Package 'pedquant'

February 7, 2022

Version 0.1.8

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, 163, Sina, Eastmoney and etc.

Depends R (>= 3.1.0)

Imports data.table, TTR, zoo, PerformanceAnalytics, curl, xml2, httr, rvest, webdriver, 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 7.1.2

Encoding UTF-8

NeedsCompilation no

Author Shichen Xie [aut, cre]

Maintainer Shichen Xie <xie@shichen.name>

Repository CRAN

Date/Publication 2022-02-07 08:10:02 UTC

R topics documented:

`dt_banks` *dataset of bank stocks in sse*

Description

The daily historical data of bank stocks

Usage

```
dt_banks
```

Format

A data frame with 7506 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

close_prev 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

`dt_ssec` *dataset of shanghai composite index*

Description

The daily historical Shanghai Composite Index

Usage

```
dt_ssec
```

Format

A data frame with 7506 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

close_prev 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

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_bond *query bond data*

Description

md_bond query bond market data from FRED and ChinaBond.

Usage

```
md_bond(symbol = NULL, type = "history", date_range = "3y",
  from = NULL, to = Sys.Date(), print_step = 1L, ...)
```

Arguments

symbol	bond symbols. Default is NULL.
type	the data type. Default is history.
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.
...	Additional parameters.

---

md_forex *query forex data*

Description

md_forex query forex market data from FRED (history data) or sina (real data).

Usage

```
md_forex(symbol = NULL, type = "history", date_range = "3y",
  from = NULL, to = Sys.Date(), print_step = 1L, ...)
```

Arguments

symbol	forex symbols. Default is NULL.
type	the data type, available values including history and real. Default is history.
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.
...	Additional parameters.

Examples

```
## Not run:
# history data
dtfx_hist1 = md_forex(c('usdcny', 'usdjpy'))

# real data
dtfx_real = md_forex(c('eurusd', 'usdcny', 'usdjpy'), type = 'real')

# interactivly choose symbols
dtfx_hist2 = md_forex()

## End(Not run)
```

`md_future` *query future market data*

Description

`md_future` query future market data from sina finance, https://finance.sina.com.cn/futuremarket/.

Usage

```
md_future(symbol, type = "history", date_range = "max", from = NULL,
  to = Sys.Date(), freq = "daily", print_step = 1L, ...)
```

Arguments

`symbol`	future symbols It is available via function `md_future_symbol` or its website.
`type`	the data type, including history, real and info. Default is history.
`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.
`freq`	data frequency, default is daily.
`print_step`	a non-negative integer, which will print symbol name by each print_step iteration. Default is 1L.
`...`	Additional parameters.

Examples

```
## Not run:
# history data
df_hist = md_future(symbol = c('IF0', 'A0', 'CU0', 'CF0', 'XAU'))

# real data
df_real = md_future(symbol = c('IF0', 'A0', 'CU0', 'CF0', 'XAU'),
                    type = 'real')

## End(Not run)
```

---

md_future_symbol *symbol of future market data*

Description

md_future_symbol returns all future symbols that provided by sina finance, see details on http://vip.stock.finance.sina.com.cn/quotes_service/view/qihuohangqing.html or http://vip.stock.finance.sina.com.cn/mkt/#global_qh)

Usage

```
md_future_symbol()
```

Examples

```
## Not run:
sybs = md_future_symbol()

## End(Not run)
```

---

md_money *query interbank offerd rate*

Description

md_money query libor from FRED or shibor from chinamoney.

Usage

```
md_money(symbol = NULL, type = "history", date_range = "3y",
  from = NULL, to = Sys.Date(), print_step = 1L)
```

Arguments

symbol	ibor symbols. Default is NULL.
type	the data type. Default is history.
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.

---

md_stock *query stock market data*

Description

md_stock provides an interface to query stock or fund data from 163 for SSE and SZSE shares, from eastmoney for HKEX and US shares.

Usage

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

Arguments

symbol	symbols of stock shares.
type	the data type, including history, adjfactor, real and info. Default is history.
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	whether to adjust the OHLC prices, defaults to NULL. If it is NULL, return the original data; if it is FALSE, create a close_adj column if not exist; if it is TRUE, adjust all open, high, low, close columns. For the yahoo data, the adjustment is based on the close_adj; for the 163 data, the adjustment is based on the cumulative products of close/close_prev.
freq	data frequency, default is daily. .
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 data
# us
FAANG = md_stock(c('FB', 'AMZN', 'AAPL', 'NFLX', 'GOOG'))

# hkex
TMX = md_stock(c('00700.hk', '03690.hk', '01810.hk'))

# sse/szse
## the symbol without suffix
dt_cn1 = md_stock(c("000001", "^000001", "512510"))
## the symbol with suffix
dt_cn2 = md_stock(c("000001.sz", "000001.ss", '512510.ss'))


# Example II: price adjust factors
# adjust factors, splits and dividend
dt_adj = md_stock(symbol=c("000001", "^000001"), type='adjfactor', date_range='max')


# Example III: query real prices
# real price for equities


# real prices of all A shares in sse and szse
dt_real2 = md_stock(symbol='a', type='real')
# real prices of all A/B shares and index in sse and szse
dt_real3 = md_stock(symbol=c('a', 'b', 'index'), type='real')

# show real prices and sector/industry
dt_real4 = md_stock(symbol = c('a', 'b', 'index', 'fund'),
  type = 'real', show_tags = TRUE)


# Example IV:
# valuation ratios (pe, pb, ps) for shares in sse and szse
dt_valuation = md_stock(symbol=c('600000', '000001', '^000001', '^399001'),
               valuation = TRUE)


# query company information (profile/ipo), revenue and staff
dt_info1 = md_stock('600036', type = 'info')
# query history revenue
dt_info2 = md_stock('600036', type = 'info', rev_hist = 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, adjust = FALSE, source = NULL, ...)
```

Arguments

`dt`	a list/dataframe of time series datasets that didnt adjust for split or dividend.
`adjust`	whether to adjust the OHLC prices, defaults to FALSE. If it is NULL, return the original data; if it is FALSE, create close_adj or change_pct column if not exist; if it is TRUE, adjust all open, high, low, close columns. For the yahoo data, the adjustment is based on the close_adj; for the 163 data, the adjustment is based on the cumulative products of close/close_prev.
`source`	the available data sources are 'yahoo' and '163'. The source will set to yahoo, if the dt has close_adj column; and will set to 163, if the dt has close_prev column.
`...`	Additional parameters.

Examples

```
dt = md_stock('600547', source = '163', date_range = 'max')

dtadj = md_stock_adjust(dt, source = '163')
```

`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*

Description

`md_stock_symbol` returns all stock symbols by exchange

Usage

```
md_stock_symbol(exchange = NULL)
```

Arguments

`exchange`	the available stock exchanges are sse, szse, hkex, amex, nasdaq, nyse.

Examples

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

## choose exchanges interactivly
ex_syb2 = md_stock_symbol()


## End(Not run)
```

md_symbol *symbol of market data by category*

Description

md_stock_symbol returns all symbols by market category, including forex, money, bond, stock, future.

Usage

```
md_symbol(cate = NULL, ...)
```

Arguments

cate	the market category, including forex, money, bond, stock, future. Default is NULL.
...	ignored parameters

Examples

```
## Not run:
syblst = md_symbol()

## End(Not run)
```

pq_addti *adding technical indicators*

Description

pq_addti creates technical indicators using the functions provided in 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('dt_ssec')

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

# only technical indicators
dt_ti2 = pq_addti(
  dt_ssec, sma=list(n=20), sma=list(n=50), macd = list(),
  col_kp = c('symbol', 'name')
)

dt_ti3 = pq_addti(
  dt_ssec, sma=list(n=20), sma=list(n=50), macd = list(),
  col_kp = NULL
)

# 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(dt_ssec, bias = list(n = 200))
```

pq_addti_funs *technical functions*

Description

Technical functions provided in TTR package.

Usage

```
pq_addti_funs()
```

pq_freq *converting frequency of daily data*

Description

pq_freq convert a daily OHLC dataframe into a specified frequency.

Usage

```
pq_freq(dt, freq = "monthly", date_type = "eop")
```

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.
date_type	the available date type are eop (end of period) and bop (bebinning of period), defaults to the eop.

Examples

```
data(dt_ssec)
dat1_weekly = pq_freq(dt_ssec, "weekly")

data(dt_banks)
dat2_weekly = pq_freq(dt_banks, "monthly")
```

pq_performance *calculating performance metrics*

Description

pq_performance calculates performance metrics based on returns of market price or portfolio. The performance analysis functions are calling from PerformanceAnalytics package, which includes many widely used performance metrics.

Usage

```
pq_performance(dt, Ra, Rb = NULL, perf_fun, ...)
```

Arguments

dt	a list/dataframe of time series datasets.
Ra	the column name of asset returns.
Rb	the column name of baseline returns, defaults to NULL.
perf_fun	performance function from PerformanceAnalytics package, see pq_perf_funs.
...	additional parameters, the arguments used in PerformanceAnalytics functions.

Examples

```
library(pedquant)
library(data.table)

# load data
data(dt_banks)
data(dt_ssec)

# calculate returns
datret1 = pq_return(dt_banks, 'close', freq = 'monthly', rcol_name = 'Ra')
datret2 = pq_return(dt_ssec, 'close', freq = 'monthly', rcol_name = 'Rb')

# merge returns of assets and baseline
datRaRb = merge(
    rbindlist(datret1)[, .(date, symbol, Ra)],
    rbindlist(datret2)[, .(date, Rb)],
    by = 'date', all.x = TRUE
)

# claculate table.CAPM metrics
perf_capm = pq_performance(datRaRb, Ra = 'Ra', Rb = 'Rb', perf_fun = 'table.CAPM')
rbindlist(perf_capm, idcol = 'symbol')
```

pq_performance_funs *performance functions*

Description

A complete list of performance functions from `PerformanceAnalytics` package.

Usage

```
pq_performance_funs()
```

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", date_range = "max", from = NULL,
  to = Sys.Date(), x = "close|value", addti = NULL,
  linear_trend = NULL, cumreturns = FALSE, freq = "daily",
  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.
`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.
`cumreturns`	logical, display the cumulative returns. Default is FALSE.
`freq`	the data frequency. It supports c('daily', 'weekly', 'monthly', 'quarterly', 'yearly').

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
data(dt_ssec)
# dt_ssec = md_stock('^000001', source='163', date_range = 'max')

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

# add technical indicators
pq_plot(dt_ssec, chart_type = 'line', addti = list(
        sma = list(n = 200),
        sma = list(n = 50),
        macd = list()
))
# linear trend with yaxis in log
pq_plot(dt_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')

data(dt_banks)
dat = md_stock_adjust(dt_banks, adjust = TRUE)

# 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, x='close', multi_series = list(nrow=2), cumreturns=TRUE, date_range = 'ytd')
```

```
pq_plot(dat, x='close', multi_series = list(nrow=1, ncol=1), cumreturns=TRUE, date_range = 'ytd')
```

`pq_portfolio` *calculating returns/equity of portfolio*

Description

`pq_portfolio` calculates the weighted returns or the equity of a portfolio assets.

Usage

```
pq_portfolio(dt, dtv, x, v = "volume", init_fund = NULL,
  method = "arithmetic", ...)
```

Arguments

`dt`	a list/dataframe of price by asset.
`dtv`	a dataframe of transaction volume by asset.
`x`	the column name of adjusted asset price.
`v`	the column name of asset volume, defaults to volume.
`init_fund`	initial fund value.
`method`	the method to calculate asset returns, the available values include arithmetic and log, defaults to arithmetic.
`...`	ignored

Examples

```
library(pedquant)
library(data.table)

data(dt_banks)
datadj = md_stock_adjust(dt_banks, adjust = FALSE)

# example I
dtv = data.table(
    symbol = c("601288.SS","601328.SS","601398.SS","601939.SS","601988.SS"),
    volume = c(100, 200, 300, 300, 100)
)
dtRa = pq_portfolio(datadj, x='close_adj', dtv=dtv)
pq_plot(dtRa, x = 'cumreturns')

dtRb = pq_return(dt_ssec, x = 'close', freq = 'daily', cumreturns = TRUE)
pq_plot(list(Ra = dtRa, Rb = dtRb$`000001.SS`), x = 'cumreturns',
        multi_series = list(nrow=1, ncol=1))
```

```
# example II
dtv = data.table(
    symbol = rep(c("601288.SS","601328.SS","601398.SS","601939.SS","601988.SS"), 3),
    date = rep(c('2009-03-02', '2010-01-04', '2014-09-01'), each = 5),
    volume = rep(c(100, 200, 300, 300, 100), 3) * rep(c(1, -1, 2), each = 5)
)
dtRa2 = pq_portfolio(datadj, x='close_adj', dtv=dtv, init_fund = 10000)
pq_plot(dtRa2, x = 'balance',
        addti = list(equity = list(), fund = list()))
```

---

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, freq = "monthly", num = 1, date_type = "eop",
  method = "arithmetic", leading = TRUE, cumreturns = FALSE,
  rcol_name = "returns", cols_keep = NULL, date_range = "max",
  from = NULL, to = Sys.Date())
```

Arguments

dt	a list/dataframe of daily series dataset
x	the column name of adjusted asset price.
freq	the frequency of returns. It supports c('all', 'daily', 'weekly', 'monthly', 'quarterly', 'yearly').
num	the number of preceding periods used as the base value, defaults to 1, which means based on the previous period value.
date_type	the available date type are eop (end of period) and bop (beginning of period), defaults to the eop.
method	the method to calculate asset returns, the available methods including arithmetic and log, defaults to arithmetic.
leading	whether to return the incomplete leading period returns.
cumreturns	logical, whether to return cumulative returns. Defaults to FALSE.
rcol_name	setting the column name of returns, defaults to returns.
cols_keep	the columns keep in the return data. The columns of symbol, name and date will always kept if they are exist in the input data.

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.

Examples

```
data(dt_banks)

# create a close_adj column
datadj = md_stock_adjust(dt_banks, adjust = FALSE)

# set freq
dts_returns1 = pq_return(datadj, x = 'close_adj', freq = 'all')

# set method
dts_returns2 = pq_return(datadj, x = 'close_adj', method = 'log')

# set cols_keep
dts_returns3 = pq_return(datadj, x = 'close_adj', cols_keep = 'cap_total')
```

Index