

Package ‘cryptoQuotes’

September 15, 2025

Title Open Access to Cryptocurrency Market Data, Sentiment Indicators
and Interactive Charts

Version 1.3.3

Description This high-level API client provides open access to cryptocurrency market data, sentiment indicators, and interactive charting tools.

The data is sourced from major cryptocurrency exchanges via 'curl' and returned in 'xts'-format. The data comes in open, high, low, and close (OHLC) format with flexible granularity, ranging from seconds to months.

This flexibility makes it ideal for developing and backtesting trading strategies or conducting detailed market analysis.

License GPL (>= 2)

Encoding UTF-8

RoxygenNote 7.3.2

Suggests data.table, knitr, quantmod, rmarkdown, testthat (>= 3.0.0),
tidyverse

Config/testthat/edition 3

Imports cli (>= 3.6.3), curl (>= 5.2.1), jsonlite (>= 1.8.8),
lifecycle (>= 1.0.4), plotly (>= 4.10.4), TTR (>= 0.24.4),
utils, xts (>= 0.14.0), zoo (>= 1.8-12)

Depends R (>= 4.0.0)

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] (ORCID:
<<https://orcid.org/0000-0002-5052-0982>>),
Jonas Cuzulan Hirani [ctb] (ORCID:
<<https://orcid.org/0000-0002-9512-1993>>)

Maintainer Serkan Korkmaz <serkor1@duck.com>

Repository CRAN

Date/Publication 2025-09-15 19:00:02 UTC

Contents

alma	3
ATOM	4
available_exchanges	5
available_intervals	6
available_tickers	8
bollinger_bands	9
BTC	10
calibrate_window	11
chart	13
dema	15
DOGE	16
donchian_channel	17
ema	18
evwma	19
fgi	21
FGIndex	22
get_fgindex	23
get_fundingrate	25
get_lsratio	27
get_mktcap	29
get_openinterest	30
get_quote	32
hma	34
kline	35
lsr	36
macd	38
ohlc	39
pline	40
remove_bound	41
rsi	43
sma	45
smi	46
split_window	48
volume	49
vwap	51
wma	52
write_xts	53
zlema	54

Index

56

alma

Add Arnaud Legoux Moving Average (ALMA) to the chart

Description

[Experimental]

A high-level `plotly::add_lines()`-wrapper function that interacts with {TTR}'s moving average family of functions. The function adds moving average indicators to the main `chart()`.

Usage

```
alma(
  price = "close",
  n      = 9,
  offset = 0.85,
  sigma  = 6,
  ...
)
```

Arguments

price	A character -vector of length 1. "close" by default The name of the vector to passed into <code>TTR::ALMA()</code> .
n	Number of periods to average over. Must be between 1 and <code>nrow(x)</code> , inclusive.
offset	Percentile at which the center of the distribution should occur.
sigma	Standard deviation of the distribution.
...	For internal use. Please ignore.

Value

A `plotly::plot_ly()`-object

Author(s)

Serkan Korkmaz

See Also

Other chart indicators: `add_event()`, `bollinger_bands()`, `chart()`, `dema()`, `donchian_channel()`, `ema()`, `evwma()`, `fgi()`, `hma()`, `lsr()`, `macd()`, `rsi()`, `sma()`, `smi()`, `volume()`, `vwap()`, `wma()`, `zlema()`

Other moving average indicators: `dema()`, `ema()`, `evwma()`, `hma()`, `sma()`, `vwap()`, `wma()`, `zlema()`

Other main chart indicators: `add_event()`, `bollinger_bands()`, `dema()`, `donchian_channel()`, `ema()`, `evwma()`, `hma()`, `sma()`, `vwap()`, `wma()`, `zlema()`

Examples

```
# script start;

cryptoQuotes::chart(
  ticker = BTC,
  main = kline(),
  indicator = list(
    cryptoQuotes::ema(n = 7),
    cryptoQuotes::sma(n = 14),
    cryptoQuotes::wma(n = 21)
  )
)

# script end;
```

ATOM

USDT Denominated ATOM (ATOMUSDT) 15-Minute Intervals

Description

This dataset contains time-series data for the ATOM (ATOM) denominated in USDT (Tether), captured in 15-minute intervals. The data spans from December 30 to December 31, 2023.

Usage

ATOM

Format

An `xts::xts()`-object with 97 rows and 5 columns,

index `<POSIXct>` The time-index

open `<numeric>` Opening price

high `<numeric>` Highest price

low `<numeric>` Lowest price

close `<numeric>` Closing price

volume `<numeric>` Trading volume

Examples

```
# Load the dataset
data("ATOM")

# chart
chart(
  ticker = ATOM,
  main = kline(),
  sub = list(volume())
)
```

available_exchanges	<i>Get available exchanges</i>
---------------------	--------------------------------

Description

[Stable]

Get a [vector](#) of all available exchanges passed into the source argument of the get-functions.

Usage

```
available_exchanges(  
  type = "ohlcv"  
)
```

Arguments

type [character](#)-vector of [length](#) 1. One of,

- "ohlcv" - Available exchanges for Open, High, Low, Close and Volume market data. See the [get_quote\(\)](#)-function.
- "lsratio" - Available exchanges for Long-Short ratios. See the [get_lsratio\(\)](#)-function.
- "fundingrate" - Available exchanges for Funding rates. See the [get_fundingrate\(\)](#)-function.
- "interest" - Available exchanges for Open interest on perpetual contracts on both sides. See the [get_openinterest\(\)](#)-function.

Details

The endpoints supported by the [available_exchanges\(\)](#) are not uniform, so exchanges available for, say, [get_lsratio\(\)](#) is not necessarily the same as those available for [get_quote\(\)](#)

Value

An [invisible\(\)](#) [character](#)-vector containing available exchanges

Author(s)

Serkan Korkmaz

See Also

Other supported calls: [available_intervals\(\)](#), [available_tickers\(\)](#)

Examples

```
# script start;

# 1) available exchanges
# on ohlc-v endpoint
cryptoQuotes::available_exchanges(
  type = "ohlc"
)

# 2) available exchanges
# on long-short ratios
cryptoQuotes::available_exchanges(
  type = "lsratio"
)

# script end;
```

available_intervals	<i>Get available intervals</i>
---------------------	--------------------------------

Description**[Stable]**

Get available intervals for the [available_tickers\(\)](#) on the [available_exchanges\(\)](#).

Usage

```
available_intervals(
  source = "binance",
  type   = "ohlc",
  futures = TRUE
)
```

Arguments

source	A character -vector of length 1. binance by default. See available_exchanges() for available exchanges.
type	character -vector of length 1. One of, "ohlc" - Available exchanges for Open, High, Low, Close and Volume market data. See the get_quote()-function. "lsratio" - Available exchanges for Long-Short ratios. See the get_lsratio()-function. "fundingrate" - Available exchanges for Funding rates. See the get_fundingrate()-function. "interest" - Available exchanges for Open interest on perpetual contracts on both sides. See the get_openinterest()-function.
futures	A logical -vector of length 1. TRUE by default. Returns futures market if TRUE , spot market otherwise.

Details

The endpoints supported by the `available_exchanges()` are not uniform, so exchanges available for, say, `get_lsratio()` is not necessarily the same as those available for `get_quote()`

Value

An `invisible()` character-vector containing the available intervals on the exchange, market and endpoint.

Sample output

```
#> i Available Intervals at "bybit" (futures):  
#> v 1m, 3m, 5m, 15m, 30m, 1h, 2h, 4h, 6h, 12h, 1d, 1M, 1w  
#> [1] "1m" "3m" "5m" "15m" "30m" "1h"
```

Author(s)

Serkan Korkmaz

See Also

Other supported calls: `available_exchanges()`, `available_tickers()`

Examples

```
## Not run:  
# script start;  
  
# available intervals  
# at kucoin futures market  
cryptoQuotes::available_intervals(  
  source = 'kucoin',  
  futures = TRUE,  
  type    = "ohlcv"  
)  
  
# available intervals  
# at kraken spot market  
cryptoQuotes::available_intervals(  
  source = 'kraken',  
  futures = FALSE,  
  type    = "ohlcv"  
)  
  
# script end;  
  
## End(Not run)
```

available_tickers	<i>Get actively traded cryptocurrency pairs</i>
-------------------	---

Description

[Stable]

Get actively traded cryptocurrency pairs on the [available_exchanges\(\)](#).

Usage

```
available_tickers(source = "binance", futures = TRUE)
```

Arguments

source	A character -vector of length 1. binance by default. See available_exchanges() for available exchanges.
futures	A logical -vector of length 1. TRUE by default. Returns futures market if TRUE , spot market otherwise.

Details

The naming-conventions across, and within, [available_exchanges\(\)](#) are not necessarily the same. This function lists all actively traded tickers.

Value

A [character](#)-vector of actively traded cryptocurrency pairs on the exchange, and the specified market.

Sample output

```
#> [1] "1000000BABYDOGEUSDT" "1000000CHEEMSUSDT" "1000000MOGUSDT"
#> [4] "1000000PEIPEIUSDT" "1000000COQUSDT" "1000000ELONUSDT"
```

Author(s)

Serkan Korkmaz

See Also

Other supported calls: [available_exchanges\(\)](#), [available_intervals\(\)](#)

Examples

```
## Not run:
# 1) available tickers
# in Binance spot market
head(
  cryptoQuotes::available_tickers(
    source = 'binance',
    futures = FALSE
  )
)

# 2) available tickers
# on Kraken futures market
head(
  cryptoQuotes::available_tickers(
    source = 'kraken',
    futures = TRUE
  )
)

## End(Not run)
```

bollinger_bands

Add Bollinger Bands to the chart

Description**[Experimental]**

A high-level `plotly::add_lines()`-wrapper function that interacts with the `TTR::BBands()`-function. The function adds bollinger bands to the main `chart()`.

Usage

```
bollinger_bands(
  n = 20,
  sd = 2,
  maType = "SMA",
  color = '#4682b4',
  ...
)
```

Arguments

n	Number of periods for moving average.
sd	The number of standard deviations to use.

maType A function or a string naming the function to be called.
color A [character](#)-vector of [length](#) 1. "#4682b4" by default.
... Other arguments to be passed to the maType function.

Value

An [invisible plotly::plot_ly\(\)](#)-object.

Author(s)

Serkan Korkmaz

See Also

Other chart indicators: [add_event\(\)](#), [alma\(\)](#), [chart\(\)](#), [dema\(\)](#), [donchian_channel\(\)](#), [ema\(\)](#), [evwma\(\)](#), [fgi\(\)](#), [hma\(\)](#), [lsr\(\)](#), [macd\(\)](#), [rsi\(\)](#), [sma\(\)](#), [smi\(\)](#), [volume\(\)](#), [vwap\(\)](#), [wma\(\)](#), [zlema\(\)](#)
Other main chart indicators: [add_event\(\)](#), [alma\(\)](#), [dema\(\)](#), [donchian_channel\(\)](#), [ema\(\)](#), [evwma\(\)](#), [hma\(\)](#), [sma\(\)](#), [vwap\(\)](#), [wma\(\)](#), [zlema\(\)](#)

Examples

```
# script start;  
  
# Charting BTC using  
# candlesticks as main  
# chart  
cryptoQuotes::chart(  
  ticker = BTC,  
  main = cryptoQuotes::kline(),  
  sub = list(  
    cryptoQuotes::volume()  
  )  
)  
  
# script end;
```

BTC	<i>USDT Denominated Bitcoin (BTCUSDT) Weekly Intervals</i>
-----	--

Description

This dataset contains time-series data for Bitcoin (BTC) denominated in USDT (Tether), captured in weekly intervals. The data spans from January 1, 2023, to December 31, 2023.

Usage

BTC

Format

An `xts::xts()`-object with 52 rows and 5 columns,

index `<POSIXct>` The time-index

open `<numeric>` Opening price

high `<numeric>` Highest price

low `<numeric>` Lowest price

close `<numeric>` Closing price

volume `<numeric>` Trading volume

Examples

```
# Load the dataset
data("BTC")

# chart
chart(
  ticker = BTC,
  main = kline(),
  sub = list(volume())
)
```

calibrate_window	<i>calibrate the time window of a list of xts objects</i>
------------------	---

Description**[Experimental]**

This function is a high-level wrapper of `do.call` and `lapply` which modifies each xts object stored in a `list()`.

Usage

```
calibrate_window(list, FUN, ...)
```

Arguments

<code>list</code>	A list of xts objects.
<code>FUN</code>	A function applied to each element of the list
<code>...</code>	optional arguments passed to FUN.

Value

Returns a xts object.

See Also

Other utility: `remove_bound()`, `split_window()`, `write_xts()`

Examples

```
# script start;

# 1) check index of BTCUSDT and
# the Fear and Greed Index
setequal(
  zoo::index(BTC),
  zoo::index(FGIndex)
)

# 2) to align the indices,
# we use the convinience functions
# by splitting the FGI by the BTC index.
FGIndex <- cryptoQuotes::split_window(
  xts = cryptoQuotes::FGIndex,
  by = zoo::index(BTC),

  # Remove upper bounds of the
  # index to avoid overlap between
  # the dates.
  #
  # This ensures that the FGI is split
  # according to start of each weekly
  # BTC candle
  bounds = 'upper'
)

# 3) as splitWindow returns a list
# it needs to be passed into calibrateWindow
# to ensure comparability
FGIndex <- cryptoQuotes::calibrate_window(
  list = FGIndex,

  # As each element in the list can include
  # more than one row, each element needs to be aggregated
  # or summarised.
  #
  # using xts::first gives the first element
  # of each list, along with its values
  FUN = xts::first
)

# 3) check if candles aligns
# accordingly
stopifnot(
  setequal(
    zoo::index(BTC),
```

```

        zoo::index(FGIndex)
    )
)

# script end;

```

chart

*Build an interactive financial chart***Description****[Experimental]**

A high-level `plotly::plot_ly()`- and `plotly::subplot()`-wrapper function for building interactive financial charts using the affiliated `chart`-functions. The `chart` consists of a main chart, and an optional subchart. The main chart supports overlaying various trading indicators like `sma` and `bollinger_bands`.

Usage

```

chart(
  ticker,
  main = kline(),
  sub = list(),
  indicator = list(),
  event_data = NULL,
  options = list()
)

```

Arguments

ticker	An object with Open, High, Low, Close and Volume columns that can be coerced to a <code>xts::xts()</code> -object.
main	A <code>plotly::plot_ly()</code> -function. <code>kline()</code> by default.
sub	An optional list of <code>plotly::plot_ly()</code> -function(s).
indicator	An optional list of <code>plotly::add_lines()</code> -function(s).
event_data	An optional <code>data.frame</code> with event line(s) to be added to the <code>chart()</code> . See <code>add_event()</code> for more details.
options	An optional list of <code>chart()</code> -options. See details below.

Details**Options:**

- `dark` A `<logical>`-value of `length` 1. `TRUE` by default. Sets the overall theme of the `chart()`
- `slider` A `<logical>`-value of `length` 1. `FALSE` by default. If `TRUE`, a `plotly::rangeslider()` is added.

- deficiency A `<logical>`-value of `length` 1. `FALSE` by default. If `TRUE`, all `chart()`-elements are colorblind friendly
- size A `<numeric>`-value of `length` 1. The relative size of the main chart. 0.6 by default. Must be between 0 and 1, non-inclusive.
- scale A `<numeric>`-value of `length` 1. 1 by default. Scales all fonts on the chart.
- width A `<numeric>`-value of `length` 1. 0.9 by default. Sets the width of all line elements on the chart.
- static A `<logical>`-value of `length` 1. `FALSE` by default. If `FALSE` the chart can be edited, annotated and explored interactively.
- palette A `<character>`-vector of `length` 1. "hawaii" by default. See `hcl.pals()` for all possible color palettes.

Charting Events:

If `event_data` is passed, vertical eventlines with appropriate labels and coloring are added to the `chart()`. This function is rigid, as it will fail if event, label and index columns are not passed.

For more details please see `add_event()`.

Value

A `plotly::plot_ly()` object.

Author(s)

Serkan Korkmaz

See Also

Other chart indicators: `add_event()`, `alma()`, `bollinger_bands()`, `dema()`, `donchian_channel()`, `ema()`, `evwma()`, `fgi()`, `hma()`, `lsr()`, `macd()`, `rsi()`, `sma()`, `smi()`, `volume()`, `vwap()`, `wma()`, `zlema()`

Other price charts: `kline()`, `ohlc()`, `pline()`

Examples

```
# script start;

# 1) charting weekly
# BTC using candlesticks
# and indicators
cryptoQuotes::chart(
  ticker = BTC,
  main = cryptoQuotes::kline(),
  sub = list(
    cryptoQuotes::volume(),
    cryptoQuotes::macd()
  ),
  indicator = list(
    cryptoQuotes::bollinger_bands(),
    cryptoQuotes::sma(),
    cryptoQuotes::alma()
  )
)
```

```

    ),
    options = list(
      dark = TRUE,
      deficiency = FALSE
    )
  )

# script end;

```

dema

Add Double Exponential Moving Average (DEMA) to the chart

Description

[Experimental]

A high-level `plotly::add_lines()`-wrapper function that interacts with {TTR}'s moving average family of functions. The function adds moving average indicators to the main `chart()`.

Usage

```

dema(
  price = "close",
  n      = 10,
  v      = 1,
  wilder = FALSE,
  ratio  = NULL,
  ...
)

```

Arguments

price	A character -vector of length 1. "close" by default. The name of the vector to passed into <code>TTR::DEMA()</code> .
n	Number of periods to average over. Must be between 1 and <code>nrow(x)</code> , inclusive.
v	The 'volume factor' (a number in [0,1]). See Notes.
wilder	logical; if TRUE, a Welles Wilder type EMA will be calculated; see notes.
ratio	A smoothing/decay ratio. ratio overrides wilder in EMA.
...	For internal use. Please ignore.

Value

A `plotly::plot_ly()`-object

Author(s)

Serkan Korkmaz

See Also

Other chart indicators: `add_event()`, `alma()`, `bollinger_bands()`, `chart()`, `donchian_channel()`, `ema()`, `evwma()`, `fgi()`, `hma()`, `lsr()`, `macd()`, `rsi()`, `sma()`, `smi()`, `volume()`, `vwap()`, `wma()`, `zlema()`

Other moving average indicators: `alma()`, `ema()`, `evwma()`, `hma()`, `sma()`, `vwap()`, `wma()`, `zlema()`

Other main chart indicators: `add_event()`, `alma()`, `bollinger_bands()`, `donchian_channel()`, `ema()`, `evwma()`, `hma()`, `sma()`, `vwap()`, `wma()`, `zlema()`

Examples

```
# script start;

cryptoQuotes::chart(
  ticker = BTC,
  main = kline(),
  indicator = list(
    cryptoQuotes::ema(n = 7),
    cryptoQuotes::sma(n = 14),
    cryptoQuotes::wma(n = 21)
  )
)

# script end;
```

DOGE

*USDT Denominated DOGECOIN (DOGEUSDT) 1-Minute Intervals***Description**

This dataset contains time-series data for the DOGECOIN (DOGE) denominated in USDT (Tether), captured in 1-minute intervals. The data spans 2022-01-14 07:00:00 CET to 2022-01-14 08:00:00 CET.

Usage

DOGE

Format

An `xts::xts()`-object with 61 rows and 5 columns,

index `<POSIXct>` The time-index

open `<numeric>` Opening price

high `<numeric>` Highest price

low `<numeric>` Lowest price

close `<numeric>` Closing price

volume `<numeric>` Trading volume

Examples

```
# Load the dataset
data("DOGE")

# chart
chart(
  ticker = DOGE,
  main = kline(),
  sub = list(volume())
)
```

donchian_channel

Add Donchian Channels to the chart

Description**[Experimental]**

A high-level `plotly::add_lines()`-wrapper function that interacts with the `TTR::DonchianChannel()`-function. The function adds Donchian Channels to the main `chart()`.

Usage

```
donchian_channel(
  n = 10,
  include.lag = FALSE,
  color = '#4682b4',
  ...
)
```

Arguments

<code>n</code>	Number of periods for moving average.
<code>include.lag</code>	Should values be lagged so that today's prices are not included in the calculation? See Note.
<code>color</code>	A character -vector of length 1. "#4682b4" by default.
<code>...</code>	For internal use. Please ignore.

Value

An [invisible](#) `plotly::plot_ly()`-object.

Author(s)

Serkan Korkmaz

See Also

Other chart indicators: `add_event()`, `alma()`, `bollinger_bands()`, `chart()`, `dema()`, `ema()`, `evwma()`, `fgi()`, `hma()`, `lsr()`, `macd()`, `rsi()`, `sma()`, `smi()`, `volume()`, `vwap()`, `wma()`, `zlema()`

Other main chart indicators: `add_event()`, `alma()`, `bollinger_bands()`, `dema()`, `ema()`, `evwma()`, `hma()`, `sma()`, `vwap()`, `wma()`, `zlema()`

Examples

```
# script start;

# Charting BTC using
# candlesticks as main
# chart
cryptoQuotes::chart(
  ticker = BTC,
  main = cryptoQuotes::kline(),
  sub = list(
    cryptoQuotes::volume()
  )
)

# script end;
```

ema

*Add Exponentially-Weighted Moving Average (EMA) to the chart***Description****[Experimental]**

A high-level `plotly::add_lines()`-wrapper function that interacts with `{TTR}`'s moving average family of functions. The function adds moving average indicators to the main `chart()`.

Usage

```
ema(
  price = "close",
  n      = 10,
  wilder = FALSE,
  ratio  = NULL,
  ...
)
```

Arguments

price	A character -vector of length 1. "close" by default. The name of the vector to passed into <code>TTR::EMA()</code> .
n	Number of periods to average over. Must be between 1 and <code>nrow(x)</code> , inclusive.

wilder	logical; if TRUE, a Welles Wilder type EMA will be calculated; see notes.
ratio	A smoothing/decay ratio. ratio overrides wilder in EMA.
...	For internal use. Please ignore.

Value

A `plotly::plot_ly()`-object

Author(s)

Serkan Korkmaz

See Also

Other chart indicators: `add_event()`, `alma()`, `bollinger_bands()`, `chart()`, `dema()`, `donchian_channel()`, `evwma()`, `fgi()`, `hma()`, `lsr()`, `macd()`, `rsi()`, `sma()`, `smi()`, `volume()`, `vwap()`, `wma()`, `zlema()`

Other moving average indicators: `alma()`, `dema()`, `evwma()`, `hma()`, `sma()`, `vwap()`, `wma()`, `zlema()`

Other main chart indicators: `add_event()`, `alma()`, `bollinger_bands()`, `dema()`, `donchian_channel()`, `evwma()`, `hma()`, `sma()`, `vwap()`, `wma()`, `zlema()`

Examples

```
# script start;

cryptoQuotes::chart(
  ticker = BTC,
  main = kline(),
  indicator = list(
    cryptoQuotes::ema(n = 7),
    cryptoQuotes::sma(n = 14),
    cryptoQuotes::wma(n = 21)
  )
)

# script end;
```

evwma

Add Elastic Volume-Weighted Moving Average (EVWMA) to the chart

Description**[Experimental]**

A high-level `plotly::add_lines()`-wrapper function that interacts with {TTR}'s moving average family of functions. The function adds moving average indicators to the main `chart()`.

Usage

```
evwma(
  price = "close",
  n     = 10,
  ...
)
```

Arguments

price	A character -vector of length 1. "close" by default. The name of the vector to passed into TTR::EVWMA()
n	Number of periods to average over. Must be between 1 and <code>nrow(x)</code> , inclusive.
...	For internal use. Please ignore.

Value

A [plotly::plot_ly\(\)](#)-object

Author(s)

Serkan Korkmaz

See Also

Other chart indicators: [add_event\(\)](#), [alma\(\)](#), [bollinger_bands\(\)](#), [chart\(\)](#), [dema\(\)](#), [donchian_channel\(\)](#), [ema\(\)](#), [fgi\(\)](#), [hma\(\)](#), [lsr\(\)](#), [macd\(\)](#), [rsi\(\)](#), [sma\(\)](#), [smi\(\)](#), [volume\(\)](#), [vwap\(\)](#), [wma\(\)](#), [zlema\(\)](#)

Other moving average indicators: [alma\(\)](#), [dema\(\)](#), [ema\(\)](#), [hma\(\)](#), [sma\(\)](#), [vwap\(\)](#), [wma\(\)](#), [zlema\(\)](#)

Other main chart indicators: [add_event\(\)](#), [alma\(\)](#), [bollinger_bands\(\)](#), [dema\(\)](#), [donchian_channel\(\)](#), [ema\(\)](#), [hma\(\)](#), [sma\(\)](#), [vwap\(\)](#), [wma\(\)](#), [zlema\(\)](#)

Examples

```
# script start;

cryptoQuotes::chart(
  ticker = BTC,
  main = kline(),
  indicator = list(
    cryptoQuotes::ema(n = 7),
    cryptoQuotes::sma(n = 14),
    cryptoQuotes::wma(n = 21)
  )
)

# script end;
```

fgi*Chart the Fear and Greed Index*

Description

[Experimental]

A high-level `plotly::plot_ly()`-wrapper function. The function adds a subchart with the fear and greed-index.

Usage

```
fgi(index, ...)
```

Arguments

index	A <code>xts::xts()</code> -object. See <code>get_fgindex()</code> for more details.
...	For internal use. Please ignore.

Details

Classification:

The Fear and Greed Index goes from 0-100, and can be classified as follows,

- 0-24, Extreme Fear
- 25-44, Fear
- 45-55, Neutral
- 56-75, Greed
- 76-100, Extreme Greed

About the Fear and Greed Index:

The fear and greed index is a market sentiment indicator that measures investor emotions to gauge whether they are generally fearful (indicating potential selling pressure) or greedy (indicating potential buying enthusiasm).

Source:

This index is fetched from [alternative.me](#), and can be different from the one provided by [coinmarketcap](#).

Value

An `invisible plotly::plot_ly()`-object.

Author(s)

Serkan Korkmaz

See Also

Other chart indicators: `add_event()`, `alma()`, `bollinger_bands()`, `chart()`, `dema()`, `donchian_channel()`, `ema()`, `evwma()`, `hma()`, `lsr()`, `macd()`, `rsi()`, `sma()`, `smi()`, `volume()`, `vwap()`, `wma()`, `zlema()`

Other sentiment indicators: `lsr()`

Other subchart indicators: `add_event()`, `lsr()`, `macd()`, `rsi()`, `smi()`, `volume()`

Examples

```
## Not run:
# script start;

# 1) get the fear and greed index
# for the last 14 days
FGIndex <- cryptoQuotes::get_fgindex(
  from = Sys.Date() - 14
)

# 2) get the BTC price
# for the last 14 days
BTC <- cryptoQuotes::get_quote(
  ticker = "BTCUSDT",
  source = "bybit",
  futures = FALSE,
  from = Sys.Date() - 14
)

# 3) chart the daily BTC
# along side the Fear and
# Greed Index
cryptoQuotes::chart(
  ticker = BTC,
  main = kline(),
  sub = list(
    fgi(
      FGIndex
    )
  )
)

# script end;

## End(Not run)
```

FGIndex

Fear and Greed Index (FGI) values for the cryptocurrency market in daily intervals

Description

This dataset contains daily values of the Fear and Greed Index for the year 2023, which is used to measure the sentiments of investors in the market. The data spans from January 1, 2023, to December 31, 2023.

Usage

FGIndex

Format

An `xts::xts()`-object with 364 rows and 1 columns,

index `<POSIXct>` The time-index

fgi `<numeric>` The daily fear and greed index value

Details

The Fear and Greed Index goes from 0-100, and can be classified as follows,

- 0-24, Extreme Fear
- 25-44, Fear
- 45-55, Neutral
- 56-75, Greed
- 76-100, Extreme Greed

Examples

```
# Load the dataset
data("FGIndex")

# Get a summary of index values
summary(FGIndex)
```

get_fgindex

Get the daily Fear and Greed Index in the cryptocurrency market

Description

[Stable]

Get the daily fear and greed index.

Usage

```
get_fgindex(
  from = NULL,
  to   = NULL
)
```

Arguments

from An optional [character](#)-, [date](#)- or [POSIXct](#)-vector of [length](#) 1. [NULL](#) by default.

to An optional [character](#)-, [date](#)- or [POSIXct](#)-vector of [length](#) 1. [NULL](#) by default.

Details**Classification:**

The Fear and Greed Index goes from 0-100, and can be classified as follows,

- 0-24, Extreme Fear
- 25-44, Fear
- 45-55, Neutral
- 56-75, Greed
- 76-100, Extreme Greed

About the Fear and Greed Index:

The fear and greed index is a market sentiment indicator that measures investor emotions to gauge whether they are generally fearful (indicating potential selling pressure) or greedy (indicating potential buying enthusiasm).

Source:

This index is fetched from [alternative.me](#), and can be different from the one provided by [coinmarketcap](#).

Value

An [<\[xts\]>](#)-object containing,

index [<POSIXct>](#) the time-index

fgi [<numeric>](#) the daily fear and greed index value

Sample output

```
#>                                fgi
#> 2024-05-12 02:00:00    56
#> 2024-05-13 02:00:00    57
#> 2024-05-14 02:00:00    66
#> 2024-05-15 02:00:00    64
#> 2024-05-16 02:00:00    70
#> 2024-05-17 02:00:00    74
```

Author(s)

Serkan Korkmaz

See Also

Other get-functions: [get_fundingrate\(\)](#), [get_lsratio\(\)](#), [get_openinterest\(\)](#), [get_quote\(\)](#)

Examples

```
## Not run:
# script start;

# 1) get the fear and greed index
# for the last 7 days
tail(
  fgi <- cryptoQuotes::get_fgindex(
    from = Sys.Date() - 7
  )
)

# script end;

## End(Not run)
```

get_fundingrate	<i>Get the funding rate on futures contracts</i>
-----------------	--

Description**[Stable]**

Get the funding rate on a cryptocurrency pair from the [available_exchanges\(\)](#) in any actively traded [available_tickers\(\)](#) on the futures markets.

Usage

```
get_fundingrate(
  ticker,
  source = 'binance',
  from = NULL,
  to = NULL
)
```

Arguments

ticker	A character -vector of length 1. See available_tickers() for available tickers.
source	A character -vector of length 1. binance by default. See available_exchanges() for available exchanges.
from	An optional character -, date - or POSIXct -vector of length 1. NULL by default.
to	An optional character -, date - or POSIXct -vector of length 1. NULL by default.

Value

An `<[xts]>`-object containing,

index `<POSIXct>` the time-index
 funding_rate `<numeric>` the current funding rate

Sample output

```
#>                                funding_rate
#> 2024-03-09 17:00:00    0.00026407
#> 2024-03-10 01:00:00    0.00031010
#> 2024-03-10 09:00:00    0.00063451
#> 2024-03-10 17:00:00    0.00054479
#> 2024-03-11 01:00:00    0.00035489
#> 2024-03-11 09:00:00    0.00078428
```

Author(s)

Serkan Korkmaz

See Also

Other get-functions: `get_fgindex()`, `get_lsratio()`, `get_openinterest()`, `get_quote()`

Examples

```
## Not run:
# script start;

# 1) check available
# exchanges for funding rates
cryptoQuotes::available_exchanges(
  type = "fundingrate"
)

# 2) get BTC funding rate
# for the last 7 days
tail(
  BTC <- cryptoQuotes::get_fundingrate(
    ticker = "BTCUSDT",
    source = "binance",
    from   = Sys.Date() - 7
  )
)

# script end;

## End(Not run)
```

get_lsratio

Get the long to short ratio of a cryptocurrency pair

Description

[Stable]

Get the long-short ratio for any [available_tickers\(\)](#) from the [available_exchanges\(\)](#)

Usage

```
get_lsratio(
  ticker,
  interval = '1d',
  source   = 'binance',
  from     = NULL,
  to       = NULL,
  top      = FALSE
)
```

Arguments

ticker	A character -vector of length 1. See available_tickers() for available tickers.
interval	A character -vector of length 1. 1d by default. See available_intervals() for available intervals.
source	A character -vector of length 1. binance by default. See available_exchanges() for available exchanges.
from	An optional character -, date - or POSIXct -vector of length 1. NULL by default.
to	An optional character -, date - or POSIXct -vector of length 1. NULL by default.
top	A logical vector. FALSE by default. If TRUE it returns the top traders Long-Short ratios.

Details

On time-zones and dates:

Values passed to `from`` or `to` must be coercible by `[as.Date()]`, or `[as.POSIXct()]`, with a format of either "`%m-%d`" or "`%Y-%m-%d %H:%M:%S`". By default all dates are passed and returned with [Sys.timezone\(\)](#).

On returns:

If only `from` is provided 200 pips are returned up to `Sys.time()`. If only `to` is provided 200 pips up to the specified date is returned.

Value

An `xts::xts`-object containing,

index	<POSIXct> the time-index
long	<numeric> the share of longs
short	<numeric> the share of shorts
ls_ratio	<numeric> the ratio of longs to shorts

Sample output

```
#>               long short ls_ratio
#> 2024-05-12 02:00:00 0.6930 0.3070 2.2573290
#> 2024-05-13 02:00:00 0.6637 0.3363 1.9735355
#> 2024-05-14 02:00:00 0.5555 0.4445 1.2497188
#> 2024-05-15 02:00:00 0.6580 0.3420 1.9239766
#> 2024-05-16 02:00:00 0.4868 0.5132 0.9485581
#> 2024-05-17 02:00:00 0.5102 0.4898 1.0416497
```

Author(s)

Jonas Cuzulan Hirani

See Also

Other get-functions: [get_fgindex\(\)](#), [get_fundingrate\(\)](#), [get_openinterest\(\)](#), [get_quote\(\)](#)

Examples

```
## Not run:
# script start;

LS_BTC <- cryptoQuotes::get_lsratio(
  ticker   = 'BTCUSDT',
  interval = '15m',
  from     = Sys.Date() - 1,
  to       = Sys.Date()
)

# end of script;

## End(Not run)
```

get_mktcap

*Get the global market capitalization***Description**

The `get_mktcap()`-functions returns the global cryptocurrency market capitalization.

Usage

```
get_mktcap(
  interval = "1d",
  from = NULL,
  to = NULL,
  altcoin = FALSE,
  reported = FALSE
)
```

Arguments

interval	A character -vector of length 1. 1d by default. See available_intervals() for available intervals.
from	An optional character -, date - or POSIXct -vector of length 1. NULL by default.
to	An optional character -, date - or POSIXct -vector of length 1. NULL by default.
altcoin	A logical -vector of length 1. FALSE by default. Returns altcoin market capitalization if TRUE
reported	A logical -vector of length 1. FALSE by default. Returns reported volume if TRUE .

Details**On time-zones and dates:**

Values passed to `from`` or `to` must be coercible by `[as.Date()]`, or `[as.POSIXct()]`, with a format of either "`%m-%d`" or "`%Y-%m-%d %H:%M:%S`". By default all dates are passed and returned with [Sys.timezone\(\)](#).

On returns:

If only `from` is provided 200 pips are returned up to `Sys.time()`. If only `to` is provided 200 pips up to the specified date is returned.

Value

An [<\[xts\]>](#)-object containing,

index	<POSIXct> The time-index
marketcap	<numeric> Market capitalization
volume	<numeric> Trading volume

Author(s)

Serkan Korkmaz

See AlsoOther get-functions: [get_fgindex\(\)](#), [get_fundingrate\(\)](#), [get_lsratio\(\)](#), [get_openinterest\(\)](#)**Examples**

```
## Not run:
# script start;

# get quote on
# BTCUSDT pair from
# Binance in 30m
# intervals from the
# last 24 hours
tail(
  BTC <- cryptoQuotes::get_quote(
    ticker  = 'BTCUSDT',
    source  = 'binance',
    interval = '30m',
    futures = FALSE,
    from    = Sys.Date() - 1
  )
)

# script end;

## End(Not run)
```

get_openinterest

*Get the open interest on perpetual futures contracts***Description****[Stable]**

Get the open interest on a cryptocurrency pair from the [available_exchanges\(\)](#) in any actively traded [available_tickers\(\)](#) on the FUTURES markets.

Usage

```
get_openinterest(
  ticker,
  interval = '1d',
  source   = 'binance',
  from     = NULL,
  to       = NULL
)
```

Arguments

ticker	A character -vector of length 1. See available_tickers() for available tickers.
interval	A character -vector of length 1. 1d by default. See available_intervals() for available intervals.
source	A character -vector of length 1. binance by default. See available_exchanges() for available exchanges.
from	An optional character -, date - or POSIXct -vector of length 1. NULL by default.
to	An optional character -, date - or POSIXct -vector of length 1. NULL by default.

Details**On time-zones and dates:**

Values passed to `from`` or `to` must be coercible by `[as.Date()]`, or `[as.POSIXct()]`, with a format of either "`%m-%d`" or "`%Y-%m-%d %H:%M:%S`". By default all dates are passed and returned with [Sys.timezone\(\)](#).

On returns:

If only `from` is provided 200 pips are returned up to `Sys.time()`. If only `to` is provided 200 pips up to the specified date is returned.

Value

An [<\[xts\]>](#)-object containing,

index	<POSIXct> the time-index
open_interest	<numeric> open perpetual contracts on both both sides

Sample output

```
#>                                open_interest
#> 2024-05-12 02:00:00          70961.07
#> 2024-05-13 02:00:00          69740.49
#> 2024-05-14 02:00:00          71110.33
#> 2024-05-15 02:00:00          67758.06
#> 2024-05-16 02:00:00          73614.70
#> 2024-05-17 02:00:00          72377.85
```

Author(s)

Serkan Korkmaz

See Also

Other get-functions: [get_fgindex\(\)](#), [get_fundingrate\(\)](#), [get_lsratio\(\)](#), [get_quote\(\)](#)

Examples

```
## Not run:
# script start;

# 1) check available
# exchanges for open interest
cryptoQuotes::available_exchanges(
  type = 'interest'
)

# 2) get BTC funding rate
# for the last 7 days
tail(
  BTC <- cryptoQuotes::get_openinterest(
    ticker = "BTCUSDT",
    source = "binance",
    from   = Sys.Date() - 7
  )
)

# script end;

## End(Not run)
```

get_quote

Get the Open, High, Low, Close and Volume data on a cryptocurrency pair

Description

[Stable]

Get a quote on a cryptocurrency pair from the [available_exchanges\(\)](#) in various [available_intervals\(\)](#) for any actively traded [available_tickers\(\)](#).

Usage

```
get_quote(
  ticker,
  source   = 'binance',
  futures  = TRUE,
  interval = '1d',
  from     = NULL,
  to       = NULL
)
```

Arguments

ticker	A character -vector of length 1. See available_tickers() for available tickers.
source	A character -vector of length 1. binance by default. See available_exchanges() for available exchanges.
futures	A logical -vector of length 1. TRUE by default. Returns futures market if TRUE , spot market otherwise.
interval	A character -vector of length 1. 1d by default. See available_intervals() for available intervals.
from	An optional character -, date - or POSIXct -vector of length 1. NULL by default.
to	An optional character -, date - or POSIXct -vector of length 1. NULL by default.

Details**On time-zones and dates:**

Values passed to from or to must be coercible by `[as.Date()]`, or `[as.POSIXct()]`, with a format of either `%m-%d` or `%Y-%m-%d %H:%M:%S`. By default all dates are passed and returned with [Sys.timezone\(\)](#).

On returns:

If only from is provided 200 pips are returned up to `Sys.time()`. If only to is provided 200 pips up to the specified date is returned.

Value

An [<xts>](#)-object containing,

index	<POSIXct> The time-index
open	<numeric> Opening price
high	<numeric> Highest price
low	<numeric> Lowest price
close	<numeric> Closing price
volume	<numeric> Trading volume

Sample output

```
#>               open    high    low  close  volume
#> 2024-05-12 02:00:00 60809.2 61849.4 60557.3 61455.8 104043.9
#> 2024-05-13 02:00:00 61455.7 63440.0 60750.0 62912.1 261927.1
#> 2024-05-14 02:00:00 62912.2 63099.6 60950.0 61550.5 244345.3
#> 2024-05-15 02:00:00 61550.5 66440.0 61316.1 66175.4 365031.7
#> 2024-05-16 02:00:00 66175.4 66800.0 64567.0 65217.7 242455.3
#> 2024-05-17 02:00:00 65217.7 66478.5 65061.2 66218.8 66139.1
```

Author(s)

Serkan Korkmaz

See Also

Other get-functions: [get_fgindex\(\)](#), [get_fundingrate\(\)](#), [get_lsratio\(\)](#), [get_openinterest\(\)](#)

Examples

```
## Not run:
# script start;

# get quote on
# BTCUSDT pair from
# Binance in 30m
# intervals from the
# last 24 hours
tail(
  BTC <- cryptoQuotes::get_quote(
    ticker = 'BTCUSDT',
    source = 'binance',
    interval = '30m',
    futures = FALSE,
    from = Sys.Date() - 1
  )
)

# script end;

## End(Not run)
```

hma

Add Hull Moving Average (HMA) to the chart

Description**[Experimental]**

A high-level [plotly::add_lines\(\)](#)-wrapper function that interacts with {TTR}'s moving average family of functions. The function adds moving average indicators to the main [chart\(\)](#).

Usage

```
hma(
  price = "close",
  n = 20,
  ...
)
```

Arguments

price	A character -vector of length 1. "close" by default. The name of the vector to passed into TTR::HMA() .
n	Number of periods to average over. Must be between 1 and <code>nrow(x)</code> , inclusive.
...	For internal use. Please ignore.

Value

A [plotly::plot_ly\(\)](#)-object

Author(s)

Serkan Korkmaz

See Also

Other chart indicators: [add_event\(\)](#), [alma\(\)](#), [bollinger_bands\(\)](#), [chart\(\)](#), [dema\(\)](#), [donchian_channel\(\)](#), [ema\(\)](#), [evwma\(\)](#), [fgi\(\)](#), [lsr\(\)](#), [macd\(\)](#), [rsi\(\)](#), [sma\(\)](#), [smi\(\)](#), [volume\(\)](#), [vwap\(\)](#), [wma\(\)](#), [zlema\(\)](#)

Other moving average indicators: [alma\(\)](#), [dema\(\)](#), [ema\(\)](#), [evwma\(\)](#), [sma\(\)](#), [vwap\(\)](#), [wma\(\)](#), [zlema\(\)](#)

Other main chart indicators: [add_event\(\)](#), [alma\(\)](#), [bollinger_bands\(\)](#), [dema\(\)](#), [donchian_channel\(\)](#), [ema\(\)](#), [evwma\(\)](#), [sma\(\)](#), [vwap\(\)](#), [wma\(\)](#), [zlema\(\)](#)

Examples

```
# script start;

cryptoQuotes::chart(
  ticker = BTC,
  main = kline(),
  indicator = list(
    cryptoQuotes::ema(n = 7),
    cryptoQuotes::sma(n = 14),
    cryptoQuotes::wma(n = 21)
  )
)

# script end;
```

kline

Candlestick Chart

Description**[Experimental]**

A high-level [plotly::plot_ly\(\)](#)-function for charting Open, High, Low and Close prices.

Usage

```
kline(...)
```

Arguments

```
...
```

 For internal use. Please ignore.**Value**

An [invisible plotly::plot_ly\(\)](#)-object.

Author(s)

Serkan Korkmaz

See Also

Other price charts: [chart\(\)](#), [ohlcv\(\)](#), [pline\(\)](#)

Examples

```
# script start;

# Charting BTC using
# candlesticks as main
# chart
cryptoQuotes::chart(
  ticker = BTC,
  main = cryptoQuotes::kline(),
  sub = list(
    cryptoQuotes::volume()
  )
)

# script end;
```

lsr

Chart the long-short ratio

Description**[Experimental]**

A high-level [plotly::plot_ly\(\)](#)-wrapper function. The function adds a subchart to the [chart](#) with long-short ratio.

Usage

```
lsr(ratio, ...)
```

Arguments

ratio A `xts::xts()`-object. See `get_lsratio()` for more details.
 ... For internal use. Please ignore.

Value

An `invisible plotly::plot_ly()`-object.

Author(s)

Serkan Korkmaz

See Also

Other chart indicators: `add_event()`, `alma()`, `bollinger_bands()`, `chart()`, `dema()`, `donchian_channel()`, `ema()`, `evwma()`, `fgi()`, `hma()`, `macd()`, `rsi()`, `sma()`, `smi()`, `volume()`, `vwap()`, `wma()`, `zlema()`

Other sentiment indicators: `fgi()`

Other subchart indicators: `add_event()`, `fgi()`, `macd()`, `rsi()`, `smi()`, `volume()`

Examples

```
## Not run:
# script start;

# 1) long-short ratio
# on BTCUSDT pair
LS_BTC <- cryptoQuotes::get_lsratio(
  ticker = 'BTCUSDT',
  interval = '15m',
  from = Sys.Date() - 1,
  to = Sys.Date()
)

# 2) BTCSDT in same period
# as the long-short ratio;
BTC <- cryptoQuotes::get_quote(
  ticker = 'BTCUSDT',
  futures = TRUE,
  interval = '15m',
  from = Sys.Date() - 1,
  to = Sys.Date()
)

# 3) plot BTCUSDT-pair
# with long-short ratio
cryptoQuotes::chart(
  ticker = BTC,
  main = cryptoQuotes::kline(),
  sub = list(
    cryptoQuotes::lsr(
```

```
        ratio = LS_BTC
      )
    )
  )

  # end of scrip;

## End(Not run)
```

macd	<i>Chart the Moving Average Convergence Divergence (MACD) indicator</i>
------	---

Description

[Experimental]

A high-level `plotly::plot_ly()`- and `plotly::add_lines()`-function that interacts with the `TTR::MACD()`-function. The function adds subchart with a `TTR::MACD()`-indicator.

Usage

```
macd(
  nFast   = 12,
  nSlow   = 26,
  nSig    = 9,
  maType  = "SMA",
  percent = TRUE,
  ...
)
```

Arguments

nFast	Number of periods for fast moving average.
nSlow	Number of periods for slow moving average.
nSig	Number of periods for signal moving average.
maType	Either: 1. A function or a string naming the function to be called. 2. A <i>list</i> with the first component like (1) above, and additional parameters specified as <i>named</i> components. See Examples.
percent	logical; if TRUE, the percentage difference between the fast and slow moving averages is returned, otherwise the difference between the respective averages is returned.
...	For internal use. Please ignore.

Value

An `invisible plotly::plot_ly()`-object.

Author(s)

Serkan Korkmaz

See Also

Other chart indicators: `add_event()`, `alma()`, `bollinger_bands()`, `chart()`, `dema()`, `donchian_channel()`, `ema()`, `evwma()`, `fgi()`, `hma()`, `lsr()`, `rsi()`, `sma()`, `smi()`, `volume()`, `vwap()`, `wma()`, `zlema()`

Other subchart indicators: `add_event()`, `fgi()`, `lsr()`, `rsi()`, `smi()`, `volume()`

Other momentum indicators: `rsi()`, `smi()`

Examples

```
# script start;

# 1) charting weekly
# BTC using candlesticks
# and indicators
cryptoQuotes::chart(
  ticker = BTC,
  main = cryptoQuotes::kline(),
  sub = list(
    cryptoQuotes::volume(),
    cryptoQuotes::macd()
  ),
  indicator = list(
    cryptoQuotes::bollinger_bands(),
    cryptoQuotes::sma(),
    cryptoQuotes::alma()
  ),
  options = list(
    dark = TRUE,
    deficiency = FALSE
  )
)

# script end;
```

ohlc

OHLC Barchart

Description**[Experimental]**

A high-level `plotly::plot_ly()`-function for charting Open, High, Low and Close prices.

Usage

```
ohlc(...)
```

Arguments

```
... For internal use. Please ignore.
```

Value

An [invisible plotly::plot_ly\(\)](#)-object.

Author(s)

Serkan Korkmaz

See Also

Other price charts: [chart\(\)](#), [kline\(\)](#), [pline\(\)](#)

Examples

```
# script start;

# Charting BTC using
# OHLC-bars as main
# chart
cryptoQuotes::chart(
  ticker = BTC,
  main = cryptoQuotes::ohlc(),
  sub = list(
    cryptoQuotes::volume()
  )
)

# script end;
```

pline

Line Chart

Description**[Experimental]**

A high-level [plotly::plot_ly\(\)](#)-function for charting Open, High, Low and Close prices.

Usage

```
pline(price = "close", ...)
```

Arguments

price A [character](#)-vector of [length](#) 1. "close" by default.
 ... For internal use. Please ignore.

Value

An [invisible plotly::plot_ly\(\)](#)-object.

Author(s)

Serkan Korkmaz

See Also

Other price charts: [chart\(\)](#), [kline\(\)](#), [ohlc\(\)](#)

Examples

```
# script start;

# Charting BTC using
# line charts with closing price
# as main chart
cryptoQuotes::chart(
  ticker = BTC,
  main = cryptoQuotes::pline(),
  sub = list(
    cryptoQuotes::volume()
  )
)

# script end;
```

remove_bound

remove upper and lower bounds from an XTS object

Description**[Experimental]**

The [stats::window\(\)](#)-function has inclusive upper and lower bounds, which in some cases is an undesirable feature. This high level function removes the bounds if desired

Usage

```
remove_bound(xts, bounds = c("upper"))
```

Arguments

<code>xts</code>	A xts-object that needs its bounds modified.
<code>bounds</code>	A character vector of length 1. Has to be one of <code>c('upper', 'lower', 'both')</code> . Defaults to <code>Upper</code> .

Value

Returns an xts-class object with its bounds removed.

See Also

Other utility: `calibrate_window()`, `split_window()`, `write_xts()`

Examples

```
# script start;

# 1) check index of BTCUSDT and
# the Fear and Greed Index
setequal(
  zoo::index(BTC),
  zoo::index(FGIndex)
)

# 2) to align the indices,
# we use the convinience functions
# by splitting the FGI by the BTC index.
FGIndex <- cryptoQuotes::split_window(
  xts = cryptoQuotes::FGIndex,
  by = zoo::index(BTC),

  # Remove upper bounds of the
  # index to avoid overlap between
  # the dates.
  #
  # This ensures that the FGI is split
  # according to start of each weekly
  # BTC candle
  bounds = 'upper'
)

# 3) as splitWindow returns a list
# it needs to passed into calibrateWindow
# to ensure comparability
FGIndex <- cryptoQuotes::calibrate_window(
  list = FGIndex,

  # As each element in the list can include
  # more than one row, each element needs to be aggregated
  # or summarised.
  #
```

```

# using xts::first gives the first element
# of each list, along with its values
FUN = xts::first
)

# 3) check if candles aligns
# accordingly
stopifnot(
  setequal(
    zoo::index(BTC),
    zoo::index(FGIndex)
  )
)

# script end;

```

rsi

Chart the Relative Strength Index (RSI)

Description

[Experimental]

A high-level `plotly::plot_ly()`- and `plotly::add_lines()`-function that interacts with the `TTR::RSI()`-function. The function adds a subchart with a `TTR::RSI()`-indicator.

Usage

```

rsi(
  price      = "close",
  n          = 14,
  maType     = "SMA",
  upper_limit = 80,
  lower_limit = 20,
  color      = '#4682b4',
  ...
)

```

Arguments

price	Price series that is coercible to xts or matrix.
n	Number of periods for moving averages.
maType	Either: 1. A function or a string naming the function to be called. 2. A <i>list</i> with the first component like (1) above, and additional parameters specified as <i>named</i> components. See Examples.
upper_limit	A numeric -vector of length 1. 80 by default. Sets the upper limit of the <code>TTR::RSI</code> .

lower_limit A [numeric](#)-vector of [length](#) 1. 20 by default. Sets the lower limit of the [TTR::RSI](#).
 color A [character](#)-vector of [length](#) 1. "#4682b4" by default.
 ... For internal use. Please ignore.

Value

An [invisible plotly::plot_ly\(\)](#)-object.

Author(s)

Serkan Korkmaz

See Also

Other chart indicators: [add_event\(\)](#), [alma\(\)](#), [bollinger_bands\(\)](#), [chart\(\)](#), [dema\(\)](#), [donchian_channel\(\)](#), [ema\(\)](#), [evwma\(\)](#), [fgi\(\)](#), [hma\(\)](#), [lsr\(\)](#), [macd\(\)](#), [sma\(\)](#), [smi\(\)](#), [volume\(\)](#), [vwap\(\)](#), [wma\(\)](#), [zlema\(\)](#)

Other subchart indicators: [add_event\(\)](#), [fgi\(\)](#), [lsr\(\)](#), [macd\(\)](#), [smi\(\)](#), [volume\(\)](#)

Other momentum indicators: [macd\(\)](#), [smi\(\)](#)

Examples

```
# script start;

# 1) charting weekly
# BTC using candlesticks
# and indicators
cryptoQuotes::chart(
  ticker = BTC,
  main = cryptoQuotes::kline(),
  sub = list(
    cryptoQuotes::volume(),
    cryptoQuotes::macd()
  ),
  indicator = list(
    cryptoQuotes::bollinger_bands(),
    cryptoQuotes::sma(),
    cryptoQuotes::alma()
  ),
  options = list(
    dark = TRUE,
    deficiency = FALSE
  )
)

# script end;
```

sma

Add Simple Moving Average (SMA) indicators to the chart

Description

[Experimental]

A high-level `plotly::add_lines()`-wrapper function that interacts with `{TTR}`'s moving average family of functions. The function adds moving average indicators to the main `chart()`.

Usage

```
sma(
  price = "close",
  n     = 10,
  ...
)
```

Arguments

price	A character -vector of length 1. "close" by default. The name of the vector to passed into <code>TTR::SMA()</code> .
n	Number of periods to average over. Must be between 1 and <code>nrow(x)</code> , inclusive.
...	For internal use. Please ignore.

Value

A `plotly::plot_ly()`-object

Author(s)

Serkan Korkmaz

See Also

Other chart indicators: `add_event()`, `alma()`, `bollinger_bands()`, `chart()`, `dema()`, `donchian_channel()`, `ema()`, `evwma()`, `fgi()`, `hma()`, `lsr()`, `macd()`, `rsi()`, `smi()`, `volume()`, `vwap()`, `wma()`, `zlema()`

Other moving average indicators: `alma()`, `dema()`, `ema()`, `evwma()`, `hma()`, `vwap()`, `wma()`, `zlema()`

Other main chart indicators: `add_event()`, `alma()`, `bollinger_bands()`, `dema()`, `donchian_channel()`, `ema()`, `evwma()`, `hma()`, `vwap()`, `wma()`, `zlema()`

Examples

```
# script start;

cryptoQuotes::chart(
  ticker = BTC,
  main = kline(),
  indicator = list(
    cryptoQuotes::ema(n = 7),
    cryptoQuotes::sma(n = 14),
    cryptoQuotes::wma(n = 21)
  )
)

# script end;
```

smi

Chart the Stochastic Momentum Index (SMI)

Description

[Experimental]

A high-level `plotly::plot_ly()`- and `plotly::add_lines()`-function that interacts with the `TTR::SMI()`-function. The function adds a subchart with a `TTR::SMI()`-indicator.

Usage

```
smi(
  nFastK = 14,
  nFastD = 3,
  nSlowD = 3,
  maType,
  bounded = TRUE,
  smooth = 1,
  upper_limit = 40,
  lower_limit = -40,
  color = "#4682b4",
  ...
)
```

Arguments

<code>nFastK</code>	Number of periods for fast %K (i.e. the number of past periods to use).
<code>nFastD</code>	Number of periods for fast %D (i.e. the number smoothing periods to apply to fast %K).
<code>nSlowD</code>	Number of periods for slow %D (i.e. the number smoothing periods to apply to fast %D).
<code>maType</code>	Either:

	1. A function or a string naming the function to be called. 2. A <i>list</i> with the first component like (1) above, and additional parameters specified as <i>named</i> components. See Examples.
bounded	Logical, should current period's values be used in the calculation?
smooth	Number of internal smoothing periods to be applied before calculating FastK. See Details.
upper_limit	A numeric -vector of length 1. 40 by default. Sets the upper limit of the TTR::SMI .
lower_limit	A numeric -vector of length 1. -40 by default. Sets the lower limit of the TTR::SMI .
color	A character -vector of length 1. "#4682b4" by default.
...	For internal use. Please ignore.

Value

An **invisible plotly::plot_ly()**-object.

Author(s)

Serkan Korkmaz

See Also

Other chart indicators: **add_event()**, **alma()**, **bollinger_bands()**, **chart()**, **dema()**, **donchian_channel()**, **ema()**, **evwma()**, **fgi()**, **hma()**, **lsr()**, **macd()**, **rsi()**, **sma()**, **volume()**, **vwap()**, **wma()**, **zlema()**

Other subchart indicators: **add_event()**, **fgi()**, **lsr()**, **macd()**, **rsi()**, **volume()**

Other momentum indicators: **macd()**, **rsi()**

Examples

```
# script start;

# 1) charting weekly
# BTC using candlesticks
# and indicators
cryptoQuotes::chart(
  ticker = BTC,
  main = cryptoQuotes::kline(),
  sub = list(
    cryptoQuotes::volume(),
    cryptoQuotes::macd()
  ),
  indicator = list(
    cryptoQuotes::bollinger_bands(),
    cryptoQuotes::sma(),
    cryptoQuotes::alma()
  ),
  options = list(
    dark = TRUE,
```

```

        deficiency = FALSE
    )
)

# script end;

```

split_window

split xts object iteratively in lists of desired intervals

Description

[Experimental]

The `split_window()`-function is a high level wrapper of the `stats::window()`-function which restricts the intervals between the first and second index value iteratively

Usage

```
split_window(xts, by, bounds = "upper")
```

Arguments

xts	A xts-object that needs needs to be split.
by	A reference <code>zoo::index()</code> -object, to be split by.
bounds	A character vector of length 1. Has to be one of <code>c('upper', 'lower', 'both')</code> . Defaults to Upper.

Value

Returns a list of iteratively restricted xts objects

See Also

Other utility: `calibrate_window()`, `remove_bound()`, `write_xts()`

Examples

```

# script start;

# 1) check index of BTCUSDT and
# the Fear and Greed Index
setequal(
  zoo::index(BTC),
  zoo::index(FGIndex)
)

# 2) to align the indices,
# we use the convinience functions
# by splitting the FGI by the BTC index.

```

```

FGIndex <- cryptoQuotes::split_window(
  xts = cryptoQuotes::FGIndex,
  by = zoo::index(BTC),

  # Remove upper bounds of the
  # index to avoid overlap between
  # the dates.
  #
  # This ensures that the FGI is split
  # according to start of each weekly
  # BTC candle
  bounds = 'upper'
)

# 3) as splitWindow returns a list
# it needs to be passed into calibrateWindow
# to ensure comparability
FGIndex <- cryptoQuotes::calibrate_window(
  list = FGIndex,

  # As each element in the list can include
  # more than one row, each element needs to be aggregated
  # or summarised.
  #
  # using xts::first gives the first element
  # of each list, along with its values
  FUN = xts::first
)

# 3) check if candles aligns
# accordingly
stopifnot(
  setequal(
    zoo::index(BTC),
    zoo::index(FGIndex)
  )
)

# script end;

```

volume

Chart the trading volume

Description

[Experimental]

A high-level `plotly::plot_ly()`-function. The function adds a subchart with the trading trading.

Usage

```
volume(...)
```

Arguments

```
...           For internal use. Please ignore.
```

Value

An [invisible plotly::plot_ly\(\)](#)-object.

Author(s)

Serkan Korkmaz

See Also

Other chart indicators: [add_event\(\)](#), [alma\(\)](#), [bollinger_bands\(\)](#), [chart\(\)](#), [dema\(\)](#), [donchian_channel\(\)](#), [ema\(\)](#), [evwma\(\)](#), [fgi\(\)](#), [hma\(\)](#), [lsr\(\)](#), [macd\(\)](#), [rsi\(\)](#), [sma\(\)](#), [smi\(\)](#), [vwap\(\)](#), [wma\(\)](#), [zlema\(\)](#)

Other subchart indicators: [add_event\(\)](#), [fgi\(\)](#), [lsr\(\)](#), [macd\(\)](#), [rsi\(\)](#), [smi\(\)](#)

Examples

```
# script start;

# 1) charting weekly
# BTC using candlesticks
# and indicators
cryptoQuotes::chart(
  ticker = BTC,
  main = cryptoQuotes::kline(),
  sub = list(
    cryptoQuotes::volume(),
    cryptoQuotes::macd()
  ),
  indicator = list(
    cryptoQuotes::bollinger_bands(),
    cryptoQuotes::sma(),
    cryptoQuotes::alma()
  ),
  options = list(
    dark = TRUE,
    deficiency = FALSE
  )
)

# script end;
```

vwap

*Add Volume-Weighted Moving Average (VWAP) to the chart***Description****[Experimental]**

A high-level `plotly::add_lines()`-wrapper function that interacts with {TTR}'s moving average family of functions. The function adds moving average indicators to the main `chart()`.

Usage

```
vwap(
  price = "close",
  n      = 10,
  ratio = NULL,
  ...
)
```

Arguments

<code>price</code>	A character -vector of length 1. "close" by default. The name of the vector to passed into <code>TTR::VWAP()</code>
<code>n</code>	Number of periods to average over. Must be between 1 and <code>nrow(x)</code> , inclusive.
<code>ratio</code>	A smoothing/decay ratio. <code>ratio</code> overrides <code>wilder</code> in EMA.
<code>...</code>	For internal use. Please ignore.

Value

A `plotly::plot_ly()`-object

Author(s)

Serkan Korkmaz

See Also

Other chart indicators: `add_event()`, `alma()`, `bollinger_bands()`, `chart()`, `dema()`, `donchian_channel()`, `ema()`, `evwma()`, `fgi()`, `hma()`, `lsr()`, `macd()`, `rsi()`, `sma()`, `smi()`, `volume()`, `wma()`, `zlema()`

Other moving average indicators: `alma()`, `dema()`, `ema()`, `evwma()`, `hma()`, `sma()`, `wma()`, `zlema()`

Other main chart indicators: `add_event()`, `alma()`, `bollinger_bands()`, `dema()`, `donchian_channel()`, `ema()`, `evwma()`, `hma()`, `sma()`, `wma()`, `zlema()`

Examples

```
# script start;

cryptoQuotes::chart(
  ticker = BTC,
  main = kline(),
  indicator = list(
    cryptoQuotes::ema(n = 7),
    cryptoQuotes::sma(n = 14),
    cryptoQuotes::wma(n = 21)
  )
)

# script end;
```

wma

Add Weighted Moving Average (WMA) to the chart

Description

[Experimental]

A high-level `plotly::add_lines()`-wrapper function that interacts with {TTR}'s moving average family of functions. The function adds moving average indicators to the main `chart()`.

Usage

```
wma(
  price = "close",
  n      = 10,
  wts    = 1:n,
  ...
)
```

Arguments

price	A character -vector of length 1. "close" by default. The name of the vector to passed into <code>TTR::WMA()</code> .
n	Number of periods to average over. Must be between 1 and <code>nrow(x)</code> , inclusive.
wts	Vector of weights. Length of wts vector must equal the length of x, or n (the default).
...	For internal use. Please ignore.

Value

A `plotly::plot_ly()`-object

Author(s)

Serkan Korkmaz

See Also

Other chart indicators: [add_event\(\)](#), [alma\(\)](#), [bollinger_bands\(\)](#), [chart\(\)](#), [dema\(\)](#), [donchian_channel\(\)](#), [ema\(\)](#), [evwma\(\)](#), [fgi\(\)](#), [hma\(\)](#), [lsr\(\)](#), [macd\(\)](#), [rsi\(\)](#), [sma\(\)](#), [smi\(\)](#), [volume\(\)](#), [vwap\(\)](#), [zlema\(\)](#)

Other moving average indicators: [alma\(\)](#), [dema\(\)](#), [ema\(\)](#), [evwma\(\)](#), [hma\(\)](#), [sma\(\)](#), [vwap\(\)](#), [zlema\(\)](#)

Other main chart indicators: [add_event\(\)](#), [alma\(\)](#), [bollinger_bands\(\)](#), [dema\(\)](#), [donchian_channel\(\)](#), [ema\(\)](#), [evwma\(\)](#), [hma\(\)](#), [sma\(\)](#), [vwap\(\)](#), [zlema\(\)](#)

Examples

```
# script start;

cryptoQuotes::chart(
  ticker = BTC,
  main = kline(),
  indicator = list(
    cryptoQuotes::ema(n = 7),
    cryptoQuotes::sma(n = 14),
    cryptoQuotes::wma(n = 21)
  )
)

# script end;
```

write_xts

Read and Write xts-objects

Description**[Experimental]**

The [write_xts\(\)](#)- and [read_xts\(\)](#)-functions are [zoo::write.zoo\(\)](#)- and [zoo::read.zoo\(\)](#)-wrapper functions.

Usage

```
# write XTS-object
write_xts(
  x,
  file,
  ...
)

# read XTS-object
read_xts(
  file
)
```

Arguments

<code>x</code>	An <code><[xts]></code> -object.
<code>file</code>	character string or strings giving the name of the file(s) which the data are to be read from/written to. See read.table and write.table for more information. Alternatively, in <code>read.zoo</code> , <code>file</code> can be a <code>connection</code> or a <code>data.frame</code> (e.g., resulting from a previous <code>read.table</code> call) that is subsequently processed to a "zoo" series.
<code>...</code>	further arguments passed to other functions. In the <code>read.*.zoo</code> the arguments are passed to the function specified in <code>read</code> (unless <code>file</code> is a <code>data.frame</code> already). In <code>write.zoo</code> the arguments are passed to write.table .

Details

When reading and writing `<[xts]>`-objects the [attributes](#) does not follow the object.

Author(s)

Serkan Korkmaz

See Also

Other utility: [calibrate_window\(\)](#), [remove_bound\(\)](#), [split_window\(\)](#)

Other utility: [calibrate_window\(\)](#), [remove_bound\(\)](#), [split_window\(\)](#)

zlema

Add Zero Lag Exponential Moving Average (ZLEMA) to the chart

Description

[Experimental]

A high-level [plotly::add_lines\(\)](#)-wrapper function that interacts with {TTR}'s moving average family of functions. The function adds moving average indicators to the main [chart\(\)](#).

Usage

```
zlema(
  price = "close",
  n      = 10,
  ratio = NULL,
  ...
)
```

Arguments

price	A character -vector of length 1. "close" by default. The name of the vector to passed into TTR::ZLEMA() .
n	Number of periods to average over. Must be between 1 and <code>nrow(x)</code> , inclusive.
ratio	A smoothing/decay ratio. ratio overrides wilder in EMA.
...	For internal use. Please ignore.

Value

A [plotly::plot_ly\(\)](#)-object

Author(s)

Serkan Korkmaz

See Also

Other chart indicators: [add_event\(\)](#), [alma\(\)](#), [bollinger_bands\(\)](#), [chart\(\)](#), [dema\(\)](#), [donchian_channel\(\)](#), [ema\(\)](#), [evwma\(\)](#), [fgi\(\)](#), [hma\(\)](#), [lsr\(\)](#), [macd\(\)](#), [rsi\(\)](#), [sma\(\)](#), [smi\(\)](#), [volume\(\)](#), [vwap\(\)](#), [wma\(\)](#)

Other moving average indicators: [alma\(\)](#), [dema\(\)](#), [ema\(\)](#), [evwma\(\)](#), [hma\(\)](#), [sma\(\)](#), [vwap\(\)](#), [wma\(\)](#)

Other main chart indicators: [add_event\(\)](#), [alma\(\)](#), [bollinger_bands\(\)](#), [dema\(\)](#), [donchian_channel\(\)](#), [ema\(\)](#), [evwma\(\)](#), [hma\(\)](#), [sma\(\)](#), [vwap\(\)](#), [wma\(\)](#)

Examples

```
# script start;

cryptoQuotes::chart(
  ticker = BTC,
  main = kline(),
  indicator = list(
    cryptoQuotes::ema(n = 7),
    cryptoQuotes::sma(n = 14),
    cryptoQuotes::wma(n = 21)
  )
)

# script end;
```

Index

* chart indicators

- alma, [3](#)
- bollinger_bands, [9](#)
- chart, [13](#)
- dema, [15](#)
- donchian_channel, [17](#)
- ema, [18](#)
- evwma, [19](#)
- fgi, [21](#)
- hma, [34](#)
- lsr, [36](#)
- macd, [38](#)
- rsi, [43](#)
- sma, [45](#)
- smi, [46](#)
- volume, [49](#)
- vwap, [51](#)
- wma, [52](#)
- zlema, [54](#)

* datasets

- ATOM, [4](#)
- BTC, [10](#)
- DOGE, [16](#)
- FGIndex, [22](#)

* get-functions

- get_fgindex, [23](#)
- get_fundingrate, [25](#)
- get_lsratio, [27](#)
- get_openinterest, [30](#)
- get_quote, [32](#)

* main chart indicators

- alma, [3](#)
- bollinger_bands, [9](#)
- dema, [15](#)
- donchian_channel, [17](#)
- ema, [18](#)
- evwma, [19](#)
- hma, [34](#)
- sma, [45](#)

- vwap, [51](#)
- wma, [52](#)
- zlema, [54](#)

* momentum indicators

- macd, [38](#)
- rsi, [43](#)
- smi, [46](#)

* moving average indicators

- alma, [3](#)
- dema, [15](#)
- ema, [18](#)
- evwma, [19](#)
- hma, [34](#)
- sma, [45](#)
- vwap, [51](#)
- wma, [52](#)
- zlema, [54](#)

* price charts

- chart, [13](#)
- kline, [35](#)
- ohlc, [39](#)
- pline, [40](#)

* sentiment indicators

- fgi, [21](#)
- lsr, [36](#)

* subchart indicators

- fgi, [21](#)
- lsr, [36](#)
- macd, [38](#)
- rsi, [43](#)
- smi, [46](#)
- volume, [49](#)

* supported calls

- available_exchanges, [5](#)
- available_intervals, [6](#)
- available_tickers, [8](#)

* utility

- calibrate_window, [11](#)
- remove_bound, [41](#)

- split_window, 48
- write_xts, 53
- add_event, 3, 10, 14, 16, 18–20, 22, 35, 37, 39, 44, 45, 47, 50, 51, 53, 55
- add_event(), 13, 14
- alma, 3, 10, 14, 16, 18–20, 22, 35, 37, 39, 44, 45, 47, 50, 51, 53, 55
- ATOM, 4
- attributes, 54
- available_exchanges, 5, 7, 8
- available_exchanges(), 5–8, 25, 27, 30–33
- available_intervals, 5, 6, 8
- available_intervals(), 27, 29, 31–33
- available_tickers, 5, 7, 8
- available_tickers(), 6, 25, 27, 30–33
- bollinger_bands, 3, 9, 13, 14, 16, 18–20, 22, 35, 37, 39, 44, 45, 47, 50, 51, 53, 55
- BTC, 10
- calibrate_window, 11, 42, 48, 54
- character, 3, 5–8, 10, 14, 15, 17, 18, 20, 24, 25, 27, 29, 31, 33, 35, 41, 44, 45, 47, 51, 52, 55
- chart, 3, 10, 13, 13, 16, 18–20, 22, 35–37, 39–41, 44, 45, 47, 50, 51, 53, 55
- chart(), 3, 9, 13–15, 17–19, 34, 45, 51, 52, 54
- data.frame, 13
- date, 24, 25, 27, 29, 31, 33
- dema, 3, 10, 14, 15, 18–20, 22, 35, 37, 39, 44, 45, 47, 50, 51, 53, 55
- do.call, 11
- DOGE, 16
- donchian_channel, 3, 10, 14, 16, 17, 19, 20, 22, 35, 37, 39, 44, 45, 47, 50, 51, 53, 55
- ema, 3, 10, 14, 16, 18, 18, 20, 22, 35, 37, 39, 44, 45, 47, 50, 51, 53, 55
- ewwma, 3, 10, 14, 16, 18, 19, 19, 22, 35, 37, 39, 44, 45, 47, 50, 51, 53, 55
- FALSE, 13, 14, 27, 29
- fgi, 3, 10, 14, 16, 18–20, 21, 35, 37, 39, 44, 45, 47, 50, 51, 53, 55
- FGIndex, 22
- get_fgindex, 23, 26, 28, 30, 31, 34
- get_fgindex(), 21
- get_fundingrate, 24, 25, 28, 30, 31, 34
- get_fundingrate(), 5, 6
- get_lsratio, 24, 26, 27, 30, 31, 34
- get_lsratio(), 5–7, 37
- get_mktcap, 29
- get_mktcap(), 29
- get_openinterest, 24, 26, 28, 30, 30, 34
- get_openinterest(), 5, 6
- get_quote, 24, 26, 28, 31, 32
- get_quote(), 5–7
- hcl.pals(), 14
- hma, 3, 10, 14, 16, 18–20, 22, 34, 37, 39, 44, 45, 47, 50, 51, 53, 55
- invisible, 10, 17, 21, 36, 37, 39–41, 44, 47, 50
- invisible(), 5, 7
- kline, 14, 35, 40, 41
- kline(), 13
- lapply, 11
- length, 3, 5, 6, 8, 10, 13–15, 17, 18, 20, 24, 25, 27, 29, 31, 33, 35, 41, 43–45, 47, 51, 52, 55
- list, 13
- list(), 11
- logical, 6, 8, 13, 14, 27, 29, 33
- lsr, 3, 10, 14, 16, 18–20, 22, 35, 36, 39, 44, 45, 47, 50, 51, 53, 55
- macd, 3, 10, 14, 16, 18–20, 22, 35, 37, 38, 44, 45, 47, 50, 51, 53, 55
- NULL, 24, 25, 27, 29, 31, 33
- numeric, 4, 11, 14, 16, 23, 24, 26, 28, 29, 31, 33, 43, 44, 47
- ohlc, 14, 36, 39, 41
- pline, 14, 36, 40, 40
- plotly::add_lines(), 3, 9, 13, 15, 17–19, 34, 38, 43, 45, 46, 51, 52, 54
- plotly::plot_ly(), 3, 10, 13–15, 17, 19–21, 35–41, 43–47, 49–52, 55
- plotly::rangeslider(), 13
- plotly::subplot(), 13
- POSIXct, 4, 11, 16, 23–29, 31, 33

`read.table`, 54
`read_xts(write_xts)`, 53
`read_xts()`, 53
`remove_bound`, 12, 41, 48, 54
`rsi`, 3, 10, 14, 16, 18–20, 22, 35, 37, 39, 43, 45, 47, 50, 51, 53, 55

`sma`, 3, 10, 13, 14, 16, 18–20, 22, 35, 37, 39, 44, 45, 47, 50, 51, 53, 55
`smi`, 3, 10, 14, 16, 18–20, 22, 35, 37, 39, 44, 45, 46, 50, 51, 53, 55
`split_window`, 12, 42, 48, 54
`split_window()`, 48
`stats::window()`, 41, 48
`Sys.timezone()`, 27, 29, 31, 33

`TRUE`, 6, 8, 13, 14, 27, 29, 33
`TTR::ALMA()`, 3
`TTR::BBands()`, 9
`TTR::DEMA()`, 15
`TTR::DonchianChannel()`, 17
`TTR::EMA()`, 18
`TTR::EVWMA()`, 20
`TTR::HMA()`, 35
`TTR::MACD()`, 38
`TTR::RSI`, 43, 44
`TTR::RSI()`, 43
`TTR::SMA()`, 45
`TTR::SMI`, 47
`TTR::SMI()`, 46
`TTR::VWAP()`, 51
`TTR::WMA()`, 52
`TTR::ZLEMA()`, 55

`vector`, 5
`volume`, 3, 10, 14, 16, 18–20, 22, 35, 37, 39, 44, 45, 47, 49, 51, 53, 55
`vwap`, 3, 10, 14, 16, 18–20, 22, 35, 37, 39, 44, 45, 47, 50, 51, 53, 55

`wma`, 3, 10, 14, 16, 18–20, 22, 35, 37, 39, 44, 45, 47, 50, 51, 52, 55
`write.table`, 54
`write_xts`, 12, 42, 48, 53
`write_xts()`, 53

`xts`, 24, 26, 29, 31, 33, 54
`xts::xts`, 28
`xts::xts()`, 4, 11, 13, 16, 21, 23, 37

`zlema`, 3, 10, 14, 16, 18–20, 22, 35, 37, 39, 44, 45, 47, 50, 51, 53, 54
`zoo::index()`, 48
`zoo::read.zoo()`, 53
`zoo::write.zoo()`, 53