Package 'rameritrade'

October 15, 2020

Title 'TD Ameritrade' API Interface for R

Version 0.1.4

Author Tony Trevisan [aut, cre]

Maintainer Tony Trevisan <anthonytrevisan@gmail.com>

URL https://github.com/tonytrevisan/rameritrade/, https://developer.tdameritrade.com/

BugReports https://github.com/tonytrevisan/rameritrade/issues

Description Use R to interface with the 'TD Ameritrade' API, including authentication, trading, price requests, account information, and option chains. A user will need a TD brokerage account and TD Ameritrade developer app. See README for authentication process and examples.

License GPL-3

Encoding UTF-8

LazyData true

Imports httr, urltools (>= 1.7.3), lubridate, dplyr, jsonlite, magrittr

RoxygenNote 7.1.1

Suggests testthat

NeedsCompilation no

Repository CRAN

Date/Publication 2020-10-15 13:20:02 UTC

R topics documented:

`rameritrade` *rameritrade: A package for using the TD Ameritrade API*

Description

The rameritrade package provides four categories of important functions:

1. authenticating into a TD Brokerage account
2. working with orders
3. getting account information
4. pulling pricing/market data

Details

Details for the functions can be found within the function help pages. Detailed examples also available within the README.

`td_accountData` *Get account balances positions, and orders returned as a list*

Description

Retrieves a account data for the accounts linked to the Access Token

Usage

```
td_accountData(output = "df", accessToken = NULL)
```

Arguments

`output`	Use 'df' for a list of 3 data frames containing balances, positions, and orders. Otherwise the data will be returned as a list of lists
`accessToken`	A valid Access Token must be set using `td_auth_accessToken`. The most recent Access Token will be used by default unless one is manually passed into the function.

Details

The output will be either a list of three data frames or a list of three lists that contain balances, positions, and orders for TD Ameritrade accounts linked to the access token. For historical orders, see td_orderSearch. The default is for a data frame output which is much cleaner.

Value

a list of requested account details

Examples

```
## Not run:

# Get stored refresh token
refreshToken = readRDS('/secure/location/')

# Generate a new access token
accessToken = td_auth_accessToken(refreshToken, 'consumerKey')

# Passing the accessToken is optional. The default will return balances
asDF = td_accountData()
asList = td_accountData('list',accessToken)


## End(Not run)
```

---

td_auth_accessToken *Auth Step 3: Get Access Token*

Description

Get a new Access Token using a valid Refresh Token

Usage

```
td_auth_accessToken(consumerKey, refreshToken)
```

Arguments

consumerKey	TD generated Consumer key for the registered TD app. Essentially an API key.
refreshToken	An existing Refresh Token generated using td_auth_refreshToken

Details

An Access Token is required for the functions within rameritrade. It serves as a user login to a TD Brokerage account. The token is valid for 30 minutes and allows the user to place trades, get account information, get order history, pull historical stock prices, etc. A Refresh Token is required to generate an Access Token. td_auth_refreshToken can be used to generate Refresh Tokens which stay valid for 90 days. The Consumer Key is generated automatically when an App is registered on the TD Ameritrade Developer site. By default, the Access Token is stored into options and will automatically be passed to downstream functions. However, the user can also submit an Access Token manually if multiple tokens are in use (for example: when managing more than one log in.)

When running this function manually (i.e. through RStudio), the function will check for a default Access Token. If the default Access Token has not expired, the user will be prompted to verify a new Access Token is desired. This may be the case if more than one TD login is being used. When running this function in a non-interactive environment (i.e. CRON Job), the default behavior will be to refresh the Access Token.

DISCLOSURE: This software is in no way affiliated, endorsed, or approved by TD Ameritrade or any of its affiliates. It comes with absolutely no warranty and should not be used in actual trading unless the user can read and understand the source code. The functions within this package have been tested under basic scenarios. There may be bugs or issues that could prevent a user from executing trades or canceling trades. It is also possible trades could be submitted in error. The user will use this package at their own risk.

Value

Access Token that is valid for 30 minutes. By default it is stored in options.

See Also

td_auth_loginURL to generate a login url which leads to an authorization code, td_auth_refreshToken to generate a Refresh Token using an existing Refresh Token or an authorization code with a call-back URL when loggin in manually, td_auth_accessToken to generate a new Access Token

Examples

```
## Not run:

# A valid Refresh Token can be fed into the function below for a new Access Token
refreshToken = readRDS('/secure/location/')
accessToken = td_auth_accessToken('TD_CONSUMER_KEY', refreshToken)


## End(Not run)
```

td_auth_loginURL *Auth Step 1: Generate LogIn URL*

Description

Create URL to grant App access to a TD Brokerage account

Usage

```
td_auth_loginURL(consumerKey, callbackURL)
```

Arguments

consumerKey	TD generated Consumer key for the registered TD app. Essentially an API key.
callbackURL	User generated Callback URL for the registered TD app

Details

To use the TD Ameritrade API, both a TD Brokerage account and a registered developer app are required. The developer app functions as a middle layer between the brokerage account and the API. A developer app should be registered on the TD Ameritrade Developer site. Once logged in to the developer site, use My Apps to register an application. An App will have a Consumer Key provided by TD that functions as an API key. The Consumer Key is auto generated and can be found under My Apps > Keys. The user must also create a Callback URL. The Callback URL can be anything. The example below assumes the Callback URL is https://myTDapp.

This function will use these two inputs to generate a URL where the user can log in to their standard TD Ameritrade Brokerage Account and grant the application access to the brokerage account, enabling the API. The Authorization Code generated at the end of the log in process will feed into td_auth_refreshToken. For questions, please reference the TD Ameritrade Authentication FAQ or see the examples in the rameritrade readme. If the callback URL uses punctuation, numbers, or an IP (e.g. 127.0.0.0.1), the output of this function may not work. Please reference the TD documentation to create the appropriate URL.

Value

login url to grant app permission to TD Brokerage account

See Also

td_auth_loginURL to generate a login url which leads to an authorization code, td_auth_refreshToken to generate a Refresh Token using an existing Refresh Token or an authorization code with a callback URL when loggin in manually, td_auth_accessToken to generate a new Access Token

Examples

```
## Not run:

# Visit the URL generated from the function below to log in to a TD Brokerage account
# Once a successful log in is completed the landing page will be a blank page
# The full URL of the landing page is the Authorization Code for td_auth_refreshToken

loginURL = td_auth_loginURL('https://myTDapp','consumerKey')


## End(Not run)
```

`td_auth_refreshToken` *Auth Step 2: Obtain Refresh Token*

Description

Get a Refresh Token using the Authorization Code or an existing Refresh Token

Usage

```
td_auth_refreshToken(consumerKey, callbackURL, codeToken)
```

Arguments

`consumerKey`	TD generated Consumer key for the registered TD app. Essentially an API key.
`callbackURL`	User generated Callback URL for the registered TD app
`codeToken`	Will be either an authorization code when manually logging in or a Refresh Token if the current Refresh Token is nearing expiration.

Details

Once a URL has been generated using `td_auth_loginURL`, a user can visit that URL to log into a TD brokerage account, granting the TD app access to the account. Once the button "Allow" is pressed, the user will be redirected, potentially to "This site can't be reached". This indicates a successful log in. The URL of this page contains the Authorization Code. Paste the entire URL, not just the Authorization Code, into td_auth_refreshToken. The authorization code will be an extremely long alpha numeric string starting with 'https'.

The output of td_auth_refreshToken will be a Refresh Token which will be used to gain access to the TD Brokerage account(s) going forward. The Refresh Token will be valid for 90 days. Be sure to save the Refresh Token to a safe location or the manual log in process will be required again. The user can use td_auth_refreshToken to reset the token before expiration.

The Refresh Token output should be saved in a very safe location, but also accessible. It will be needed to generate an Access Token using `td_auth_accessToken`, which is used for general account access. The Access Token expires after 30 minutes.

Note: When running this function interactively (e.g. through RStudio) using an existing Refresh Token, the function will check the days left until expiration for the Refresh Token being passed. If the remaining time is greater than 15 days, the user will be prompted to verify that a new Refresh Token should be created. The user can select to request a new token, but there is no net benefit in doing so and TD encourages limiting new token generation. When running this function in a non-interactive environment (e.g. CRON Job), if the remaining time until expiration is greater than 15 days, the default behavior will be to NOT reset the Refresh Token because the new token will have the same access and capabilities as the existing token.

Value

Refresh Token that is valid for 90 days

See Also

td_auth_loginURL to generate a login url which leads to an authorization code, td_auth_refreshToken to generate a Refresh Token using an existing Refresh Token or an authorization code with a callback URL when logging in manually, td_auth_accessToken to generate a new Access Token

Examples

```
## Not run:

# Initial access will require manually logging in to the URL from td_auth_loginURL
# After a successful log in, the URL authorization code can be fed with a callbackURL
refreshToken = td_auth_refreshToken(consumerKey = 'TD_CONSUMER_KEY',
                                    callbackURL = 'https://myTDapp',
                                   codeToken = 'https://myTDapp/?code=Auhtorizationcode')

# If an existing Refresh Token exists. Pass this. CallbackURL is not needed.
refreshToken = readRDS('/secure/location/')
refreshToken = td_auth_refreshToken(consumerKey = 'TD_CONSUMER_KEY',
                                    codeToken = refreshToken)

# Save the Refresh Token somewhere safe where it can be retrieved
saveRDS(refreshToken,'/secure/location/')


## End(Not run)
```

---

td_cancelOrder *Cancel an Open Order*

Description

Pass an Order ID and Account number to cancel an existing open order

Usage

```
td_cancelOrder(orderId, accountNumber, accessToken = NULL)
```

Arguments

orderId	A valid TD Ameritrade Order ID
accountNumber	The TD brokerage account number associated with the Access Token
accessToken	A valid Access Token must be set using td_auth_accessToken. The most recent Access Token will be used by default unless one is manually passed into the function.

Value

order API URL. Message confirming cancellation

Examples

```
 ## Not run:

td_cancelOrder(orderId = 123456789, accountNumber = 987654321)


## End(Not run)
```

---

td_marketHours *Get Market Hours*

Description

Returns a list output for current day and specified market that details the trading window

Usage

```
td_marketHours(
  marketType = c("EQUITY", "OPTION", "BOND", "FUTURE", "FOREX"),
  accessToken = NULL
)
```

Arguments

marketType	The asset class to pull: 'EQUITY','OPTION','BOND','FUTURE','FOREX'. Default is EQUITY
accessToken	A valid Access Token must be set using td_auth_accessToken. The most recent Access Token will be used by default unless one is manually passed into the function.

Value

List output of times and if the current date is a trading day

Examples

```
## Not run:

# Access Token must be set using td_auth_accessToken
# Market hours for the current date
td_marketHours()
td_marketHours('2020-06-24', 'OPTION')


## End(Not run)
```

`td_optionChain` *Get Options Chain*

Description

Search an Option Chain for a specific ticker

Usage

```
td_optionChain(
  ticker,
  strikes = 10,
  inclQuote = TRUE,
  startDate = Sys.Date(),
  endDate = Sys.Date() + 360,
  accessToken = NULL
)
```

Arguments

`ticker`	underlying ticker for the options chain
`strikes`	the number of strikes above and below the current strike
`inclQuote`	set TRUE to include pricing details (will be delayed if account is set for delayed quotes)
`startDate`	the start date for expiration (should be greater than or equal to today). Format: yyyy-mm-dd
`endDate`	the end date for expiration (should be greater than or equal to today). Format: yyyy-mm-dd
`accessToken`	A valid Access Token must be set using `td_auth_accessToken`. The most recent Access Token will be used by default unless one is manually passed into the function.

Details

Return a list containing two data frames. The first is the underlying data for the symbol. The second item in the list is a data frame that contains the options chain for the specified ticker.

Value

a list of 2 data frames - underlying and options chain

Examples

```
## Not run:

# Pull all option contracts expiring over the next 6 months
# with 5 strikes above and below the at-the-money price
td_optionChain(ticker = 'SPY',
             strikes = 5,
             endDate = Sys.Date() + 180)


## End(Not run)
```

---

td_orderDetail *Get Details for a Single Order*

Description

Pass an order ID and Account number to get details such as status, quantity, ticker, executions (if applicable), account, etc.

Usage

```
td_orderDetail(orderId, accountNumber, accessToken = NULL)
```

Arguments

orderId	A valid TD Ameritrade Order ID
accountNumber	The TD brokerage account number associated with the Access Token
accessToken	A valid Access Token must be set using td_auth_accessToken. The most recent Access Token will be used by default unless one is manually passed into the function.

Value

list of order details

Examples

```
## Not run:

# Get stored refresh token
refreshToken = readRDS('/secure/location/')

# generate a new access token
```

```
accessToken = td_auth_accessToken(refreshToken, 'consumerKey')

# Get order details for a single order
# Passing Access Token is optional once it's been set
td_orderDetail(orderId = 123456789, accountNumber = 987654321)


## End(Not run)
```

`td_orderSearch` *Search for orders by date*

Description

Search for orders associated with a TD account over the previous 60 days. The result is a list of three objects:

1. jsonlite formatted extract of all orders
2. all entered orders with details
3. a data frame of all executed orders with the executions

Usage

```
td_orderSearch(
  accountNumber,
  startDate = Sys.Date() - 30,
  endDate = Sys.Date(),
  maxResults = 50,
  orderStatus = "",
  accessToken = NULL
)
```

Arguments

`accountNumber`	The TD brokerage account number associated with the Access Token
`startDate`	Orders from a certain date with. Format yyyy-mm-dd. TD indicates there is a 60 day max, but this limit may not always apply
`endDate`	Filter orders that occurred before a certain date. Format yyyy-mm-dd
`maxResults`	the max results to return in the query
`orderStatus`	search by order status (ACCEPTED, FILLED, EXPIRED, CANCELED, REJECTED, etc)
`accessToken`	A valid Access Token must be set using `td_auth_accessToken`. The most recent Access Token will be used by default unless one is manually passed into the function.

Value

a list of three objects: a jsonlite formatted extract of all orders, all entered orders with details, a data frame of all executed orders with the executions

Examples

```
## Not run:

# Get all orders run over the last 50 days (up to 500)
td_orderSearch(accountNumber = 987654321,
             startDate = Sys.Date()-days(50),
             maxResult = 500, orderStatus = 'FILLED')


## End(Not run)
```

`td_placeOrder` *Place Order for a specific account*

Description

Place trades through the TD Ameritrade API using a range of parameters

Usage

```
td_placeOrder(
  accountNumber,
  ticker,
  quantity,
  instruction,
  orderType = "MARKET",
  limitPrice = NULL,
  stopPrice = NULL,
  assetType = c("EQUITY", "OPTION"),
  session = "NORMAL",
  duration = "DAY",
  stopPriceBasis = NULL,
  stopPriceType = NULL,
  stopPriceOffset = NULL,
  accessToken = NULL
)
```

Arguments

`accountNumber`	The TD brokerage account number associated with the Access Token
`ticker`	a valid Equity/ETF or option. If needed, use td_symbolDetail to confirm. This should be a ticker/symbol, not a CUSIP

quantity	the number of shares to be bought or sold. Must be an integer.
instruction	Equity instructions include 'BUY', 'SELL', 'BUY_TO_COVER', or 'SELL_SHORT'. Options instructions include 'BUY_TO_OPEN', 'BUY_TO_CLOSE', 'SELL_TO_OPEN', or 'SELL_TO_CLOSE'
orderType	MARKET, LIMIT (requiring limitPrice), STOP (requiring stopPrice), STOP_LIMIT, TRAILING_STOP (requiring stopPriceBasis, stopPriceType, stopPriceOffset)
limitPrice	the limit price for a LIMIT or STOP_LIMIT order
stopPrice	the stop price for a STOP or STOP_LIMIT order
assetType	EQUITY or OPTION. No other asset types are available at this time. EQUITY is the default.
session	NORMAL for normal market hours, AM or PM for extended market hours
duration	how long will the trade stay open without a fill: DAY, GOOD_UNTIL_CANCEL, FILL_OR_KILL
stopPriceBasis	LAST, BID, or ASK which is the basis for a STOP, STOP_LIMIT, or TRAILING_STOP
stopPriceType	the link to the stopPriceBasis. VALUE for dollar difference or PERCENT for a percentage offset from the price basis
stopPriceOffset	an integer that indicates the offset used for the stopPriceType, 10 and PERCENT is a 10 percent offset from the current price basis. 5 and VALUE is a 5 dollar offset from the current price basis
accessToken	A valid Access Token must be set using td_auth_accessToken. The most recent Access Token will be used by default unless one is manually passed into the function.

Details

A valid account and access token must be passed. An access token will be passed by default when td_auth_accessToken is executed successfully and the token has not expired, which occurs after 30 minutes. Only equities and options can be traded at this time. This function is built to allow a single trade submission. More complex trades can be executed through the API, but a custom function or submission will need to be constructed. To build more custom trading strategies, reference the TD Ameritrade API Instructions or the order sample guide. A full list of the input parameters and details can be found at the links above. Please note that in rare cases, the documentation may not be accurate in the API section, so the Order Sample guide is a better reference. TEST ALL ORDERS FIRST WITH SMALL DOLLAR AMOUNTS!!!

Four parameters are required for submission: ticker, instruction, quantity, and account number associated with the Access Token. The following parameters default: session - NORMAL, duration - DAY, asset type - EQUITY, and order type - MARKET

Value

the trade id, account id, and other order details

Warning

TRADES THAT ARE SUCCESSFULLY ENTERED WILL BE SUBMITTED IMMEDIATELY THERE IS NO REVIEW PROCESS. THIS FUNCTION HAS HUNDREDS OF POTENTIAL COMBINATIONS AND ONLY A HANDFUL HAVE BEEN TESTED. IT IS STRONGLY RECOMMENDED TO TEST THE DESIRED ORDER ON A VERY SMALL QUANTITY WITH LITTLE MONEY AT STAKE. ANOTHER OPTION IS TO USE LIMIT ORDERS FAR FROM THE CURRENT PRICE. TD AMERITRADE HAS THEIR OWN ERROR HANDLING BUT IF A SUCCESSFUL COMBINATION IS ENTERED IT COULD BE EXECUTED IMMEDIATELY. DOUBLE CHECK ALL ENTRIES BEFORE SUBMITTING.

Examples

```
## Not run:

# Get stored refresh token
refreshToken = readRDS('/secure/location/')

# generate a new access token
accessToken = td_auth_accessToken(refreshToken, 'consumerKey')

# Set Account Number
accountNumber = 1234567890

# Standard market buy order
# Every order must have at least these 4 paramters
td_placeOrder(accountNumber = accountNumber,
            ticker = 'AAPL',
            quantity = 1,
            instruction = 'buy')

# Stop limit order - good until canceled
td_placeOrder(accountNumber = accountNumber,
            ticker = 'AAPL',
            quantity = 1,
            instruction = 'sell',
            duration = 'good_till_cancel',
            orderType = 'stop_limit',
            limitPrice = 98,
            stopPrice = 100)

# Trailing Stop Order
td_placeOrder(accountNumber = accountNumber,
            ticker='AAPL',
            quantity = 1,
            instruction='sell',
            orderType = 'trailing_stop',
            stopPriceBasis = 'BID',
            stopPriceType = 'percent',
            stopPriceOffset = 10)

# Option Order with a limit price
```

```
td_placeOrder(accountNumber = accountNumber,
              ticker = 'SLV_091820P24.5',
              quantity = 1,
              instruction = 'BUY_TO_OPEN',
              duration = 'Day',
              orderType = 'LIMIT',
              limitPrice = .02,
              assetType = 'OPTION')


## End(Not run)
```

`td_priceHistory` *Get price history for a multiple securities*

Description

Open, Close, High, Low, and Volume for one or more securities

Usage

```
td_priceHistory(
  tickers = c("AAPL", "MSFT"),
  startDate = Sys.Date() - 30,
  endDate = Sys.Date(),
  freq = c("daily", "1min", "5min", "10min", "15min", "30min"),
  accessToken = NULL
)
```

Arguments

`tickers`	a vector of tickers - no more than 15 will be pulled. for bigger requests, split up the request or use Tiingo, FMP Cloud, or other free data providers
`startDate`	the Starting point of the data
`endDate`	the Ending point of the data
`freq`	the frequency of the interval. Can be daily, 1min, 5min, 10min, 15min, or 30min
`accessToken`	A valid Access Token must be set using `td_auth_accessToken`. The most recent Access Token will be used by default unless one is manually passed into the function.

Details

Pulls price history for a list of security based on the parameters that include a date range and frequency of the interval. Depending on the frequency interval, data can only be pulled back to a certain date. For example, at a one minute interval, data can only be pulled for 30-35 days. Prices are adjusted for splits but not dividends.

PLEASE NOTE: Large data requests will take time to pull back because of the looping nature. TD Does not allow bulk ticker request, so this is simply running each ticker individually. For faster and better historical data pulls, try Tiingo or FMP Cloud

Value

a tibble of historical price data

Examples

```
## Not run:

# Set the access token and a provide a vector of one or more tickers
refreshToken = readRDS('/secure/location/')
accessToken = td_auth_accessToken(refreshToken, 'consumerKey')
tickHist5min = td_priceHistory(c('TSLA','AAPL'), freq='5min')

# The default is daily. Access token is optional once it's been set
tickHistDay = td_priceHistory(c('SPY','IWM'), startDate = '1990-01-01')


## End(Not run)
```

---

td_priceQuote *Get Quotes for specified tickers in List form*

Description

Enter tickers for real time or delayed quotes returned as a list

Usage

```
td_priceQuote(tickers = c("AAPL", "MSFT"), output = "df", accessToken = NULL)
```

Arguments

tickers	One or more tickers
output	indication on whether the data should be returned as a list or df. The default is 'df' for data frame, anything else would be a list.
accessToken	A valid Access Token must be set using td_auth_accessToken. The most recent Access Token will be used by default unless one is manually passed into the function.

Details

Quotes may be delayed depending on agreement with TD Ameritrade. If the account is set up for real-time quotes then this will return real-time. Otherwise the quotes will be delayed.

Value

a list or data frame with quote details for each valid ticker submitted

Examples

```
## Not run:

# Get stored refresh token
refreshToken = readRDS('/secure/location/')

# generate a new access token
accessToken = td_auth_accessToken('consumerKey', refreshToken)

# Pass one or more tickers as a vector
# accessToken is optional once it is set
quoteSPY = td_priceQuote('SPY')
quoteList = td_priceQuote(c('GOOG','TSLA'), output = 'list', accessToken)


## End(Not run)
```

---

td_symbolDetail *Get ticker details*

Description

Get identifiers and fundamental data for a specific ticker

Usage

```
td_symbolDetail(ticker, accessToken = NULL)
```

Arguments

ticker	a valid ticker or symbol
accessToken	A valid Access Token must be set using td_auth_accessToken. The most recent Access Token will be used by default unless one is manually passed into the function.

Value

data frame of ticker details

Examples

```
## Not run:

# Details for Apple
td_symbolDetail('AAPL')


## End(Not run)
```

`td_transactSearch` *Search for all Transaction types*

Description

Can pull trades as well as transfers, dividend reinvestment, interest, etc. Any activity associated with the account.

Usage

```
td_transactSearch(
  accountNumber,
  startDate = Sys.Date() - 30,
  endDate = Sys.Date(),
  transType = "All",
  accessToken = NULL
)
```

Arguments

`accountNumber`	The TD brokerage account number associated with the Access Token
`startDate`	Transactions after a certain date. Will not pull back transactions older than 1 year. format yyyy-mm-dd
`endDate`	Filter transactions that occurred before a certain date. format yyyy-mm-dd
`transType`	Filter for a specific Transaction type. No entry will return all types. For example: TRADE, CASH_IN_OR_CASH_OUT, CHECKING, DIVIDEND, INTEREST, OTHER
`accessToken`	A valid Access Token must be set using `td_auth_accessToken`. The most recent Access Token will be used by default unless one is manually passed into the function.

Value

a jsonlite data frame of transactions

Examples

```
## Not run:

# Access Token must be set using td_auth_accessToken
# Transactions for the last 5 days
td_transactSearch(accountNumber = 987654321,
                startDate = Sys.Date()-days(5))


## End(Not run)
```

Index