Package 'RTL'

February 26, 2025

Type Package

Title Risk Tool Library - Trading, Risk, Analytics for Commodities

Version 1.3.7

Date 2025-02-25

Description A toolkit for Commodities 'analytics', risk management and trading professionals. Includes functions for API calls to <https://commodities.morningstar.com/#/>, <https://developer.genscape.com/>, and <https://www.bankofcanada.ca/valet/docs>.

License MIT + file LICENSE

URL https://github.com/risktoollib/RTL

Depends R (>= 4.0)

Imports dplyr, ggplot2, httr, jsonlite, lubridate, magrittr, plotly, purrr, readr, rlang, stringr, tibble, tidyr, timetk, tsibble, xts, zoo, glue, Rcpp, lifecycle, TTR, tidyselect, PerformanceAnalytics, numDeriv

Suggests testthat (>= 3.0.0), covr, lpSolve, rugarch, tidyquant, feasts, fabletools, MASS, sf

Encoding UTF-8

LazyData true

LazyDataCompression xz

RoxygenNote 7.3.2

Config/testthat/edition 3

LinkingTo Rcpp

BugReports https://github.com/risktoollib/RTL/issues

NeedsCompilation yes

Author Philippe Cote [aut, cre], Nima Safaian [aut]

Maintainer Philippe Cote <pcote@ualberta.ca>

Repository CRAN

Date/Publication 2025-02-26 03:30:02 UTC

Contents

`barrierSpreadOption` *Barrier Spread Option Pricing*

Description

This model applies Kirk's (1995) closed-form approximation for pricing spread options, incorporating barrier adjustments for continuous and terminal monitoring.

Usage

```
barrierSpreadOption(
  F1 = -12,
  F2 = -3,
  X = 5.5,
  B = 9,
  sigma1 = 0.6,
  sigma2 = 0.6,
  rho = 0.3,
  T2M = 1/12,
  r = 0.045,
  type = "call",
```

```
  barrier_type = "uo",
  monitoring = "continuous"
)
```

Arguments

F1	numeric, forward price of the first asset (e.g., pipeline origination price as a futures)
F2	numeric, forward price of the second asset (e.g., pipeline destination price as a futures)
X	numeric, strike price of the spread option
B	numeric, barrier level for the spread (F2 - F1)
sigma1	numeric, volatility of the first asset (annualized)
sigma2	numeric, volatility of the second asset (annualized)
rho	numeric, correlation coefficient between the two assets
T2M	numeric, time to maturity in years
r	numeric, risk-free interest rate (annualized)
type	character, "call" or "put"
barrier_type	character, "do" or "uo" (down-and-out, up-and-out)
monitoring	character, "continuous" or "terminal"

Value

A list containing option price and Greeks.

---

bond *Bond pricing*

Description

Compute bond price, cash flow table or duration

Usage

```
bond(ytm = 0.05, C = 0.05, T2M = 1, m = 2, output = "price")
```

Arguments

ytm	Yield to Maturity. numeric
C	Coupon rate per annum. numeric
T2M	Time to maturity in years. numeric
m	Periods per year for coupon payments e.g semi-annual = 2. numeric
output	"price", "df" or "duration". character

Value

Returns price numeric, cash flows tibble, or duration numeric

Author(s)

Philippe Cote

Examples

```
bond(ytm = 0.05, C = 0.05, T2M = 1, m = 2, output = "price")
bond(ytm = 0.05, C = 0.05, T2M = 1, m = 2, output = "df")
bond(ytm = 0.05, C = 0.05, T2M = 1, m = 2, output = "duration")
```

---

chart_eia_sd *EIA weekly supply-demand information by product group*

Description

Given a product group extracts all information to create SD Balances.

Usage

```
chart_eia_sd(
  market = "mogas",
  key = "your EIA.gov API key",
  from = "2011-01-01",
  legend.pos = list(x = 0.4, y = 0.53),
  output = "chart"
)
```

Arguments

market	"mogas", "dist", "jet" or "resid". character
key	Your private EIA API token. character
from	Date as character "2020-07-01". Default to all dates available. character
legend.pos	Defaults to list(x = 0.4, y = 0.53). list
output	"chart" for plotly object or "data" for dataframe.

Value

A plotly chart htmlwidget or a tibble.

Author(s)

Philippe Cote

Examples

```
## Not run:
chart_eia_sd(key = key, market = "mogas")

## End(Not run)
```

`chart_eia_steo` *EIA Short Term Energy Outlook*

Description

Extract data and either plots or renders dataframe.

Usage

```
chart_eia_steo(
  market = "globalOil",
  key = "your EIA.gov API key",
  from = "2018-07-01",
  fig.title = "EIA STEO Global Liquids SD Balance",
  fig.units = "million barrels per day",
  legend.pos = list(x = 0.4, y = 0.53),
  output = "chart"
)
```

Arguments

`market`	"globalOil" only currently implemented. `character`
`key`	Your private EIA API token. `character`
`from`	Date as character "2020-07-01". Default to all dates available. `character`
`fig.title`	Defaults to "EIA STEO Global Liquids SD Balance". `character`
`fig.units`	Defaults to "million barrels per day" `character`
`legend.pos`	Defaults to list(x = 0.4, y = 0.53) `list`
`output`	"chart" for plotly object or "data" for dataframe.

Value

A plotly chart `htmlwidget` or a `tibble`.

Author(s)

Philippe Cote

Examples

```
## Not run:
chart_eia_steo(key = EIAkey, market = "globalOil")

## End(Not run)
```

`chart_fwd_curves` *Plots historical forward curves*

Description

Returns a plot of forward curves through time

Usage

```
chart_fwd_curves(df = dfwide, cmdty = "cmewti", weekly = TRUE, ...)
```

Arguments

`df`	Wide dataframe with date column and multiple series columns (multivariate). `tibble`
`cmdty`	Futures contract code in expiry_table object: unique(expiry_table$cmdty). `character`
`weekly`	Defaults to TRUE for weekly forward curves. `logical`
`...`	other graphical parameters

Value

plot of forward curves through time. `NULL`

Author(s)

Philippe Cote

Examples

```
df <- dfwide %>%
  dplyr::select(date, dplyr::starts_with("CL")) %>%
  tidyr::drop_na()
chart_fwd_curves(
  df = df, cmdty = "cmewti", weekly = TRUE,
  main = "WTI Forward Curves", ylab = "$ per bbl", xlab = "", cex = 2
)
```

chart_pairs *Pairwise scatter plots for timeseries*

Description

Plots pairwise scatter plots with the time dimension. Useful when exploring structural changes in timeseries properties for modeling.

Usage

```
chart_pairs(df = df, title = "Time Series Pairs Plot")
```

Arguments

df	Wide data frame. tibble
title	Chart title. character

Value

A plotly object. htmlwidget

Author(s)

Philippe Cote

Examples

```
df <- dfwide %>%
  dplyr::select(date, CL01, NG01, HO01, RB01) %>%
  tidyr::drop_na()
chart_pairs(df = df, title = "example")
```

chart_PerfSummary *Cumulative performance and drawdown summary.*

Description

Multi Asset Display of Cumulative Performance and Drawdowns

Usage

```
chart_PerfSummary(
  ret = ret,
  geometric = TRUE,
  main = "Cumulative Returns and Drawdowns",
  linesize = 1.25
)
```

Arguments

ret	Wide dataframe univariate or multivariate of percentage returns. tibble
geometric	Use geometric returns TRUE or FALSE. logical
main	Chart title. character
linesize	Size of lines in chart and legend. numeric

Value

Cumulative performance and drawdown charts. ggplot

Author(s)

Philippe Cote

Examples

```
ret <- data.frame(
  date = seq.Date(Sys.Date() - 60, Sys.Date(), 1),
  CL01 = rnorm(61, 0, .01), RB01 = rnorm(61, 0, 0.02)
)
chart_PerfSummary(ret = ret,
geometric = TRUE,
main = "Cumulative Returns and Drawdowns",
linesize = 1.25)
```

---

chart_spreads *Futures contract spreads comparison across years*

Description

Plots specific contract pairs across years with time being days from expiry.

Usage

```
chart_spreads(
  cpairs = cpairs,
  daysFromExpiry = 200,
  from = "2012-01-01",
  conversion = c(1, 1),
  feed = "CME_NymexFutures_EOD",
  iuser = "x@xyz.com",
  ipassword = "pass",
  title = "March/April ULSD Nymex Spreads",
  yaxis = "$ per bbl",
  output = "chart"
)
```

Arguments

cpairs	Tibble of contract pairs - see example for expiry when not expired yet. tibble
daysFromExpiry	Number of days from expiry to compute spreads. numeric
from	From date character
conversion	Defaults to c(1,1) first and second contracts. 42 from $ per gallons to bbls. numeric
feed	Morningstar Feed Table. character
iuser	Morningstar user name as character - sourced locally in examples. character
ipassword	Morningstar user password as character - sourced locally in examples. character
title	Title for chart. character
yaxis	y-axis label. character
output	"chart" for htmlwidget or "data" for tibble.

Value

A plotly object or a dataframe

Author(s)

Philippe Cote

Examples

```
## Not run:
cpairs <- dplyr::tibble(
 year = c("2018", "2019", "2020","2021","2022","2023"),
 first = c("@HO8H", "@HO9H", "@HO0H","@HO21H","@HO22H","@HO23H"),
 second = c("@CL8H", "@CL9H", "@CL0H","@CL21H","@CL22H","@CL23H"),
 expiry = c(NA,NA,NA,NA,NA,"2023-02-23")
)
chart_spreads(
  cpairs = cpairs, daysFromExpiry = 200, from = "2012-01-01",
  conversion = c(42, 1), feed = "CME_NymexFutures_EOD",
  iuser = "x@xyz.com", ipassword = "pass",
  title = "March/April ULSD Nymex Spreads",
  yaxis = "$ per bbl",
  output = "data"
)

## End(Not run)
```

chart_zscore *Z-Score applied to seasonal data divergence*

Description

Supports analytics and display of seasonal data. Z-Score is computed on residuals conditional on their seasonal period. Beware that most seasonal charts in industry e.g. (NG Storage) is not detrended so results once you apply an STL decomposition will vary from the unajusted seasonal plot.

Usage

```
chart_zscore(
  df = df,
  title = "NG Storage Z Score",
  per = "yearweek",
  output = "zscore",
  chart = "seasons"
)
```

Arguments

df	Long data frame with columns series, date and value. tibble
title	Default is a blank space returning the unique value in df$series. character
per	Frequency of seasonality "yearweek" (DEFAULT). "yearmonth", "yearquarter" character
output	"stl" for STL decomposition chart, "stats" for STL fitted statistics. "res" for STL fitted data. "zscore" for residuals Z-score, "seasonal" for standard seasonal chart.
chart	"seasons" for feasts::gg_season() (DEFAULT) "series" for feasts::gg_subseries()

Value

Time series of STL decomposition residuals Z-Scores, or standard seasonal chart with feast package.

Author(s)

Philippe Cote

Examples

```
## Not run:
df <- eiaStocks %>% dplyr::filter(series == "NGLower48")
title <- "NGLower48"
chart_zscore(df = df, title = " ", per = "yearweek", output = "stl", chart = "seasons")
chart_zscore(df = df, title = " ", per = "yearweek", output = "stats", chart = "seasons")
```

```
chart_zscore(df = df, title = " ", per = "yearweek", output = "res", chart = "seasons")
chart_zscore(df = df, title = " ", per = "yearweek", output = "zscore", chart = "seasons")
chart_zscore(df = df, title = " ", per = "yearweek", output = "seasonal", chart = "seasons")

## End(Not run)
```

---

cma *metadata for WTI CMA*

Description

CME WTI Calendar Month Average swap information

Usage

```
cma
```

Format

data frame

Value

tibble

Source

cme

CRReuro *Cox-Ross-Rubinstein binomial option model*

Description

European option binomial model on a stock without dividends.For academic purpose only. Use RTL::CRRoption for real-life usage.

Usage

```
CRReuro(S, X, sigma, r, T2M, N, type)
```

Arguments

S	Stock price. numeric
X	Strike price. numeric
sigma	Implied volatility e.g. 0.20 numeric
r	Risk-free rate. numeric
T2M	Time to maturity in years numeric
N	Number of time steps. Internally dt = T2M/N. numeric
type	"call" or "put" character

Value

List of asset price tree, option value tree and option price. list

Author(s)

Philippe Cote

Examples

```
CRReuro(S = 100, X = 100, sigma = 0.2, r = 0.1, T2M = 1, N = 5, type = "call")
```

CRROption *Cox-Ross-Rubinstein Option Pricing Model*

Description

Computes the price of European and American options using the Cox-Ross-Rubinstein binomial model. This function is optimized for performance and implemented in C++. Haug (2007) provides a detailed description of the model.

Usage

```
CRROption(S, X, sigma, r, b, T2M, N, type, optionStyle)
```

Arguments

S	Numeric, the current stock price (also known as the underlying asset price).
X	Numeric, the strike price of the option.
sigma	Numeric, the implied volatility of the underlying stock (annualized).
r	Numeric, the risk-free interest rate (annualized).
b	Numeric, the cost of carry, b = r - q for dividend paying assets, where q is the dividend yield rate.
T2M	Numeric, the time to maturity of the option (in years).
N	Integer, the number of time steps in the binomial tree.
type	Character, the type of option ("call" or "put").
optionStyle	Character, the style of the option ("european" or "american").

Value

A list containing the computed price of the option and a note indicating if the model is suitable for the provided parameters.

Examples

```
# CRROption(S = 100, X = 100, sigma = 0.25, r = 0.1, b = 0, T2M = 1, N = 500,
# type = "call", optionStyle = "european")
# CRROption(S = 100, X = 100, sigma = 0.25, r = 0.1, b = 0, T2M = 1, N = 500,
# type = "call", optionStyle = "american")
```

---

crudeOil *dataset: crude assays*

Description

crude assays

Usage

```
crudeOil
```

Format

list

Value

list

---

cushing *dataset: WTI Cushing Futures and storage utilization*

Description

c1, c2, c1c2 and Cushing storage utilization

Usage

```
cushing
```

Format

list

Value

list

Source

CME and EIA

---

dflong *dataset: commodity prices in a long dataframe format*

Description

Futures settlement data set.

Usage

```
dflong
```

Format

data frame

Value

tibble

Source

Morningstar Commodities

---

dfwide *dataset: commodity prices in a wide dataframe format*

Description

Futures settlement data set.

Usage

```
dfwide
```

Format

data frame

Value

tibble

Source

Morningstar Commodities

---

efficientFrontier *Markowitz Efficient Frontier*

Description

Generates random portfolio weights statistics based on absolute returns.

Usage

```
efficientFrontier(
  nsims = 5000,
  x = RTL::fizdiffs %>% dplyr::select(date, dplyr::contains("WCS")),
  expectedReturns = NULL
)
```

Arguments

nsims	Number of portfolio simulations. Defaults to 5000 numeric
x	List as provided by output of RTL::simMultivariates(). list
expectedReturns	Defaults to NULL using periodic returns means. numeric

Details

Commodities:

Unlike traditional portfolio management, in commodities many transactions are with derivatives (futures and swaps) and have zero or low initial investments.

Return types:

This function is used for commodities where returns are dollars per units for real assets e.g. storage tanks, pipelines...Here we measure directly the periodic return in dollars per contract unit.

Empirical Finance:

I would encourage you to pick a commodity futures contract of your choice and draw a scatter plot of price level versus the daily dollar per unit change as measure of risk. As a trading analyst or risk manager, then ask yourself about the implications of using log returns that you then re-apply to current forward curve level to arrive at a dollar risk measure per units instead of measuring directly risk in dollars per unit.

Value

List of portfolios and chart of efficient frontier `list`

Author(s)

Philippe Cote

Examples

```
## Not run:
x =  RTL::fizdiffs %>% dplyr::select(date, dplyr::contains("WCS"))
efficientFrontier(nsims = 10, x = x, expectedReturns = NULL)
efficientFrontier(nsims = 10, x = x, expectedReturns = c(0.5,0.8,0.9))

## End(Not run)
```

---

eia2tidy *EIA API call with tidy output*

Description

Extracts data from the Energy Information Administration (EIA) API to tibble format with optional custom series name. Makes a clean wrapper for use with purrr for multiple series extraction. Query Browser at https://www.eia.gov/opendata/qb.php.

Usage

```
eia2tidy(ticker, key, name = " ")
```

Arguments

`ticker`	EIA series name. `character`
`key`	Your private EIA API token as character "yourapikey". `character`
`name`	Name you want to give the series. Defaults to ticker if set to " " `character`

Value

A tibble object with class date for weekly, monthly, quarterly or annual data and class POSIXct for hourly. `tibble`

Author(s)

Philippe Cote

Examples

```
## Not run:
# Single Series
RTL::eia2tidy(ticker = "PET.MCRFPTX2.M", key = "yourapikey", name = "TexasProd")
# Multiple Series
# Use eia2tidy_all() or pivot_longer, drop_na and then pivot_wider to wrangled results.

## End(Not run)
```

`eia2tidy_all` *EIA API multiple calls with tidy output*

Description

Extracts data from the Energy Information Administration (EIA) API to tibble format with optional custom series name. Makes a clean wrapper for use with purrr for multiple series extraction. Query Browser at https://www.eia.gov/opendata/qb.php.

Usage

```
eia2tidy_all(
  tickers = tibble::tribble(~ticker, ~name, "PET.W_EPC0_SAX_YCUOK_MBBL.W",
    "CrudeCushing", "NG.NW2_EPG0_SWO_R48_BCF.W", "NGLower48"),
  key,
  long = TRUE
)
```

Arguments

`tickers`	tribble of EIA series and names you want to assign. `character`
`key`	Your private EIA API token as character "yourapikey". `character`
`long`	TRUE (default) to return a long data frame or FASLE for wide. `logical`

Value

A tibble object with class date for weekly, monthly, quarterly or annual data and class POSIXct for hourly. `tibble`

Author(s)

Philippe Cote

Examples

```
## Not run:
eia2tidy_all(tickers = tibble::tribble(~ticker, ~name,
                          "PET.W_EPC0_SAX_YCUOK_MBBL.W", "CrudeCushing",
                          "NG.NW2_EPG0_SWO_R48_BCF.W", "NGLower48"),
             key = "your API key", long = TRUE)

## End(Not run)
```

---

eiaStocks *dataset: EIA weekly stocks*

Description

EIA weekly crude, NG, ULSD and RBOB stocks.

Usage

```
eiaStocks
```

Format

data frame

Value

tibble

---

eiaStorageCap *dataset: EIA working storage capacity*

Description

EIA working storage capacity in kbs except NG in bcf.

Usage

```
eiaStorageCap
```

Format

data frame

Value

tibble

eurodollar *dataset: Eurodollar futures contracts*

Description

ED futures contract for December 2024

Usage

```
eurodollar
```

Format

data frame

Value

tibble

Source

Morningstar

expiry_table *dataset: expiry of common commodity futures contract.*

Description

This dataframe provides detailed information on major futures contracts specifications pertaining to last settlement, notices and delivery dates. It also provides tickers in some data service.

Usage

```
expiry_table
```

Format

data frame

Value

tibble

fitOU — *Fits a Ornstein–Uhlenbeck process to a dataset*

Description

Parameter estimation for Ornstein–Uhlenbeck process using OLS

Usage

```
fitOU(spread, dt = 1/252)
```

Arguments

spread	Spread time series. tibble
dt	Time step size in fractions of a year. Default is 1/252.

Value

List of theta, mu, annualized sigma estimates. It returns half life consistent with periodicity list

Author(s)

Philippe Cote

Examples

```
spread <- simOU(nsims = 1, mu = 5, theta = .5, sigma = 0.2, T = 5, dt = 1 / 252)
fitOU(spread = spread$sim1)
```

fizdiffs — *dataset: randomised physical crude differentials*

Description

Randomized data set for education purpose of selected physical crude differentials to WTI.

Usage

```
fizdiffs
```

Format

data frame

Value

tibble

futuresRef *dataset: futures contracts metadata*

Description

Exchange-traded contract month codes and specifications.

Usage

```
futuresRef
```

Format

data frame

Value

tibble

fxfwd *dataset: USDCAD FX forward rates*

Description

USDCAD historicals and forward curve

Usage

```
fxfwd
```

Format

list

Value

list

Source

Morningstar and https://ca.investing.com/rates-bonds/forward-rates

garch *Wrapper for a Garch(1,1) returning either a plot or data.*

Description

Computes annualised Garch(1,1) volatilities using fGarch package.

Usage

```
garch(x = x, out = TRUE)
```

Arguments

x	Wide dataframe with date column and single series (univariate). tibble
out	"chart" to return replot_xts chart, "data" to return xts data or "fit" for uGARCHfit fit output

Value

replot_xts chart, xts data, or uGARCHfit fit

Author(s)

Philippe Cote

Examples

```
## Not run:
x <- dflong %>% dplyr::filter(series == "CL01")
x <- returns(df = x, retType = "rel", period.return = 1, spread = TRUE)
x <- rolladjust(x = x, commodityname = c("cmewti"), rolltype = c("Last.Trade"))
summary(garch(x = x, out = "fit"))
garch(x = x, out = "chart")
garch(x = x, out = "data")

## End(Not run)
```

GBSOption *Generalized Black-Scholes (GBS) Option Pricing Model*

Description

Computes the price and Greeks of European call and put options using the Generalized Black-Scholes model.

Usage

```
GBSOption(S, X, T2M, r, b, sigma, type = "call")
```

Arguments

S	numeric, the current stock price (also known as the underlying asset price).
X	numeric, the strike price of the option.
T2M	numeric, the time to maturity (in years). Previously denoted as T.
r	numeric, the risk-free interest rate (annualized).
b	numeric, the cost of carry, b = r - q for dividend paying assets, where q is the dividend yield rate.
sigma	numeric, the volatility of the underlying asset (annualized).
type	character, the type of option to evaluate, either "call" or "put". Default is "call".

Value

A list containing the following elements:

- price: The price of the option.
- delta: The sensitivity of the option's price to a change in the price of the underlying asset.
- gamma: The rate of change in the delta with respect to changes in the underlying price.
- vega: The sensitivity of the option's price to the volatility of the underlying asset.
- theta: The sensitivity of the option's price to the passage of time.
- rho: The sensitivity of the option's price to the interest rate.

Examples

```
GBSOption(S = 100, X = 100, T2M = 1, r = 0.05, b = 0.02, sigma = 0.2, type = "call")
```

---

getBoC *Bank of Canada Valet API*

Description

Extracts series from the Bank of Canada's Valet API. API documentation at https://www.bankofcanada.ca/valet/docs.

Usage

```
getBoC(series)
```

Arguments

series	Array of series name: c("FXCADUSD","BD.CDN.2YR.DQ.YLD"). character

Value

A long data frame. tibble

Author(s)

Philippe Cote

Examples

```
RTL::getBoC(series = c("FXCADUSD","BD.CDN.2YR.DQ.YLD"))
```

---

getCurve *Morningstar Commodities API forward curves*

Description

Returns forward curves from Morningstar API. See below for current feeds supported. You need your own credentials with Morningstar.

Usage

```
getCurve(
  feed = "CME_NymexFutures_EOD_continuous",
  contract = "CL",
  numOfcontracts = 12,
  date = "2023-08-24",
  fields =
   c("open_price, high_price, low_price, settlement_price, volume, open_interest"),
  iuser = "x@xyz.com",
  ipassword = "pass"
)
```

Arguments

feed	Morningstar Feed Table e.g "Crb_Futures_Price_Volume_And_Open_Interest". character
contract	Morningstar contract root e.g. "CL" for CME WTI and "BG" for ICE Brent. character
numOfcontracts	Number of listed contracts to retrieve. numeric
date	Date yyyy-mm-dd. character
fields	Defaults to c("open_price, high_price, low_price, settlement_price, volume, open_interest"). character
iuser	Morningstar user name as character - sourced locally in examples. character
ipassword	Morningstar user password as character - sourced locally in examples. character

Value

wide data frame. tibble

Current Feeds Supported

- CME_NymexFutures_EOD_continuous

Author(s)

Philippe Cote

Examples

```
## Not run:
# CME WTI Futures
getCurve(
  feed = "CME_NymexFutures_EOD_continuous", contract = "CL",
  date = "2023-08-24",
 fields = c("open_price, high_price, low_price, settlement_price, volume, open_interest"),
  iuser = "x@xyz.com", ipassword = "pass"
)

## End(Not run)
```

---

getGenscapePipeOil *Genscape API call for oil pipelines*

Description

Returns oil pipeline flows in barrels per day data from Genscape API.You need your own credentials. Refer to API documentation for argument values. It is assumed if you use this function that you know the pipelines you need to extract to build supply demand balances. Use the online API to identify the pipeline IDs. https://developer.genscape.com/docs/services/oil-transportation/operations/GetPipelineFlowValues

Usage

```
getGenscapePipeOil(
  frequency = "daily",
  regions = "Canada",
  pipelineIDs = c(97),
  revision = "revised",
  limit = 5000,
  offset = 0,
  startDate = "2015-01-01",
  endDate = as.character(Sys.Date()),
  apikey = "yourapikey"
)
```

Arguments

frequency	"daily" DEFAULT. character
regions	See API webpage. Multiple values separated by commas e.g. "Canada", "Gulf-Coast"). character
pipelineIDs	See API webpage. c(98,54...) for specific pipes. numeric
revision	See API webpage. character
limit	See API webpage. Max 5000. numeric
offset	See API webpage. numeric
startDate	"yyyy-mm-dd". character
endDate	"yyyy-mm-dd". character
apikey	Your API key. character

Value

wide data frame. tibble

Author(s)

Philippe Cote

Examples

```
## Not run:
getGenscapePipeOil(
  frequency = "daily", regions = "Canada", pipelineIDs = c(97),
  revision = "revised", limit = 5000, offset = 0,
  startDate = "2015-01-01", endDate = as.character(Sys.Date()),
  apikey = "yourapikey"
)

## End(Not run)
```

---

getGenscapeStorageOil *Genscape API call for oil storage*

Description

Returns oil storage data from Genscape API.You need your own credentials. Refer to API documentation for argument values. https://developer.genscape.com/docs/services/oil-storage/operations/StorageVolumeByOwnerGet

Usage

```
getGenscapeStorageOil(
  feed = "owner-volumes",
  regions = "Canada",
  products = "Crude",
  revision = "revised",
  limit = 5000,
  offset = 0,
  startDate = "2011-01-01",
  endDate = as.character(Sys.Date()),
  apikey = "yourapikey"
)
```

Arguments

feed	"owner-volumes" DEFAULT or "tank-volumes". character
regions	See API webpage. Multiple values separated by commas e.g. "Canada, Cushing"). character
products	See API webpage. Multiple values separated by commas e.g. "Crude, JetFuel"). character
revision	See API webpage. character
limit	See API webpage. Max 5000. numeric
offset	See API webpage. numeric
startDate	"yyyy-mm-dd". character
endDate	"yyyy-mm-dd". character
apikey	Your API key as a character string. character

Value

wide data frame tibble

Author(s)

Philippe Cote

Examples

```
## Not run:
# where yourapikey = "yourapikey".
getGenscapeStorageOil(
  feed = "owner-volumes", regions = "Canada", products = "Crude",
  revision = "revised", limit = 5000, offset = 0,
  startDate = "2011-01-01", endDate = "2020-11-01", apikey = yourapikey
)

## End(Not run)
```

getGIS *Extract and convert GIS data from a URL*

Description

Returns a SpatialPointsDataFrame from a shapefile URL. @section Examples with EIA and Government of Alberta

- from https://www.eia.gov/maps/layer_info-m.php :
- crudepipelines <- getGIS(url = "https://www.eia.gov/maps/map_data/CrudeOil_Pipelines_US_EIA.zip")
- refineries <- getGIS(url = "https://www.eia.gov/maps/map_data/Petroleum_Refineries_US_EIA.zip")
- from https://gis.energy.gov.ab.ca/Geoview/OSPNG
- AB <- getGIS(url = "https://gis.energy.gov.ab.ca/GeoviewData/OS_Agreements_Shape.zip")

Usage

```
getGIS(url = "https://www.eia.gov/maps/map_data/CrudeOil_Pipelines_US_EIA.zip")
```

Arguments

url	URL of the zipped shapefile. character

Value

SpatialPointsDataFrame. SpatialPolygonsDataFrame

Author(s)

Philippe Cote

Examples

```
## Not run:
getGIS(url = "https://www.eia.gov/maps/map_data/CrudeOil_Pipelines_US_EIA.zip")

## End(Not run)
```

`getPrice` *Morningstar Commodities API single call*

Description

Returns data from Morningstar API. See below for current feeds supported. You need your own credentials with Morningstar. In examples sourced locally.

Usage

```
getPrice(
  feed = "CME_NymexFutures_EOD",
  contract = "@CL21Z",
  from = "2020-09-01",
  iuser = "x@xyz.com",
  ipassword = "pass"
)
```

Arguments

`feed`	Morningstar Feed Table. `character`
`contract`	Morningstar key. `character`
`from`	From date yyyy-mm-dd. `character`
`iuser`	Morningstar user name as character - sourced locally in examples. `character`
`ipassword`	Morningstar user password as character - sourced locally in examples. `character`

Value

wide data frame. `tibble`

Current Feeds Supported

- CME_CbotFuturesEOD and CME_CbotFuturesEOD_continuous
- CME_NymexFutures_EOD, CME_NymexFuturesFinal_EOD andCME_NymexFutures_EOD_continuous
- CME_NymexOptionsFinal_EOD and CME_NymexOptions_EOD
- CME_CmeFutures_EOD and CME_CmeFutures_EOD_continuous
- CME_Comex_FuturesSettlement_EOD and CME_Comex_FuturesSettlement_EOD_continuous
- LME_AskBidPrices_Delayed
- SHFE_FuturesSettlement_RT
- ICE_EuroFutures and ICE_EuroFutures_continuous
- ICE_NybotCoffeeSugarCocoaFutures and ICE_NybotCoffeeSugarCocoaFutures_continuous
- CME_STLCPC_Futures

- CFTC_CommitmentsOfTradersCombined. Requires multiple keys. Separate them by a space e.g. "N10 06765A NYME 01".
- Morningstar_FX_Forwards. Requires multiple keys. Separate them by a space e.g. "USD-CAD 2M".
- ERCOT_LmpsByResourceNodeAndElectricalBus.
- PJM_Rt_Hourly_Lmp.
- AESO_ForecastAndActualPoolPrice.

Author(s)

Philippe Cote

Examples

```
## Not run:
getPrice(
  feed = "CME_NymexFuturesFinal_EOD", contract = "CL 2024 07",
  from = "2019-08-26", iuser = username, ipassword = password
)
getPrice(
  feed = "CME_NymexFutures_EOD", contract = "@CL21Z",
  from = "2019-08-26", iuser = username, ipassword = password
)
getPrice(
  feed = "CME_NymexFutures_EOD_continuous", contract = "CL_006_Month",
  from = "2019-08-26", iuser = username, ipassword = password
)
getPrice(
  feed = "CME_NymexOptionsFinal_EOD", contract = "06 2024 P LO 8000",
  from = "2020-03-15", iuser = username, ipassword = password
)
getPrice(
  feed = "CME_NymexOptions_EOD", contract = "@LO21ZP4000",
  from = "2020-03-15", iuser = username, ipassword = password
)
getPrice(
  feed = "CME_CbotFuturesEOD", contract = "C0Z",
  from = "2019-08-26", iuser = username, ipassword = password
)
getPrice(
  feed = "CME_CbotFuturesEOD_continuous", contract = "ZB_001_Month",
  from = "2019-08-26", iuser = username, ipassword = password
)
getPrice(
  feed = "CME_CmeFutures_EOD_continuous", contract = "HE_006_Month",
  from = "2019-08-26", iuser = username, ipassword = password
)
getPrice(
  feed = "Morningstar_FX_Forwards", contract = "USDCAD 2M",
  from = "2019-08-26", iuser = username, ipassword = password
)
```

```
getPrice(
  feed = "CME_CmeFutures_EOD", contract = "LH0N",
  from = "2019-08-26", iuser = username, ipassword = password
)
getPrice(
  feed = "CME_CmeFutures_EOD_continuous", contract = "HE_006_Month",
  from = "2019-08-26", iuser = username, ipassword = password
)
getPrice(
  feed = "ICE_EuroFutures", contract = "BRN0Z",
  from = "2019-08-26", iuser = username, ipassword = password
)
getPrice(
  feed = "ICE_EuroFutures_continuous", contract = "BRN_001_Month",
  from = "2019-08-26", iuser = username, ipassword = password
)
getPrice(
  feed = "ICE_NybotCoffeeSugarCocoaFutures", contract = "SB21H",
  from = "2019-08-26", iuser = username, ipassword = password
)
getPrice(
  feed = "ICE_NybotCoffeeSugarCocoaFutures_continuous", contract = "SF_001_Month",
  from = "2019-08-26", iuser = username, ipassword = password
)
getPrice(
  feed = "AESO_ForecastAndActualPoolPrice", contract = "Forecast_Pool_Price",
  from = "2021-04-01", iuser = username, ipassword = password
)
getPrice(
  feed = "LME_MonthlyDelayed_Derived", contract = "AHD 2021-12-01 2021-12-31",
  from = "2021-04-01", iuser = username, ipassword = password
)

## End(Not run)
```

`getPrices` *Morningstar Commodities API multiple calls*

Description

Multiple Morningstar API calls using getPrice functions. Refer to `getPrices()` for list of currently supported data feeds.

Usage

```
getPrices(
  feed = "CME_NymexFutures_EOD",
  contracts = c("CL9Z", "CL0F", "CL0M"),
  from = "2019-01-01",
```

```
  iuser = "x@xyz.com",
  ipassword = "pass"
)
```

Arguments

feed	Morningstar Feed Table. character
contracts	Symbols vector. character
from	From date yyyy-mm-dd. character
iuser	Morningstar user name as character - sourced locally in examples. character
ipassword	Morningstar user password as character - sourced locally in examples. character

Value

wide data frame. tibble

Author(s)

Philippe Cote

Examples

```
## Not run:
getPrices(
  feed = "CME_NymexFutures_EOD", contracts = c("@CL0Z", "@CL1F", "@CL21H", "@CL21Z"),
  from = "2020-01-01", iuser = username, ipassword = password
)

## End(Not run)
```

---

holidaysOil *dataset: NYMEX and ICE holiday calendars*

Description

Holiday calendars for NYMEX and ICE Brent

Usage

```
holidaysOil
```

Format

data frame

Value

tibble

npv *NPV*

Description

Computes NPV with discount factor interpolation. This function is used for teaching NPV and NPV at Risk and needs to be customized.

Usage

```
npv(
  init.cost = -375,
  C = 50,
  cf.freq = 0.25,
  TV = 250,
  T2M = 2,
  disc.factors = us.df,
  BreakEven = FALSE,
  BE.yield = 0.01
)
```

Arguments

`init.cost`	Initial investment cost. `numeric`
`C`	Periodic cash flow. `numeric`
`cf.freq`	Cash flow frequency in year fraction e.g. quarterly = 0.25. `numeric`
`TV`	Terminal Value. `numeric`
`T2M`	Time to Maturity in years. `numeric`
`disc.factors`	Data frame of discount factors using ir.df.us() function. `numeric`
`BreakEven`	TRUE when using a flat discount rate assumption. `logical`
`BE.yield`	Set the flat IR rate when BreakEven = TRUE. `logical`

Value

List of NPV and NPV Data frame. `list`

Author(s)

Philippe Cote

Examples

```
npv(
  init.cost = -375, C = 50, cf.freq = .5, TV = 250, T2M = 2,
  disc.factors = RTL::usSwapCurves, BreakEven = FALSE, BE.yield = .0399
)$npv
npv(
  init.cost = -375, C = 50, cf.freq = .5, TV = 250, T2M = 2,
  disc.factors = RTL::usSwapCurves, BreakEven = FALSE, BE.yield = .0399
)$df
```

---

ohlc *dataset: randomiser to convert settlement into OHLC*

Description

OHLC profile using historical CL 1st Contract OHLC

Usage

```
ohlc
```

Format

data frame

Value

tibble

Source

CME

---

planets *dataset: IR compounding*

Description

Planet metrics from NASA

Usage

```
planets
```

Format

data frame

Value

tibble

Source

https://nssdc.gsfc.nasa.gov/planetary/factsheet/index.html

---

promptBeta *Computes betas of futures contracts with respect to the 1st line contract*

Description

Returns betas of futures contracts versus front futures contract.

Usage

```
promptBeta(x = x, period = "all", betatype = "all", output = "chart")
```

Arguments

x	Wide dataframe with date column and multiple series columns (multivariate). tibble
period	"all" or numeric period of time in last n periods as character eg "100". character
betatype	"all" "bull" "bear". character
output	"betas" or "chart". character

Value

betas data frame tibble or plotly chart of betas htmlwidgets

Author(s)

Philippe Cote

Examples

```
## Not run:
 x <- dflong %>%
 dplyr::filter(grepl("CL",series)) %>%
 dplyr::mutate(series = readr::parse_number(series)) %>% dplyr::group_by(series) %>%
 RTL::returns(df = ., retType = "abs",period.return = 1,spread = TRUE) %>%
 RTL::rolladjust(x = .,commodityname = c("cmewti"),rolltype = c("Last.Trade")) %>%
 # removing the day it prices went negative...
 dplyr::filter(!date %in% c(as.Date("2020-04-20"),as.Date("2020-04-21")))
promptBeta(x = x, period = "all", betatype = "all", output = "chart")
promptBeta(x = x, period = "all", betatype = "bull", output = "betas")
promptBeta(x = x, period = "100", betatype = "bear", output = "betas")
```

```
## End(Not run)
```

`refineryLP` *LP model for refinery optimization*

Description

Plain vanilla refinery optimization LP model.

Usage

```
refineryLP(
  crudes = RTL::refineryLPdata$inputs,
  products = RTL::refineryLPdata$outputs
)
```

Arguments

`crudes`	Data frame of crude inputs. `tibble`
`products`	Data frame of product outputs and max outputs. `tibble`

Value

Optimal crude slate and profits. `tibble`

Author(s)

Philippe Cote

Examples

```
refineryLP(crudes = RTL::refineryLPdata$inputs, products = RTL::refineryLPdata$outputs)
```

`refineryLPdata` *dataset: refinery LP model sample inputs and outputs*

Description

Simple refinery to be used in running LP modeling for education purposes.

Usage

```
refineryLPdata
```

Format

list

Value

list

---

returns *Compute absolute, relative or log returns.*

Description

Computes periodic returns from a dataframe ordered by date

Usage

```
returns(df = dflong, retType = "abs", period.return = 1, spread = FALSE)
```

Arguments

df	Long dataframe with colnames = c("date","value","series"). character
retType	"abs" for absolute, "rel" for relative, or "log" for log returns. character
period.return	Number of rows over which to compute returns. numeric
spread	TRUE if you want to spread into a long dataframe. logical

Value

A dataframe object of returns. tibble

Author(s)

Philippe Cote

Examples

```
x <- dflong %>% dplyr::filter(grepl("CL01", series))
returns(df = x, retType = "abs", period.return = 1, spread = TRUE)
```

rolladjust *Adjusts daily returns for futures contracts roll*

Description

Returns a xts price or return object adjusted for contract roll. The methodology used to adjust returns is to remove the daily returns on the day after expiry and for prices to adjust historical rolling front month contracts by the size of the roll at each expiry. This is conducive to quantitative trading strategies as it reflects the PL of a financial trader.

Usage

```
rolladjust(x, commodityname = c("cmewti"), rolltype = c("Last.Trade"), ...)
```

Arguments

x	A df of returns.
commodityname	Name of commodity in expiry_table: unique(expiry_table$cmdty) or "cmecan" for WCW
rolltype	Type of contract roll: "Last.Trade" or "First.Notice".
...	Other parms

Value

Roll-adjusted xts object of returns

Author(s)

Philippe Cote

Examples

```
ret <- dplyr::tibble(date = seq.Date(Sys.Date() - 60, Sys.Date(), 1), CL01 = rnorm(61, 0, 1))
rolladjust(x = ret, commodityname = c("cmewti"), rolltype = c("Last.Trade"))
```

simGBM *GBM process simulation*

Description

Simulates a Geometric Brownian Motion process

Usage

```
simGBM(
  nsims = 1,
  S0 = 10,
  drift = 0,
  sigma = 0.2,
  T2M = 1,
  dt = 1/12,
  vec = TRUE
)
```

Arguments

nsims	Number of simulations. Defaults to 1. numeric
S0	Spot price at t=0. numeric
drift	Drift term in percentage. numeric
sigma	Standard deviation. numeric
T2M	Maturity in years. numeric
dt	Time step in period e.g. 1/250 = 1 business day. numeric
vec	Vectorized implementation. Defaults to TRUE. logical

Value

A tibble of simulated values. tibble

Author(s)

Philippe Cote

Examples

```
simGBM(nsims = 2, S0 = 10, drift = 0, sigma = 0.2, T2M = 1, dt = 1 / 12, vec = TRUE)
```

---

simMultivariates *Multivariate normal from historical dataset*

Description

Generates multivariate random epsilons using absolute returns.

Usage

```
simMultivariates(nsims = 10, x, s0 = NULL)
```

Arguments

nsims	Number of simulations. Defaults to 10. numeric
x	Wide data frame of prices with date as first column. tibble
s0	Vector of starting value for each variables. Defaults to NULL with zero. numeric

Value

List of means, sds, covariance matrix, correlation matrix and simulated values. list

Author(s)

Philippe Cote

Examples

```
## Not run:
simMultivariates(nsims = 10, x = RTL::fizdiffs, s0 = NULL)

## End(Not run)
```

---

simOU *OU process simulation*

Description

Simulates a Ornstein–Uhlenbeck process

Usage

```
simOU(
  nsims = 2,
  S0 = 5,
  mu = 5,
  theta = 0.5,
  sigma = 0.2,
  T2M = 1,
  dt = 1/12,
  epsilon = NULL
)
```

Arguments

nsims	number of simulations. Defaults to 2. numeric
S0	S at t=0. numeric
mu	Mean reversion level. numeric
theta	Mean reversion speed. numeric

sigma	Standard deviation. numeric
T2M	Maturity in years. numeric
dt	Time step size e.g. 1/250 = 1 business day. numeric
epsilon	Defaults to NULL function generates its own. numeric OPTIONAL: Array of epsilons for nsims = 1, if you want to feed your own e.g. in a multivariate context.

Value

Simulated values. tibble

Author(s)

Philippe Cote

Examples

```
simOU(nsims = 5, S0 = 5, mu = 5, theta = .5, sigma = 0.2, T2M = 1, dt = 1 / 12, epsilon = NULL)
simOU(nsims = 1, S0 = 5, mu = 5, theta = .5, sigma = 0.2, T2M = 1, dt = 1 / 12,
epsilon = matrix(rnorm(12,0,sqrt(1/12))))
simOU(nsims = 2, S0 = 5, mu = 5, theta = .5, sigma = 0.2, T2M = 1, dt = 1 / 12,
epsilon = replicate(2,rnorm(12,0,sqrt(1/12))))
```

---

simOUJ *OUJ process simulation*

Description

Simulates a Ornstein–Uhlenbeck process with Jumps

Usage

```
simOUJ(
  nsims = 2,
  S0 = 5,
  mu = 5,
  theta = 10,
  sigma = 0.2,
  jump_prob = 0.05,
  jump_avesize = 2,
  jump_stdv = 0.05,
  T2M = 1,
  dt = 1/250
)
```

Arguments

nsims	number of simulations. Defaults to 2. numeric
S0	S at t=0. numeric
mu	Mean reversion level. numeric
theta	Mean reversion speed. numeric
sigma	Standard deviation. numeric
jump_prob	Probability of jumps. numeric
jump_avesize	Average size of jumps. numeric
jump_stdv	Standard deviation of jump average size. numeric
T2M	Maturity in years. numeric
dt	Time step size e.g. 1/250 = 1 business day. numeric

Value

Simulated values. tibble

Author(s)

Philippe Cote

Examples

```
simOUJ(nsims = 2, S0 = 5, mu = 5, theta = .5, sigma = 0.2,
jump_prob = 0.05, jump_avesize = 3, jump_stdv = 0.05,
T2M = 1, dt = 1 / 12)
```

---

simOUt *OU process simulation*

Description

Simulates a Ornstein–Uhlenbeck process with mu as a function of time

Usage

```
simOUt(
  nsims = 2,
  S0 = 0,
  mu = dplyr::tibble(t = 0:20, mr = c(rep(2, 7), rep(4, 14))),
  theta = 12,
  sigma = 0.2,
  T2M = 1,
  dt = 1/12
)
```

Arguments

nsims	number of simulations. Defaults to 2. numeric
S0	S at t=0. numeric
mu	data frame of mean reversion level as a function of time. tibble
theta	Mean reversion speed. numeric
sigma	Standard deviation. numeric
T2M	Maturity in years. numeric
dt	Time step size e.g. 1/250 = 1 business day. numeric

Value

Simulated values. tibble

Author(s)

Philippe Cote

Examples

```
mu = dplyr::tibble(t = 0:20,mr = c(rep(2,7),rep(4,14)))
simOUt(nsims = 2, S0 = 5, mu = mu, theta = .5, sigma = 0.2, T2M = 1, dt = 1 / 12)
```

---

spot2futConvergence *dataset: spot to futures convergence*

Description

Cash and futures

Usage

```
spot2futConvergence
```

Format

data frame

Value

tibble

Source

Morningstar, EIA

spot2futCurve *dataset: spot to futures convergence curve*

Description

Forward Curve

Usage

```
spot2futCurve
```

Format

data frame

Value

tibble

Source

Morningstar, EIA

spreadOption *Kirk's Approximation for Spread Option Pricing*

Description

Computes the price and Greeks of European spread options using Kirk's 1995 approximation. The spread option gives the holder the right to receive the difference between two asset prices (F2 - F1) at maturity, if positive, in exchange for paying the strike price X.

Usage

```
spreadOption(F1, F2, X, sigma1, sigma2, rho, T2M, r, type = "call")
```

Arguments

F1	numeric, the forward price of the first asset.
F2	numeric, the forward price of the second asset.
X	numeric, the strike price of the spread option.
sigma1	numeric, the volatility of the first asset (annualized).
sigma2	numeric, the volatility of the second asset (annualized).
rho	numeric, the correlation coefficient between the two assets (-1 <= rho <= 1).
T2M	numeric, the time to maturity in years.
r	numeric, the risk-free interest rate (annualized).
type	character, the type of option to evaluate, either "call" or "put". Default is "call".

Details

Kirk's approximation is particularly useful for spread options where the exercise price is zero or small relative to the asset prices. The approximation assumes that the ratio of the assets follows a lognormal distribution.

The implementation includes a small constant (epsilon) to avoid numerical instabilities that might arise from division by zero.

Value

A list containing the following elements:

- `price`: The price of the spread option
- `delta_F1`: The sensitivity of the option price to changes in F1
- `delta_F2`: The sensitivity of the option price to changes in F2
- `gamma_F1`: The second derivative of the option price with respect to F1
- `gamma_F2`: The second derivative of the option price with respect to F2
- `gamma_cross`: The mixed second derivative with respect to F1 and F2
- `vega_1`: The sensitivity of the option price to changes in sigma1
- `vega_2`: The sensitivity of the option price to changes in sigma2
- `theta`: The sensitivity of the option price to the passage of time
- `rho`: The sensitivity of the option price to changes in the interest rate

References

Kirk, E. (1995) "Correlation in the Energy Markets." Managing Energy Price Risk, Risk Publications and Enron, London, pp. 71-78.

Examples

```
# Price a call spread option with the following parameters:
F1 <- 100  # Forward price of first asset
F2 <- 110  # Forward price of second asset
X <- 5     # Strike price
sigma1 <- 0.2  # Volatility of first asset
sigma2 <- 0.25 # Volatility of second asset
rho <- 0.5     # Correlation between assets
T2M <- 1       # One year to maturity
r <- 0.05      # Risk-free rate

result_call <- spreadOption(F1, F2, X, sigma1, sigma2, rho, T2M, r, type = "call")
result_put <- spreadOption(F1, F2, X, sigma1, sigma2, rho, T2M, r, type = "put")
```

steo *dataset: EIA Short Term Energy Outlook*

Description

Short Term Energy Outlook from the EIA.

Usage

```
steo
```

Format

plotly object

Value

htmlwidget

Source

eia

stocks *dataset: Yahoo Finance data sets*

Description

Traded equity prices and returns

Usage

```
stocks
```

Format

list

Value

list

Source

Yahoo Finance

swapCOM *Commodity Calendar Month Average Swaps*

Description

Commodity swap pricing from exchange settlement

Usage

```
swapCOM(
  futures = futs,
  futuresNames = c("CL0M", "CL0N"),
  pricingDates = c("2020-05-01", "2020-05-30"),
  contract = "cmewti",
  exchange = "nymex"
)
```

Arguments

futures	Wide data frame of futures prices for the given swap pricing dates. tibble
futuresNames	Tickers of relevant futures contracts. character
pricingDates	Vector of start and end pricing dates. See example. character
contract	Contract code in data(expiry_table). sort(unique(expiry_table$cmdty)) for options. character
exchange	Exchange code in data(holidaysOil). Currently only "nymex" and "ice" supported. character

Value

Data frame of histocial swap prices. tibble

Author(s)

Philippe Cote

Examples

```
## Not run:
c <- paste0("CL0", c("M", "N", "Q"))
futs <- getPrices(
  feed = "CME_NymexFutures_EOD", contracts = c, from = "2019-08-26",
  iuser = username, ipassword = password
)
swapCOM(
  futures = futs, futuresNames = c("CL0M", "CL0N"),
  pricingDates = c("2020-05-01", "2020-05-30"), contract = "cmewti", exchange = "nymex"
)
```

```
## End(Not run)
```

---

swapFutWeight *Commodity Calendar Month Average Swap futures weights*

Description

Returns the percentage weight of the future in Calendar Month Average swaps

Usage

```
swapFutWeight(
  Month = "2020-09-01",
  contract = "cmewti",
  exchange = "nymex",
  output = "first.fut.weight"
)
```

Arguments

Month	First calendar day of the month. character
contract	Contract code in data(expiry_table). sort(unique(expiry_table$cmdty)) for options. character
exchange	Exchange code in data(holidaysOil). Currently only "nymex" and "ice" supported. character
output	Either "numDaysFut1", "numDaysFut2" or "first.fut.weight". character

Value

Depending on output setting. numeric If first.fut.weight, to compute swap 1 - first.fut.weight = % applied to 2nd line contract.

Author(s)

Philippe Cote

Examples

```
swapFutWeight(
  Month = "2020-09-01",
  contract = "cmewti", exchange = "nymex", output = "first.fut.weight"
)
```

swapInfo *Commodity Swap details to learn their pricing*

Description

Returns dataframe required to price a WTI averaging instrument based on first line settlements.

Usage

```
swapInfo(
  date = "2023-08-24",
  feed = "CME_NymexFutures_EOD_continuous",
  ticker = "CL",
  contract = "cmewti",
  exchange = "nymex",
  iuser = "x@xyz.com",
  ipassword = "pass",
  output = "all"
)
```

Arguments

date	Character date as of which you want to extract daily settlement and forward values. character
feed	Feeds for Morningstar getCurve() and getPrice(). character
ticker	Nymex contract code. character
contract	Contract code in data(expiry_table). sort(unique(expiry_table$cmdty)) for options. character
exchange	Exchange code in data(holidaysOil). Defaults to "nymex". character
iuser	Morningstar user name as character - sourced locally in examples. character
ipassword	Morningstar user password as character - sourced locally in examples. character
output	"chart" or "all". character

Value

Plot or a list of data frame and plot if output = "all". htmlwidget or list

Author(s)

Philippe Cote

Examples

```
## Not run:
swapInfo(
  date = "2020-05-06", feed = "CME_NymexFutures_EOD_continuous",
  ticker = "CL",
  contract = "cmewti", exchange = "nymex",
  iuser = "x@xyz.com", ipassword = "pass", output = "all"
)

## End(Not run)
```

---

swapIRS *Interest Rate Swap*

Description

Computes the mark to market of an IRS

Usage

```
swapIRS(
  trade.date = lubridate::today(),
  eff.date = lubridate::today() + 2,
  mat.date = lubridate::today() + 2 + lubridate::years(2),
  notional = 1e+06,
  PayRec = "Rec",
  fixed.rate = 0.05,
  float.curve = usSwapCurves,
  reset.freq = 3,
  disc.curve = usSwapCurves,
  convention = c("act", 360),
  bus.calendar = "NY",
  output = "price"
)
```

Arguments

`trade.date`	Date object. Defaults to today(). `Date`
`eff.date`	Date object. Defaults to today() + 2 days. `Date`
`mat.date`	Date object. Defaults to today() + 2 years. `Date`
`notional`	Numeric value of notional. Defaults to 1,000,000. `numeric`
`PayRec`	"Pay" or "Rec" fixed. `character`
`fixed.rate`	Numeric fixed interest rate. Defaults to 0.05. `Date`
`float.curve`	List of interest rate curves. Defaults to data("usSwapCurves"). `list`

reset.freq	Numeric where 1 = "monthly", 3 = quarterly, 6 = Semi annual 12 = yearly. character
disc.curve	List of interest rate curves. Defaults to data("usSwapCurves"). list
convention	Vector of convention e.g. c("act",360) c(30,360),... character
bus.calendar	Banking day calendar. Not implemented.
output	"price" for swap price or "all" for price, cash flow data frame, duration. character

Value

List of swap price, cash flow data frame, duration. list

Author(s)

Philippe Cote

Examples

```
data("usSwapCurves")
swapIRS(
  trade.date = as.Date("2020-01-04"), eff.date = as.Date("2020-01-06"),
  mat.date = as.Date("2022-01-06"), notional = 1000000,
  PayRec = "Rec", fixed.rate = 0.05, float.curve = usSwapCurves, reset.freq = 3,
  disc.curve = usSwapCurves, convention = c("act", 360),
  bus.calendar = "NY", output = "all"
)
```

---

tickers_eia *datasest: metadata of key EIA tickers grouped by products.*

Description

Supports automated upload of EIA data through its API by categories. Data frame organized by Supply Demand categories and products.

Usage

```
tickers_eia
```

Format

data frame

Value

tibble

tradeCycle *dataset: Canadian and US physical crude trading calendars*

Description

Crude Trading Trade Cycles. Note that is uses NYMEX calendar (WIP)

Usage

```
tradeCycle
```

Format

data frame

Value

tibble

tradeHubs *dataset: GIS locations for crude oil trading hubs*

Description

Trading Hubs

Usage

```
tradeHubs
```

Format

data frame

Value

tibble

tradeprocess *dataset: data for teaching the various ways to monetize a market call.*

Description

Data set for explaining the various ways to monetize a market view.

Usage

```
tradeprocess
```

Format

data frame

Value

tibble

tradeStats *Risk-reward statistics for quant trading*

Description

Compute list of risk reward metrics

Usage

```
tradeStats(x, Rf = 0)
```

Arguments

x	Univariate xts object of returns OR dataframe with date and return variables. xts
Rf	Risk-free rate. numeric

Value

List of risk/reward metrics. list

Author(s)

Philippe Cote

Examples

```
library(PerformanceAnalytics)
tradeStats(x = stocks$spy, Rf = 0)
```

tradeStrategyDY *Sample quantitative trading strategy*

Description

Based on dividend yield.

Usage

```
tradeStrategyDY(data, par1value = 50, par2value = 200)
```

Arguments

data	Dataframe of OHLC data e.g. RTL::uso. tibble
par1value	Value of first parameter e.g. short MA. numeric
par2value	Value of second parameter e.g. long MA. numeric

Value

Dataframe with indicators, signals, trades and profit and loss. tibble

Author(s)

Philippe Cote

Examples

```
tradeStrategyDY(data = RTL::stocks$ry, par1value = 50, par2value = 200)
```

tradeStrategySMA *Sample quantitative trading strategy*

Description

Moving average crossover strategy

Usage

```
tradeStrategySMA(data = RTL::stocks$uso, par1value = 50, par2value = 200)
```

Arguments

data	Dataframe of OHLC data e.g. RTL::uso. tibble
par1value	Value of first parameter e.g. short MA. numeric
par2value	Value of second parameter e.g. long MA. numeric

Value

Dataframe with indicators, signals, trades and profit and loss. `tibble`

Author(s)

Philippe Cote

Examples

```
tradeStrategySMA(data = RTL::stocks$uso, par1value = 50, par2value = 200)
```

---

tsQuotes *dataset: interest rate curve data for RQuantlib .*

Description

USD IR curve input for RQuantlib::DiscountCurve

Usage

```
tsQuotes
```

Format

data frame

Value

`tibble`

---

usSwapCurves *dataset: US bootstrapped interest rate curve.*

Description

USD IR Discount, Forward and Zero curves from RQuantlib::DiscountCurve

Usage

```
usSwapCurves
```

Format

List

Value

list

Source

Morningstar and FRED

---

usSwapCurvesPar *dataset: US bootstrapped interest rate curve parallel sample.*

Description

USD IR Discount, Forward and Zero curves from RQuantlib::DiscountCurve - Parallel toy data set

Usage

```
usSwapCurvesPar
```

Format

data frame

Value

tibble

---

wtiSwap *dataset: WTI Calendar Month Average Swap pricing data*

Description

WTI Crude futures

Usage

```
wtiSwap
```

Format

data frame

Value

tibble

Source

Morningstar

Index