assets = cm.get_supported_assets()
print("supported assets:\n", supported_assets)

# List all available metrics for BTC.
asset = "btc"
available_data_types = cm.get_available_data_types_for_asset(asset)
print("available data types:\n", available_data_types)

# Fetch the `PriceUSD` and `ROI30d` data for BTC from 2019-01-01 to 2019-01-08.
asset = "btc"
metric = "PriceUSD,ROI30d"
begin_timestamp = "2019-01-01"  # The `datetime` type is also accepted
end_timestamp = "2019-01-08"  # The `datetime` type is also accepted
asset_data = cm.get_asset_data_for_time_range(asset, metric, begin_timestamp, end_
↪timestamp)
print("data given timerange:\n", asset_data)
```

3.2 Transforming Data

```
# Convert the data object we recieved to a Pandas DataFrame for further processing.
# We are reusing the `asset_data` from the previous step.
pandas_data_frame = coinmetrics.cm_to_pandas(asset_data)
print("pandas data frame:\n", pandas_data_frame)

# Save the Pandas DataFrame OR raw Coimetrics object into a CSV.
# We are resuing the `pandas_data_frame` from the previous step, but the `asset_data`
# from the step before that is also a valid input.
```

(continues on next page)

(continued from previous page)

```
path_or_filename = "output.csv"
coinmetrics.csv(pandas_data_frame, path_or_filename)
print("CSV written:", path_or_filename)
```

From here, the sky is the limit. Apply your logic and profit.

CHAPTER

FOUR

THANKS

A huge thanks to Coin Metrics (coinmetrics.io) for providing a community edition of the API and providing the world with a bit more trustworthy data in an age of misinformation.

The official API reference that this project is based on can be found on Coin Metrics' API Reference

PYTHON MODULE INDEX

C

INDEX

Symbols

A

B

C

E

G

M

N

T