Package ‘cryptoQuotes’

December 13, 2023

Title  Access OHLC Market Data from Major Cryptocurrency Exchanges

Version  1.0.0

Description  A high level library to extract cryptocurrency OHLC market data from major central-
ized exchanges.
   The library supports all available intervals supported by the exchange API for spot and perpet-
ual futures markets.

License  GPL (>= 2)

Encoding  UTF-8

RoxygenNote  7.2.3

Suggests  knitr, quantmod, rmarkdown, testthat (>= 3.0.0)

Imports  curl (>= 5.1.0), httr (>= 1.4.7), jsonlite (>= 1.8.7),
   lifecycle, magrittr (>= 2.0.3), plotly (>= 4.10.2), rlang (>=
   1.1.1), TTR, xts (>= 0.13.1), zoo (>= 1.8-12)

Depends  R (>= 2.10)

LazyData  true

VignetteBuilder  knitr

URL  https://serkor1.github.io/cryptoQuotes/,
   https://github.com/serkor1/cryptoQuotes

BugReports  https://github.com/serkor1/cryptoQuotes/issues

NeedsCompilation  no

Author  Serkan Korkmaz [cre, aut, ctb, cph]
   (<https://orcid.org/0000-0002-5052-0982>)

Maintainer  Serkan Korkmaz <serkor1@duck.com>

Repository  CRAN

Date/Publication  2023-12-13 12:00:02 UTC
R topics documented:

\- addBBands
\- addEvents
\- addMA
\- addMACD
\- addRSI
\- addVolume
\- ATOMUSD
\- availableExchanges
\- availableIntervals
\- availableTickers
\- chart
\- getQuote
\- kline
\- ohlc

Index

---

### addBBands

*Add Bollinger Bands indicators to the charts*

#### Description

A short description...

[Experimental]

#### Usage

```
addBBands(plot, cols = c("High", "Low", "Close"), ...)
```

#### Arguments

- **plot**: a kline, or OHLC, chart.
- **cols**: a vector of column names for the Bollinger bands calculations.
- **...**: See `TTR::BBands()`

#### Value

Invisibly returns a plotly object.

#### See Also

Other chart indicators: `addEvents()`, `addMACD()`, `addMA()`, `addRSI()`, `addVolume()`
Examples

# script: scr_charting
# date: 2023-10-25
# author: Serkan Korkmaz, serkor1@duck.com
# objective: Charting in general
# script start;

# library
library(cryptoQuotes)

# charting klines
# with various indicators
chart(
  chart = kline(
    ATOMUSDT
  ) %>% addVolume() %>% addMA(
    FUN = TTR::SMA,
    n = 7
  ) %>% addMA(
    FUN = TTR::SMA,
    n = 14
  ) %>%
    addBBands() %>%
    addMACD() %>%
    addRSI()

)

# script end;

addEvents

add eventlines to the chart

Description

[Experimental]

Usage

addEvents(plot, event)

Arguments

plot          Chart
event         a data.frame with index, event and colors.

Value

Invisibly returns a plotly object.
See Also

Other chart indicators: addBBands(), addMACD(), addMA(), addRSI(), addVolume()

Examples

```r
# script: scr_addEvents
# date: 2023-12-07
# author: Serkan Korkmaz, serkor1@duck.com
# objective: Describe the usage
# of addEvents
# script start;

# load library
library(cryptoQuotes)

# 1) Generate random events
# of buys and sells and convert
# to data.frame
# Note: tibbles, data.tables are also supported
# but only base R is shown here to avoid
# too many dependencies
set.seed(1903)
event_data <- ATOMUSDT[
  sample(1:nrow(ATOMUSDT), size = 2)
]

# 1.1) Extract the index
# from the event data
index <- zoo::index(
  event_data
)

# 1.2) Convert the coredata
# into a data.frame
event_data <- as.data.frame(
  zoo::coredata(event_data)
)

# 1.3) Add the index into the data.frame
# case insensitive
event_data$index <- index

# 1.4) add events to the data.
# here we use Buys and Sells.
event_data$event <- rep(
  x = c('Buy', 'Sell'),
  length.out = nrow(event_data)
)

# 1.5) add colors based
# on the event; here buy is colored
```
# darkgrey, and if the position is closed
# with profit the color is green
event_data$color <- ifelse(
    event_data$event == 'Buy',
    yes = 'darkgrey',
    no = ifelse(
        subset(event_data, event == 'Buy')$Close < subset(event_data, event == 'Sell')$Close,
        yes = 'green',
        no = 'red'
    )
)

# 1.6) modify the event to add
# closing price at each event
event_data$event <- paste0(
    event_data$event, '@', event_data$Close
)

# 2) Chart the klines
# and add the buy and sell events
chart(
    chart = kline(
        ATOMUSDT
    ) %>% addEvents(
        event = event_data
    )
)

# script end;

---

addMA

Add various Moving Average indicators to the chart

Description

The function supports all moving averages calculated by the TTR library. Has to be explicitly called

[Experimental]

Usage

addMA(plot, FUN = TTR::SMA, ...)

Arguments

plot A plotly object of either klines or OHLC
FUN A named function calculating MAs. See TTR::SMA()
... See TTR::SMA()
Add MACD indicator to the chart

**Description**

A short description...

**[Experimental]**

**Usage**

```
addMACD(plot, ...)
```
addRSI

Arguments

plot
   A plotly object of either klines or OHLC
   ...
   See TTR::MACD()

Value

Invisbly returns a plotly object.

See Also

Other chart indicators: addBBands(), addEvents(), addMA(), addRSI(), addVolume()

Examples

# script: scr_charting
# date: 2023-10-25
# author: Serkan Korkmaz, serkor1@duck.com
# objective: Charting in general
# script start;

# library
library(cryptoQuotes)

# charting klines
# with various indicators
cart(
    chart = kline(
        ATOMUSDT
    ) %>% addVolume() %>% addMA(
        FUN = TTR::SMA, 
        n = 7
    ) %>% addMA(
        FUN = TTR::SMA, 
        n = 14
    ) %>%
        addBBands() %>%
        addMACD() %>%
        addRSI()
)

# script end;
Description

A short description...

[Experimental]

Usage

addRSI(plot, ...)

Arguments

plot A plotly object of either klines or OHLC
... See TTR::RSI()

Value

Invisibly returns a plotly object.

See Also

Other chart indicators: addBBands(), addEvents(), addMACD(), addMA(), addVolume()

Examples

# script: scr_charting
# date: 2023-10-25
# author: Serkan Korkmaz, serkor1@duck.com
# objective: Charting in general
# script start;

# library
library(cryptoQuotes)

# charting klines
# with various indicators
chart()
  chart = kline(
    ATOMUSDT
  ) %>% addVolume() %>% addMA(
    FUN = TTR::SMA,
    n = 7
  ) %>% addMA(
    FUN = TTR::SMA,
    n = 14
  ) %>%
  addBBands() %>%
  addMACD() %>%
  addMA() %>%
  addVolume()

)
addVolume

# script end;

---

addVolume Add volume indicator to the chart

Description
A short description...

[Experimental]

Usage
addVolume(plot)

Arguments
plot A plotly object of either klines or OHLC

Value
Invisibly returns a plotly object.

See Also
Other chart indicators: addBBands(), addEvents(), addMACD(), addMA(), addRSI()

Examples

# script: scr_charting
# date: 2023-10-25
# author: Serkan Korkmaz, serkor1@duck.com
# objective: Charting in general
# script start;

# library
library(cryptoQuotes)

# charting klines
# with various indicators
chart(  
  chart = kline(  
    ATOMUSDT  
  )  
)  
%>% addVolume()  
%>% addMA(  
  FUN = TTR::SMA,  
  n = 7  
)  
%>% addMA(  
  FUN = TTR::SMA,  
  n = 14  
)  
%>%
```r
availableExchanges

 addBBands() %>%
 addMACD() %>%
 addRSI()

# script end;

<table>
<thead>
<tr>
<th>ATOMUSDT</th>
<th>USDT denominated ATOMS, with 15m intervals</th>
</tr>
</thead>
</table>

**Description**

A `xts` object with 15m OHLC of USDT denominated ATOM with 97 rows and 4 columns, from 2023-01-01 to 2023-01-02:

- **Open** Opening price
- **High** Highest price
- **Low** Lowest price
- **Close** Closing price

**Usage**

ATOMUSDT

**Format**

An object of class `xts` (inherits from `zoo`) with 97 rows and 5 columns.

**Description**

This function returns all available exchanges as a message in the console.

**Usage**

`availableExchanges()`

**Value**

Invisibly returns a character vector.
availableIntervals

Examples

# script:
# date: 2023-10-06
# author: Serkan Korkmaz, serkor1@duck.com
# objective:
# script start;

## return all
## available exchanges
cryptoQuotes::availableExchanges()

# script end;

availableIntervals See all available intervals for the futures and spot markets on the desired exchange

Description

This function shows all available intervals available from each exchange

Usage

availableIntervals(source = "binance", futures = TRUE)

Arguments

source character vector of length one. Must be the name of the supported exchange
futures logical. TRUE by default. If FALSE, spot market are returned

Value

Invisibly returns a character vector.

Examples

# script:
# date: 2023-10-06
# author: Serkan Korkmaz, serkor1@duck.com
# objective:
# script start;

# available intervals
# at kucoin futures market
cryptoQuotes::availableIntervals(
  source = 'kucoin',
  futures = TRUE
)
availableTickers

Get all the available tickers on the desired exchange and market

Description
This function returns all available pairs on the exchanges.

Usage
availableTickers(source = "binance", futures = TRUE)

Arguments
source a character vector of length 1. The source of the API
futures a logical value. Default TRUE.

Value
Returns a character vector of length N equal to the tradable tickers

Examples
## available tickers
## in Binance spot market
head(try(
cryptoQuotes::availableTickers(  
  source = 'binance',
  futures = FALSE
)
))

## available tickers
## in Kraken futures market
head(try(
cryptoQuotes::availableTickers(  
  source = 'kraken',
  futures = FALSE
)
))
Description
Chart your data

Usage
chart(chart, slider = TRUE)

Arguments
chart A kline or ohlc chart with and without indicators
slider Logical. TRUE by default.

Value
Returns a plotly object

See Also
Other charting: kline(), ohlc()

Examples
# script: scr_charting
# date: 2023-10-25
# author: Serkan Korkmaz, serkor1@duck.com
# objective: Charting in general
# script start;

# library
library(cryptoQuotes)

# charting klines
# with various indicators
chart(
  chart = kline(
    ATOMUSDT
  ) %>% addVolume() %>% addMA(
    FUN = TTR::SMA,
n = 7 
) %>% addMA(
  FUN = TTR::SMA,
  n = 14
) %>%
  addBBands() %>%
  addMACD() %>%
  addRSI()

# script end;

---

getCode

*Get a quote on a cryptopair from one of the major exchanges*

**Description**

This function returns a crypto quote from one of the available exchanges. The function supports futures and spot markets.

**Usage**

```r
getCode(
  ticker,
  source = "binance",
  futures = TRUE,
  interval,
  from = NULL,
  to = NULL
)
```

**Arguments**

- **ticker**: A character vector of length 1. Uppercase.
- **source**: A character vector of length 1. See `availableExchanges()` for a full list of available exchanges.
- **futures**: A logical value. TRUE by default. If FALSE, the function will return spot prices.
- **interval**: A character vector of length 1. See `availableIntervals()` for a full list.
- **from**: A character vector of length 1. Given in %Y-%m-%d format.
- **to**: A character vector of length 1. Given in %Y-%m-%d format.

**Value**

an xts object with Open, High, Low, Close and Volume.
Examples

# 1) load perpetual futures from Binance with 15m intervals
perpAtom <- try(
  cryptoQuotes::getQuote(
    ticker = 'ATOMUSDT',
    source = 'binance',
    interval = '15m',
    futures = TRUE
  )
)

# 2) chart the futures using the chart function
if (!inherits(perpAtom, 'try-error')){
  cryptoQuotes::chart(
    chart = cryptoQuotes::kline(perpAtom) %>%
    cryptoQuotes::addVolume() %>%
    cryptoQuotes::addBBands(cols = c('Close'))
  )
}

# NOTE: Without the try the examples fails on Github Actions

# script end;

kline Create a candlestick chart

Description

This function returns a plotly kline chart with the most common indicators.

Usage

kline(quote, deficiency = FALSE, slider = TRUE)

Arguments

quote A cryptoQuote in xts/zoo format.
deficiency Logical. FALSE by default, if TRUE color deficiency compliant colors are used.
slider Logical. TRUE by default. If FALSE, no slider will be included.
Value
  Invisibly returns a plotly object.

See Also
  Other charting: `chart()`, `ohlc()`

---

### ohlc

`chart quote using ohlc bars`

---

**Description**

`chart quote using ohlc bars`

**Usage**

```r
ohlc(quote, deficiency = FALSE, slider = TRUE)
```

**Arguments**

- `quote`: A `cryptoQuote` in `xts/zoo` format.
- `deficiency`: Logical. FALSE by default, if TRUE deficiency compliant colors are used.
- `slider`: Logical. TRUE by default. If FALSE, no slider will be included.

**Value**

Invisibly returns a plotly object.

**See Also**

Other charting: `chart()`, `kline()`

**Examples**

```r
# script: scr_charting
# date: 2023-10-25
# author: Serkan Korkmaz, serkor1@duck.com
# objective: Charting in general
# script start;

# library
library(cryptoQuotes)

# charting klines
# with various indicators
chart(
  chart = kline(
    ATOMUSDT
```
ohlc

) %>% addVolume() %>% addMA(
  FUN = TTR::SMA,
  n = 7
) %>% addMA(
  FUN = TTR::SMA,
  n = 14
) %>%
  addBBands() %>%
  addMACD() %>%
  addRSI()

)

# script end;
Index

* chart indicators
  addBBands, 2
  addEvents, 3
  addMA, 5
  addMACD, 6
  addRSI, 7
  addVolume, 9
* charting
  chart, 13
  kline, 15
  ohlc, 16
* datasets
  ATOMUSDT, 10

  addBBands, 2, 4, 6–9
  addEvents, 2, 3, 6–9
  addMA, 2, 4, 5, 7–9
  addMACD, 2, 4, 6, 6, 8, 9
  addRSI, 2, 4, 6, 7, 7, 9
  addVolume, 2, 4, 6–8, 9
  ATOMUSDT, 10
  availableExchanges, 10
  availableExchanges(), 14
  availableIntervals, 11
  availableIntervals(), 14
  availableTickers, 12

  chart, 13, 16

  getQuote, 14

  kline, 13, 15, 16

  ohlc, 13, 16, 16

  TTR::BBands(), 2
  TTR::MACD(), 7
  TTR::RSI(), 8
  TTR::SMA(), 5