Package ‘TwitterAutomatedTrading’

October 12, 2022

Type Package
Title Automated Trading Using Tweets
Version 0.1.0
Author Lucas Godeiro
Maintainer Lucas Godeiro <lucas.godeiro@hotmail.com>

Description Provides an integration to the 'metatrader 5'.
The functionalities carry out automated trading using
sentiment indexes computed from 'twitter' and/or 'stockwits'.
The sentiment indexes are based on the ph.d. dissertation
The integration between the 'R' and the 'metatrader 5' allows sending buy/sell orders to the bro-
kerage.

License GPL-3
Encoding UTF-8
LazyData true
Depends R (>= 3.1.0)
RoxygenNote 7.1.0

URL https://github.com/lucasgodeiro/TwitterAutomatedTrading

BugReports https://github.com/lucasgodeiro/TwitterAutomatedTrading/issues
Suggests knitr, rmarkdown, covr
VignetteBuilder knitr
Imports curl, dplyr, jsonlite, lubridate, plyr, purrr, tibble,
twitteR, naptime, utils, tidytext, magrittr

NeedsCompilation no
Repository CRAN
Date/Publication 2020-05-31 09:50:13 UTC
check_frequency function

Description
This function checks if the EA can send order to the platform trading.

Usage
check_frequency(hours_frequency, time_zone)

Arguments
hours_frequency
The vector containing the hours of operations.

time_zone
The time zone.

Value
A logical vector TRUE if the EA can compute the sentiment.

Examples

time_zone <- "Brazil/East"
hour_freq <- generate_trade_frequency(9,17,10)
check_freq <- check_frequency(hours_frequency = hour_freq,
    time_zone = time_zone)
**Close_Position**

**Description**
This function closes an open position.

**Usage**
```r
Close_Position(actual_decision)
```

**Arguments**
- `actual_decision`
  The current position status("BUY IT NOW", "SELL IT NOW", "SELL IT NOW CLOSE", "BUY IT NOW CLOSE").

**Value**
A vector with the new decision.

**Examples**
```r
decision <- "SELL IT NOW"
decision <- Close_Position(actual_decision = decision)
```

---

**generate_trade_frequency**

**Description**
`generate_trade_frequency` function

**Usage**
```r
generate_trade_frequency(initial_time, final_time, freq_trade)
```

**Arguments**
- `initial_time`
The time the algorithm starts trading.
- `final_time`
The time the algorithm ends trading.
- `freq_trade`
The frequency which the algorithm recalculates the sentiment index.
get_sentiment_stocktwits

Value

A vector containing the hours of operation.

Examples

hours_candle_10 <- generate_trade_frequency(9,17,10)
# For example, for 17:30, you should use minutes/60, i.e. 17.5
hours_candle_20 <- generate_trade_frequency(9,17.5,10)

get_sentiment_stocktwits

Description

This function computes the sentiment based on bullish and bearish tag from stocktwits using the last 30 twits.

Usage

get_sentiment_stocktwits(stock_symbol, path_twits, sentiment_index_type)

Arguments

stock_symbol A vector with the stocks symbols.
path_twits The path where the Json files will be stored.
sentiment_index_type

The sentiment type to be used according to the dictionary, positive, negative or both. Default is both, positive and negative.

Value

A numeric value with the value of the sentiment index.

Examples

## Not run:
# Not run:
path_twits <- 'your path'
symbols <- c("EWZ", "SPX", "SPY", "USO")

stocktwits_index <- get_sentiment_stocktwits(stock_symbol = symbols,
path_twits = path_twits)

## End(Not run)
get_sentiment_tweets

Description

This function computes the sentiment from tweets. Remind to connect with twitter using your API Key.

Usage

get_sentiment_tweets(
  ntweets,  # Number of tweets to be searched
  time_tweet,  # Time in hours where the tweets will be filtered
  terms_list,  # Terms to be searched
  time_zone,  # The time zone
  positive_dictionary,  # The list of positive terms of the dictionary
  negative_dictionary,  # The list of negative terms of the dictionary
  sentiment_index_type  # The sentiment type to be used according to the dictionary, positive, negative or both. Default is both, positive and negative
)

Arguments

- `ntweets`: Number of tweets to be searched.
- `time_tweet`: Time in hours where the tweets will be filtered.
- `terms_list`: Terms to be searched.
- `time_zone`: The time zone.
- `positive_dictionary`: The list of positive terms of the dictionary.
- `negative_dictionary`: The list of negative terms of the dictionary.
- `sentiment_index_type`: The sentiment type to be used according to the dictionary, positive, negative or both. Default is both, positive and negative.

Value

A list with: (1) - the sentiment index, (2) a tibble with the words counting, (3) a tibble with the negative words counting and (4)

Examples

```r
## Not run:
# Not run:
ntweets <- 500
time_tweet <- 6
terms_list <- c("IBOVESPA OR bovespa OR ibov OR petroleo OR $SPX OR $SPY OR $EWZ")
```


```r
# Having IP

# Having IP Function

Description

Function to test if the internet connection is available

Usage

```
havingIP(operational_system)
```

Arguments

```
operational_system

The operational system.
```

Value

A logical vector TRUE if internet connection is available.

Examples

```
## Not run:
internet <- havingIP()

## End(Not run)
```
**my_dictionary**

**Description**

A simple list containing a dictionary with positive and negative words (English and Portuguese).

**Usage**

my_dictionary

**Format**

A list with 2 components.

- **positive_terms** The positive words.
- **negative_terms** The negative words.

---

**operation_hours**

**Description**

This function defines the operations hours of the EA.

**Usage**

operation_hours(start_time, end_time, time_zone)

**Arguments**

- **start_time** The time that the EA should start to trade.
- **end_time** The time that the EA should stop to trade and close the open positions.
- **time_zone** The time zone.

**Value**

A logical variable TRUE if the Expert Advisor can trade.
Examples

```r
time_zone <- "Brazil/East"
op_hours<- operation_hours(start_time = 9.5,
end_time = 17,
time_zone = time_zone)
```

Description

This function starts the Algorithm and sends the orders to txt file that will be read for the Expert Advisor in the Metatrader 5.

Usage

```r
Start_Trading(
consumer_key,
consumer_secret,
access_token,
access_secret,
path_decision,
ntweets,
terms_list,
time_tweet,
time_zone,
positive_dictionary,
negative_dictionary,
stock_symbol,
path_twits,
Operation_Hours1,
Operation_Hours2,
Operation_Hours3,
start_time1,
start_time2,
start_time3,
end_time1,
end_time2,
end_time3,
Day_Trade,
nap_time_error,
initial_time,
```
final_time,
freq_trade,
w_twitter,
w_stocktwits,
Sentiment_Index_Threshold,
Use_Delta_Sentiment,
Signal_File_Name
)

Arguments

consumer_key Api Twitter Consumer Key
consumer_secret Api Twitter Consumer Secret
access_token Api Twitter access token
access_secret Api Twitter access secret
path_decision The path where the txt file with the decision will be saved. Generally it is saved in the 'Common' file at Metaquotes folder (see vignette for instructions).
ntweets see get_sentiment_tweets.
terms_list see get_sentiment_tweets.
time_tweet see get_sentiment_tweets.
time_zone see get_sentiment_tweets.
positive_dictionary see get_sentiment_tweets.
negative_dictionary see get_sentiment_tweets.
stock_symbol see get_sentiment_Stocktwits.
path_twits see get_sentiment_Stocktwits.
Operation_Hours1 The operation hours 1 for day trade. TRUE or FALSE.
Operation_Hours2 The operation hours 2 for day trade. TRUE or FALSE.
Operation_Hours3 The operation hours 3 for day trade. TRUE or FALSE.
start_time1 The start time 1 for day trade.
start_time2 The start time 2 for day trade.
start_time3 The start time 3 for day trade.
end_time1 The end time 1 for day trade.
end_time2 The end time 2 for day trade.
end_time3 The end time 3 for day trade.
nap_time_error The time that the EA should take a nap in case of error.
initial_time The start of operation.
final_time The time which the position in day trade mode must be closed.
freq_trade The time in minutes the EA must recompute the sentiment index and take a decision.
w_twitter The weight of the twitter sentiment index.
w_stocktwits The weight of the stocktwits sentiment index.
Sentiment_Index_Threshold see trade_decision function.
Use_Delta_Sentiment see trade_decision function
Signal_File_Name The Signal File Name.

Value
The functions just activate the algorithm.

Examples
```r
## Not run:
Not run:
Signal_File_Name <- 'Signal.txt'
ntweets <- 5000
time_tweet <- 6
terms_list <- c("IBOVESPA OR bovespa OR ibov OR petroleo OR $SPX OR $SPY OR $EWZ")
time_zone <- "Brazil/East"
positive_dictionary <- my_dictionary[['positive_terms']] negative_dictionary <- my_dictionary[['negative_terms']] path_twits <- 'your path'
stock_symbol <- c("EWZ", "SPX", "SPY", "USO")
time_zone <- "Brazil/East"

consumer_key <- "your consumer_key"
consumer_secret <- "your consumer_secret"
access_token <- "your access token"
access_secret <- " your access secret "
ap_time_error <- 7.7
path_decision <- 'metatrader txt file path'
path_twits <- 'your path'
initial_time <- 9
final_time <- 17
freq_trade <- 10
Day_Trade <- TRUE
Operation_Hours1 <- TRUE
start_time1 <- 9
end_time1 <- 17
w_twitter <- 0.9
w_stocktwits <- 0.1
Sentiment_Index_Threshold <- 0.5
```
Trade_Decision

Start_Trading(consumer_key = consumer_key, 
consumer_secret = consumer_secret, 
access_token = access_token, 
access_secret = access_secret, 
path_decision = path_decision, 
ntweets = ntweets, 
terms_list = terms_list, 
time_tweet = time_tweet, 
time_zone = time_zone, 
positive_dictionary = positive_dictionary, 
negative_dictionary = negative_dictionary, 
stock_symbol = stock_symbol, 
path_twits = path_twits, 
Operation_Hours1 = TRUE, 
Operation_Hours2 = FALSE, 
Operation_Hours3 = FALSE, 
start_time1 = start_time1, 
start_time2 = start_time1, 
start_time3 = start_time1, 
end_time1 = end_time1, 
end_time2 = end_time1, 
end_time3 = end_time1, 
Day_Trade = TRUE, 
nap_time_error = nap_time_error, 
initial_time = initial_time, 
final_time = final_time, 
freq_trade = freq_trade, 
w_twitter = w_twitter, 
w_stocktwits = w_stocktwits, 
Sentiment_Index_Threshold = Sentiment_Index_Threshold, 
Use_Delta_Sentiment = TRUE, 
Signal_File_Name = Signal_File_Name)

## End(Not run)

---

**Description**

This function takes as arguments the sentiment indexes and returns the decision.

**Usage**

Trade_Decision(
    Current_Sentiment_Index,
Trade_Decision

Past_Sentiment_Index,
Use_Delta_Sentiment,
Sentiment_Index_Threshold,
past_decision
)

Arguments

Current_Sentiment_Index
  The current sentiment index
Past_Sentiment_Index
  The sentiment index in (t-1)
Use_Delta_Sentiment
  If True the function will consider the difference in the sentiment index in the
decision.
Sentiment_Index_Threshold
  The threshold to define if the decision will be following or against the sentiment.
past_decision
  The last trade decision.

Value

The vector with the decision.

Examples

buy_sell_t1 <- 0.2
buy_sell_t <- 0.5
Use_Delta_Sentiment <- TRUE
Sentiment_Index_Threshold <- 0.5

decision <- Trade_Decision(Current_Sentiment_Index = buy_sell_t,
  Past_Sentiment_Index = buy_sell_t1,
  Use_Delta_Sentiment = Use_Delta_Sentiment,
  Sentiment_Index_Threshold = Sentiment_Index_Threshold,
  past_decision = decision
)
Index

* datasets
  my_dictionary, 7
check_frequency, 2
Close_Position, 3

  generate_trade_frequency, 3
get_sentiment_stocktwits, 4
get_sentiment_tweets, 5

havingIP, 6
my_dictionary, 7
operation_hours, 7
Start_Trading, 8
Trade_Decision, 11