Package 'pedquant'

August 27, 2020

Version 0.1.6

Title Public Economic Data and Quantitative Analysis

Description Provides an interface to access public economic and financial data for economic research and quantitative analysis. The data sources including NBS, FRED, Yahoo Finance, 163 Finance and etc.

Depends R (>= 3.1.0)

Imports data.table, TTR, zoo, curl, xml2, httr, rvest, jsonlite, stringi, readxl, readr, ggplot2, scales, gridExtra, plotly

Suggests knitr, rmarkdown

License GPL-3

URL https://github.com/ShichenXie/pedquant

BugReports https://github.com/ShichenXie/pedquant/issues

LazyData true

RoxygenNote 7.1.1

Encoding UTF-8

NeedsCompilation no

Author Shichen Xie [aut, cre]

Maintainer Shichen Xie <xie@shichen.name>

Repository CRAN

Date/Publication 2020-08-27 07:50:03 UTC

R topics documented:

---

ed_code *code list by category*

Description

ed_code get the code list of country, currency, stock exchange, commodity exchange and administrative district of mainland of China.

Usage

```
ed_code(cate = NULL)
```

Arguments

cate	The available category values including 'country', 'currency', 'stock_exchange', 'commodity_exchange', 'china_district'.

Examples

```
## Not run:
# specify the categories
code_list1 = ed_code(cate = c('country', 'currency'))

# interactivly return code list
code_list2 = ed_code()


## End(Not run)
```

ed_fred *query FRED economic data*

Description

ed_fred provides an interface to access the economic data provided by FRED (https://fred.stlouisfed.org)

Usage

```
ed_fred(symbol = NULL, date_range = "10y", from = NULL,
  to = Sys.Date(), na_rm = FALSE, print_step = 1L)
```

Arguments

symbol	symbols of FRED economic indicators. It is available via function ed_fred_symbol or its website. Default is NULL, which calls ed_fred_symbol in the back.
date_range	date range. Available value includes '1m'-'11m', 'ytd', 'max' and '1y'-'ny'. Default is '10y'.
from	the start date. Default is NULL. If it is NULL, then calculate using date_range and end date.
to	the end date. Default is the current date.
na_rm	logical, whether to remove missing values. Default is FALSE
print_step	a non-negative integer, which will print symbol name by each print_step iteration. Default is 1L.

Value

a list of dataframes with columns of symbol, name, date, value, geo, unit. The geo column might be NA according to local internet connection.

Examples

```
dat = ed_fred(c("A191RL1A225NBEA", "GDPCA"))
```

ed_fred_symbol *symbol of FRED economic data*

Description

ed_fred_symbol provides an interface to search symbols of economic data from FRED by category or keywords.

Usage

```
ed_fred_symbol(category = NULL, keywords = NULL, ...)
```

Arguments

category	the category id. If it is NULL, then search symbols from the top categories step by step.
keywords	the query text. If it is NULL, the function will search symbols by category.
...	ignored parameters

Examples

```
## Not run:
# search symbols by category
# from top categories
symbol_dt1 = ed_fred_symbol()
# specify the initial categories
symbol_dt2 = ed_fred_symbol(category = 1)

# search symbol by keywords
symbol_dt3 = ed_fred_symbol(keywords = "gdp china")


## End(Not run)
```

ed_nbs *query NBS economic data*

Description

ed_nbs provides an interface to query economic data from National Bureau of Statistics of China (NBS, http://data.stats.gov.cn/).

Usage

```
ed_nbs(symbol = NULL, freq = NULL, geo_type = NULL, subregion = NULL,
  date_range = "10y", from = NULL, to = Sys.Date(), na_rm = FALSE,
  eng = FALSE)
```

Arguments

symbol	symbols of NBS indicators. It is available via ed_nbs_symbol. Default is NULL.
freq	the frequency of NBS indicators, including 'monthly', 'quarterly', 'yearly'. Default is NULL.
geo_type	geography type in NBS, including 'nation', 'province', 'city'. Default is NULL.
subregion	codes of province or city, which is available via ed_nbs_subregion. Default is NULL.
date_range	date range. Available value includes '1m'-'11m', 'ytd', 'max' and '1y'-'ny'. Default is '10y'.
from	the start date. Default is NULL. If it is NULL, then calculate using date_range and end date.
to	the end date. Default is the current date.
na_rm	logical. Whether to remove missing values from datasets. Default is FALSE.
eng	logical. The language of the query results is in English or in Chinese Default is FALSE.

Examples

```
## Not run:
# query NBS data without setting any parameters
dt = ed_nbs()

# specify paratmeters
dt1 = ed_nbs(geo_type='nation', freq='quarterly', symbol='A010101')
# or using 'n'/'q' represents 'nation'/'quarterly'
dt2 = ed_nbs(geo_type='n', freq='q', symbol='A010101')


# query data in one province
dt3 = ed_nbs(geo_type='province', freq='quarterly',
  symbol='A010101', subregion='110000')

# query data in all province
dt4 = ed_nbs(geo_type='province', freq='quarterly',
  symbol='A010101', subregion='all')

## End(Not run)
```

ed_nbs_subregion *subregion code of NBS economic data*

Description

ed_nbs_subregion query province or city code from NBS

Usage

```
ed_nbs_subregion(geo_type = NULL, eng = FALSE)
```

Arguments

geo_type	geography type in NBS, including 'province', 'city'. Default is NULL.
eng	logical. The language of the query results is in English or in Chinese. Default is FALSE.

Examples

```
## Not run:
# province code
prov1 = ed_nbs_subregion(geo_type = 'province')
# or using 'p' represents 'province'
prov2 = ed_nbs_subregion(geo_type = 'p')

# city code in Chinese
# city = ed_nbs_subregion(geo_type = 'c', eng = FALSE)
# city code in English
city = ed_nbs_subregion(geo_type = 'c', eng = TRUE)

## End(Not run)
```

ed_nbs_symbol *symbol of NBS economic data*

Description

ed_nbs_symbol provides an interface to query symbols of economic indicators from NBS.

Usage

```
ed_nbs_symbol(symbol = NULL, geo_type = NULL, freq = NULL, eng = FALSE)
```

Arguments

symbol	symbols of NBS indicators.
geo_type	geography type in NBS, including 'nation', 'province', 'city'. Default is NULL.
freq	the frequency of NBS indicators, including 'monthly', 'quarterly', 'yearly'. Default is NULL.
eng	logical. The language of the query results is in English or in Chinese. Default is FALSE.

Examples

```
# query symbol interactively
## Not run:
sym = ed_nbs_symbol()
## End(Not run)
```

---

md_cate *query main market data by category*

Description

md_cate provides an interface to access main market data in five categories, including forex, money, bond, index, commodity.

Usage

```
md_cate(cate = NULL, symbol = NULL, date_range = "3y", from = NULL,
  to = Sys.Date(), print_step = 1L, ...)
```

Arguments

cate	the market category, forex, money, bond, index, commodity. Default is NULL.
symbol	symbols of main market indicators.
date_range	date range. Available value includes '1m'-'11m', 'ytd', 'max' and '1y'-'ny'. Default is '3y'.
from	the start date. Default is NULL. If it is NULL, then calculate using date_range and end date.
to	the end date. Default is the current date.
print_step	a non-negative integer, which will print symbol name by each print_step iteration. Default is 1L.
...	ignored parameters

Examples

```
## Not run:
dat = md_cate()

## End(Not run)
```

`md_future` *query future market data*

Description

`md_future` query future market prices data. Only Chinese future market has been considered currently.

Usage

```
md_future(symbol = NULL, source = "sina", freq = "daily",
  date_range = "3y", from = NULL, to = Sys.Date(), print_step = 1L)
```

Arguments

`symbol`	symbols of future market data. It is available via function `md_future_symbol` or its website. Default is NULL.
`source`	the data source is sina finance (https://finance.sina.com.cn/futuremarket/).
`freq`	the frequency of NBS indicators, including '5m','15m','30m','60m','daily'. Default is 'daily'.
`date_range`	date range. Available value includes '1m'-'11m', 'ytd', 'max' and '1y'-'ny'. Default is '3y'.
`from`	the start date. Default is NULL. If it is NULL, then calculate using date_range and end date.
`to`	the end date. Default is the current date.
`print_step`	a non-negative integer, which will print symbol name by each print_step iteration. Default is 1L.

Examples

```
## Not run:
dt1 = md_future(symbol = c('J0', 'RB0', 'M0', 'CF0', 'IH0', 'IF0', 'IC0'))

# interactivly choose symbols
dt2 = md_future()

## End(Not run)
```

md_future_symbol *symbol of future market data*

Description

md_future_symbol search the symbols in future market indicators that provided by sina finance only currently.

Usage

```
md_future_symbol()
```

Examples

```
## Not run:
# interactivly search future market symbols
sybs = md_future_symbol()


## End(Not run)
```

md_stock *query stock market data*

Description

md_stock provides an interface to query EOD (end of date) stock prices.

Usage

```
md_stock(symbol, source = "yahoo", type = "history", freq = "daily",
  date_range = "3y", from = NULL, to = Sys.Date(), adjust = "split",
  print_step = 1L, ...)
```

Arguments

symbol	symbols of stock shares.
source	the available data sources are 'yahoo' (https://finance.yahoo.com) and '163' (https://money.163.com).
type	the data type, including history, adjfactor and spot. Default is history.
freq	default is daily. It supports daily, weekly and monthly for yahoo data; daily for 163 data.
date_range	date range. Available value including '1m'-'11m', 'ytd', 'max' and '1y'-. Default is '3y'.

from	the start date. Default is NULL.
to	the end date. Default is current system date.
adjust	adjust the OHLC prices for split (default), or dividend (both split and dividend). If it is NULL, download the original data. For the yahoo data, the original data already adjust for split, and use the 'close_adj' column to adjust; for the 163 data, the original doesnot adjust any factors, and use the splits, dividends and issues to adjust.
print_step	A non-negative integer. Print symbol name by each print_step iteration. Default is 1L.
...	Additional parameters.

Examples

```
## Not run:
# Example I
# query history prices from yahoo
dt_yahoo1 = md_stock(symbol=c("^GSPC", "000001.SS"))

# FAANG
FAANG = md_stock(c('FB', 'AMZN', 'AAPL', 'NFLX', 'GOOG'), date_range = 'max')

# for Chinese shares/fund
## the symbol without suffix
dt_yahoo2 = md_stock(c("000001", "^000001", "512510"))
## the symbol with suffix
dt_yahoo3 = md_stock(c("000001.sz", "000001.ss"))

# adjust factors, splits and dividend
dt_adj = md_stock(symbol=c("AAPL", "000001.SZ", "000001.SS"),
                  type='adjfactor', date_range='max')


# Example II
# query history prices from 163
dt1 = md_stock(symbol=c('600000', '000001', '^000001', '^399001'),
               source="163")

# valuation ratios (pe, pb, ps)
# only available for stock shares in sse and szse
dt2 = md_stock(symbol=c('600000', '000001', '^000001', '^399001'),
               source="163", valuation = TRUE)


# Example III
# query spot prices
dt_spot1 = md_stock(symbol=c('600000.SS', '000001.SZ', '000001.SS', '399001.SZ'),
                  type='spot', source="163")

# query spot prices of all A shares in sse and szse
dt_spot2 = md_stock(symbol='a', source="163", type='spot')
# query spot prices of all A/B shares and index in sse and szse
```

```
dt_spot3 = md_stock(symbol=c('a', 'b', 'index'), source="163", type='spot')

# show spot prices and sector/industry
dt_spot4 = md_stock(symbol = c('a', 'b', 'index', 'fund'), source = '163',
  type = 'spot', show_tags = TRUE)


## End(Not run)
```

`md_stock_adjust` *adjust stock price for split and dividend*

Description

`md_stock_adjust` adjusts the open, high, low and close stock prices for split and dividend.

Usage

```
md_stock_adjust(dt, source, adjust = "split", adjfactor = NULL)
```

Arguments

`dt`	a list/dataframe of time series datasets that didnt adjust for split or dividend.
`source`	the available data sources are 'yahoo' (https://finance.yahoo.com) and '163' (https://money.163.com).
`adjust`	adjust the OHLC prices for split (default), or dividend (both split and dividend). If it is NULL, return the original data. For the yahoo data, the original data already adjust for split, and use the 'close_adj' column to adjust; for the 163 data, the original doesnot adjust any factors, and use the splits, dividends and issues to adjust.
`adjfactor`	adjust factors, including splits and dividends. Defaults to NULL, which will load adjust factors from source. It can also download from md_stock when type set as adjfactor.

Examples

```
dt = md_stock('600547', source = '163', date_range = 'max',
              type = 'history', adjust = NULL)
ds = md_stock('600547', source = '163', date_range = 'max',
              type = 'adjfactor')

dtadj = md_stock_adjust(dt, source = '163', adjust = 'dividend',
                        adjfactor = ds)
```

`md_stock_financials` *query financial statements*

Description

`md_stock_financials` provides an interface to query financial statements and indicators of listed companies in SSE and SZSE.

Usage

```
md_stock_financials(symbol, type = NULL, print_step = 1L)
```

Arguments

`symbol`	symbol of stock shares.
`type`	the type of financial statements.
`print_step`	A non-negative integer. Print symbol name by each print_step iteration. Default is 1L.

Examples

```
## Not run:
# interactively specify type of financial table
dat1 = md_stock_financials("000001")

# manually specify type of financial table
# type = "fr0"
dat2 = md_stock_financials("000001", type="fs0")
# or type = "fr0_summary"
dat3 = md_stock_financials("000001", type="fs0_summary")

# multiple symbols and statements
dat4 = md_stock_financials(c("000001", "600000"), type = "fi")

# dupont analysis indicators
fs_idx = md_stock_financials(c('000001', '^000001'), type = 'dupont')


## End(Not run)
```

md_stock_symbol *symbol components of exchange or index*

Description

md_stock_symbol returns all stock symbols of stock exchange or index.

Usage

```
md_stock_symbol(exchange = NULL, index = NULL)
```

Arguments

exchange	the available stock exchanges are sse, szse, hkex, amex, nasdaq, nyse.
index	the stock index symbol provided by China Securities Index Co.Ltd (http://www.csindex.com.cn).

Examples

```
## Not run:
# get stock symbols in a stock exchange
## specify the name of exchange
ex_syb1 = md_stock_symbol(exchange = c('sse', 'szse'))

## choose stock exchanges interactivly
ex_syb2 = md_stock_symbol()


# get stock components of a stock index (only in sse and szse)
index_syb = md_stock_symbol(index = c('000001', '000016', '000300', '000905'))


## End(Not run)
```

pq_addti *adding technical indicators*

Description

pq_addti creates technical indicators on provided datasets use TTR package.

Usage

```
pq_addti(dt, ...)
```

Arguments

dt	a list/dataframe of time series datasets.
...	list of technical indicator parameters: sma = list(n=50), macd = list().

1. There are four types of parameters.
   - set by default and do not required, such as 'OHLC', 'HLC', 'HL' and 'volume'.
   - set by default and can be modified, such as 'price', 'prices', 'x'. Its default value is 'close' or 'value' column.
   - always required, such as 'y', 'w'.
   - numeric parameters, such as 'n', 'sd', 'v', 'nFast', 'nSlow', 'nSig', 'accel'. These parameters should be provided, otherwise using default values in corresponding function.
2. TTR functions are summarized in below. See TTR package's help document for more detailed parameters.
   - moving averages: SMA, EMA, DEMA, WMA, EVWMA, ZLEMA, VWAP, VMA, HMA, ALMA, GMMA
   - rolling functions: runMin, runMax, runMean, runMedian; runCov, runCor; runVar, runSD, runMAD; runSum, wilderSum
   - bands / channels: BBands, PBands, DonchianChannel
   - SAR, ZigZag
   - trend direction/strength: aroon, CCI, ADX, TDI, VHF, EMV
   - volatility measures: ATR, chaikinVolatility, volatility, SNR
   - money flowing into/out: OBV, chaikinAD, CLV, CMF, MFI, williamsAD
   - rate of change / momentum: ROC, momentum, KST, TRIX
   - oscillator: MACD, DPO, DVI, ultimateOscillator; RSI, CMO; stoch, SMI, WPR

Examples

```
# load data
data('ssec')

# add technical indicators
dt_ti1 = pq_addti(ssec, sma=list(n=20), sma=list(n=50), macd = list())

# only technical indicators
dt_ti2 = pq_addti(ssec, sma=list(n=20), sma=list(n=50), macd = list(), col_kp = FALSE)

# self-defined technical indicators
bias = function(x, n=50, maType='SMA') {
    library(TTR)
    (x/do.call(maType, list(x=x, n=n))-1)*100
}

dt_ti3 = pq_addti(ssec, bias = list(n = 200))
```

pq_index *creating weighted index*

Description

pq_index creates a sector/industry index using the method of weighted geometric mean, based on a set of data and corresponding weights.

Usage

```
pq_index(dt, x = "close|value", w = "cap_total", base_value = 1,
  base_date = NULL, name = NULL)
```

Arguments

dt	a list/dataframe of time series dataset
x	the name of column to create index. Default is 'close|value'
w	the name of weights column. Default is 'cap_total'. If x is not available or is NULL, then using equal weights.
base_value	the base value of index. Default is 1.
base_date	the base date of index. Default is the minimum date.
name	the name of index. Default is NULL, then using 'index'.

Examples

```
# Example I bank share index
# load data
bank_symbol = c('601988', '601288', '601398', '601939', '601328')
bank_dat = md_stock(bank_symbol, source='163', date_range = 'max')

# creating index
bank_index = pq_index(bank_dat, x='close', w='cap_total')
# pq_plot(bank_index)
```

pq_perf *creating performance trends*

Description

pq_perf provides an easy way to create the performance trends for a set of time series data.

Usage

```
pq_perf(dt, date_range = "max", from = NULL, to = Sys.Date(),
  x = "close|value", base_value = 1)
```

Arguments

dt	a list/dataframe of time series dataset
date_range	date range. Available value includes '1m'-'11m', 'ytd', 'max' and '1y'-'ny'. Default is max.
from	the start date. Default is NULL. If it is NULL, then calculate using date_range and end date.
to	the end date. Default is the current date.
x	the name of column to calculate. Default is 'close|value'.
base_value	the base value of performance index. Default is 0.

Examples

```
# load data
dat = md_stock(c('000001', '^000001'), date_range = 'max', source = '163')

# create performance trends
perf = pq_perf(dat)
# pq_plot(perf)
```

pq_plot *creating charts for time series*

Description

pq_plot provides an easy way to create charts for time series dataset based on predefined formats.

Usage

```
pq_plot(dt, chart_type = "line", freq = NULL, date_range = "max",
  from = NULL, to = Sys.Date(), x = "close|value", addti = list(volume
  = list()), linear_trend = NULL, perf = FALSE, yaxis_log = FALSE,
  color_up = "#CF002F", color_down = "#000000", multi_series = list(nrow
  = NULL, ncol = NULL), rm_weekend = NULL, title = NULL,
  interact = FALSE, ...)
```

Arguments

dt	a list/dataframe of time series dataset
chart_type	chart type, including line, step, bar, candle.
freq	the frequency that the input daily data will converted to. It supports weekly, monthly, quarterly and yearly.
date_range	date range. Available value includes '1m'-'11m', 'ytd', 'max' and '1y'-'ny'. Default is max.
from	the start date. Default is NULL. If it is NULL, then calculate using date_range and end date.
to	the end date. Default is the current date.
x	the name of column display on chart.
addti	list of technical indicators or numerical columns in dt. For technical indicator, it is calculated via pq_addti, which including overlay and oscillator indicators.
linear_trend	a numeric vector. Default is NULL. If it is not NULL, then display linear trend lines on charts.
perf	logical, display the performance of input series. Default is FALSE. If it is TRUE, then call pq_code to convert data into performance trends.
yaxis_log	logical. Default is FALSE.
color_up	the color indicates price going up
color_down	the color indicates price going down
multi_series	a list. It display the number of ncol or nrow, and the yaxis scales in 'free'/'free_y'/'free_x'. Default is NULL.
rm_weekend	whether to remove weekends in xaxis. The default is TRUE for candle and bar chart, and is FALSE for line and step chart.
title	chart title. It will added to the front of chart title if it is specified.
interact	whether to create a interactive graphics, defaults to FALSE.
...	ignored

Examples

```
# single symbol
ssec = md_stock('^000001', source='163', date_range = 'max')
```

```
# chart type
  pq_plot(ssec, chart_type = 'line',   date_range = '6m') # line chart (default)
# pq_plot(ssec, chart_type = 'step',   date_range = '6m') # step line
# pq_plot(ssec, chart_type = 'candle', date_range = '6m') # candlestick
# pq_plot(ssec, chart_type = 'bar',    date_range = '6m') # bar chart

# add technical indicators
pq_plot(ssec, chart_type = 'line', addti = list(
        sma = list(n = 200),
        sma = list(n = 50),
        macd = list()
))
# linear trend with yaxis in log
pq_plot(ssec, chart_type = 'line', linear_trend = c(-0.8, 0, 0.8), yaxis_log = TRUE)


# multiple symbols
# download datasets
# dat = md_stock(c('FB', 'AMZN', 'AAPL', 'NFLX', 'GOOG'), date_range = 'max')
dat = md_stock(c('^000001', '^399001', '^399006', '^000016', '^000300', '^000905'),
               date_range = 'max', source='163')

# linear trend
pq_plot(dat, multi_series=list(nrow=2, scales='free_y'), linear_trend=c(-0.8, 0, 0.8))
pq_plot(dat, multi_series=list(nrow=2, scales='free_y'), linear_trend=c(-0.8, 0, 0.8),
        yaxis_log=TRUE)

# performance
pq_plot(dat, multi_series = list(nrow=2), perf=TRUE, date_range = 'ytd')
pq_plot(dat, multi_series = list(nrow=1, ncol=1), perf=TRUE, date_range = 'ytd')
```

`pq_return` *calculating returns by frequency*

Description

`pq_return` calculates returns for daily series based on specified column, frequency and method type.

Usage

```
pq_return(dt, x = "close|value", method = "arithmetic", freq = "all",
  date_range = "max", from = NULL, to = Sys.Date(), print_step = 1L)
```

Arguments

dt	a list/dataframe of daily series dataset
x	the variable used to calculate returns.
method	the method to calculate returns.
freq	the frequency of returns. It supports c('all', 'daily', 'weekly', 'monthly', 'quarterly', 'yearly').
date_range	date range. Available value includes '1m'-'11m', 'ytd', 'max' and '1y'-'ny'. Default is max.
from	the start date. Default is NULL. If it is NULL, then calculate using date_range and end date.
to	the end date. Default is the current date.
print_step	a non-negative integer. Print symbol name by each print_step iteration. Default is 1L.

Examples

```
dts = md_stock(c('000001', '^000001'), source = '163')

# set freq
dts_returns1 = pq_return(dts, freq = 'all')
dts_returns2 = pq_return(dts, freq = 'weekly')

# set method
dts_returns3 = pq_return(dts, freq = 'monthly', method = 'arithmetic') # default method
dts_returns4 = pq_return(dts, freq = 'monthly', method = 'log')
```

---

pq_to_freq *converting frequency of daily data*

Description

pq_to_freq convert a daily OHLC dataframe into a specified frequency.

Usage

```
pq_to_freq(dt, freq, print_step = 1L)
```

Arguments

dt	a list/dataframe of time series dataset.
freq	the frequency that the input daily data will converted to. It supports weekly, monthly, quarterly and yearly.
print_step	A non-negative integer. Print symbol name by each print_step iteration. Default is 1L.

Examples

```
dts = md_stock(c("^000001", "000001"), date_range = 'max', source = '163')

dts_weekly = pq_to_freq(dts, "weekly")
```

`ssec` *dataset of shanghai composite index*

Description

The daily historical Shanghai Composite Index from the beginning of the Index to the end of July 2020

Usage

```
ssec
```

Format

A data frame with 7240 rows and 15 variables:

symbol stock ticker symbol

name stock ticker name

date trade date

open stock price at the open of trading

high stock price at the highest point during trading

low stock price at the lowest point during trading

close stock price at the close of trading

prev_close stock price at the close of previous trading day

change_pct change percentage of stock close price

volume number of shares traded

amount monetary value of shares traded

turnover rate of shares traded over total

cap_market tradable market capitalisation

cap_total total market capitalisation

unit price unit, such as in CNY/USD

Index