Package 'RTL'

November 16, 2022

Type Package

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

Version 1.3.1

Date 2022-11-14

Description A toolkit for Commodities 'analytics', risk management and trading professionals. Includes functions for API calls to 'Morningstar Commodities' and 'Genscape'.

License MIT + file LICENSE

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

Depends R (>= 4.0)

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

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

Encoding UTF-8

LazyData true

LazyDataCompression xz

RoxygenNote 7.2.1

Config/testthat/edition 3

LinkingTo Rcpp

NeedsCompilation yes

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

Maintainer Philippe Cote <pcote@ualberta.ca>

Repository CRAN

Date/Publication 2022-11-16 01:40:02 UTC

R topics documented:

---

bond *Bond pricing*

Description

Compute bond price, cash flow table and duration

Usage

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

Arguments

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

Value

Price, cash flows data frame and/or duration

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

Value

A plotly object or a dataframe

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

Value

A plotly object or a dataframe

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)
`cmdty`	Futures contract code in expiry_table object: unique(expiry_table$cmdty)
`weekly`	Defaults to TRUE for weekly forward curves
`...`	other graphical parameters

Value

plot of forward curves through time

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
title	Chart title

Value

A plotly object

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.
geometric	Use geometric returns TRUE or FALSE.
main	Chart title.
linesize	Size of lines in chart and legend.

Value

Cumulative performance and drawdown charts.

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.
daysFromExpiry	Number of days (numeric) from expiry to compute spreads.
from	From date as character string
conversion	Defaults to c(1,1) first and second contracts. 42 from $ per gallons to bbls.
feed	Morningstar Feed Table.
iuser	Morningstar user name as character - sourced locally in examples.
ipassword	Morningstar user password as character - sourced locally in examples.
title	Title for chart.
yaxis	y-axis label.
output	"chart" for plotly object or "data" for dataframe.

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
title	Default is a blank space returning the unique value in df$series.
per	Frequency of seasonality "yearweek" (DEFAULT). "yearmonth", "yearquarter"
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)
```

CRReuro *Cox-Ross-Rubinstein binomial option model*

Description

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

Usage

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

Arguments

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

Value

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

Author(s)

Philippe Cote

Examples

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

crudeOil *dataset: crude assays*

Description

crude assays

Usage

```
crudeOil
```

Format

list

cushing *dataset: WTI Cushing Futures and storage utilization*

Description

c1, c2, c1c2 and Cushing storage utilization

Usage

```
cushing
```

Format

list

Source

CME and EIA

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

Description

Futures settlement data set.

Usage

```
dflong
```

Format

data frame

Source

Morningstar Commodities

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

Description

Futures settlement data set.

Usage

```
dfwide
```

Format

data frame

Source

Morningstar Commodities

distdescplot *Summary of distribution properties of a timeseries*

Description

Provides a summary of returns distribution

Usage

```
distdescplot(x = x)
```

Arguments

x	Wide dataframe with date column and single series (univariate).

Value

Multiple plots describing the distribution.

Author(s)

Philippe Cote

Examples

```
x <- dplyr::tibble(
  date = seq.Date(Sys.Date() - 1000, Sys.Date(), 1),
  CL01 = c(rnorm(501, 0, 0.02), rnorm(500, 0, 0.01))
)
distdescplot(x = x)
```

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
x	List as provided by output of RTL::simMultivariates().
expectedReturns	Defaults to NULL using periodic returns means.

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

Author(s)

Philippe Cote

Examples

```
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))
```

---

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

Value

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

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.
key	Your private EIA API token as character "yourapikey".
long	TRUE (default) to return a long data frame or FASLE for wide

Value

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

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

eiaStorageCap *dataset: EIA working storage capacity*

Description

EIA working storage capacity in kbs except NG in bcf.

Usage

```
eiaStorageCap
```

Format

data frame

eurodollar *dataset: Eurodollar futures contracts*

Description

ED futures contract for December 2024

Usage

```
eurodollar
```

Format

data frame

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

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

Description

Parameter estimation for Ornstein–Uhlenbeck process

Usage

```
fitOU(spread)
```

Arguments

spread	Spread time series.

Value

List of alpha, mu and sigma estimates

Author(s)

Philippe Cote

Examples

```
spread <- simOU(mu = 5, theta = .5, sigma = 0.2, T = 5, dt = 1 / 250)
fitOU(spread)
```

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

futuresRef *dataset: futures contracts metadata*

Description

Exchange-traded contract month codes and specifications.

Usage

```
futuresRef
```

Format

data frame

fxfwd *dataset: USDCAD FX forward rates*

Description

USDCAD 1-year and 5-year forward points

Usage

```
fxfwd
```

Format

data frame

Source

Morningstar

---

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).
out	"chart" to return chart, "data" to return data or "fit" for garch fit output

Value

plot.xts object or xts series

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)
```

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 = "Crb_Futures_Price_Volume_And_Open_Interest",
  contract = "CL",
  date = "2020-08-10",
  fields = c("Open, High, Low, Close"),
  iuser = "x@xyz.com",
  ipassword = "pass"
)
```

Arguments

feed	Morningstar Feed Table e.g "Crb_Futures_Price_Volume_And_Open_Interest".
contract	Morningstar contract root e.g. "CL" for CME WTI and "BG" for ICE Brent.
date	From date as character string.
fields	Defaults to c("Open, High, Low, Close").
iuser	Morningstar user name as character - sourced locally in examples.
ipassword	Morningstar user password as character - sourced locally in examples.

Value

wide data frame

Current Feeds Supported

- Crb_Futures_Price_Volume_And_Open_Interest
- CME_NymexFuturesIntraday_EOD
- ICE_EuroFutures and ICE_EuroFutures_continuous

Author(s)

Philippe Cote

Examples

```
## Not run:
# CME WTI Futures
getCurve(
  feed = "Crb_Futures_Price_Volume_And_Open_Interest", contract = "CL",
  date = "2020-07-13", fields = c("Open, High, Low, Close"),
  iuser = "x@xyz.com", ipassword = "pass"
)

getCurve(
  feed = "Crb_Futures_Price_Volume_And_Open_Interest", contract = "BG",
  date = "2020-07-13", fields = c("Open, High, Low, Close"),
  iuser = "x@xyz.com", ipassword = "pass"
)

getCurve(
  feed = "LME_ClosingPriceDelayed", contract = "AHD",
  date = "2021-06-25", fields = c("Last_Price"),
  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.
regions	See API webpage. Multiple values separated by commas e.g. "Canada", "Gulf-Coast").
pipelineIDs	See API webpage. c(98,54...) for specific pipes.
revision	See API webpage.
limit	See API webpage. Max 5000
offset	See API webpage.
startDate	"yyyy-mm-dd" as character string
endDate	"yyyy-mm-dd" as character string
apikey	Your API key as a character string.

Value

wide data frame

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

Value

wide data frame

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://gis.energy.gov.ab.ca/GeoviewData/OS_Agreements_Shape.zip"
)
```

Arguments

url	URL of the zipped shapefile

Value

SpatialPointsDataFrame

Author(s)

Philippe Cote

Examples

```
## Not run:
getGIS(url = "https://gis.energy.gov.ab.ca/GeoviewData/OS_Agreements_Shape.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.
`contract`	Morningstar key.
`from`	From date as character string
`iuser`	Morningstar user name as character - sourced locally in examples.
`ipassword`	Morningstar user password as character - sourced locally in examples.

Value

wide data frame

Current Feeds Supported

- CME_CbotFuturesEOD and CME_CbotFuturesEOD_continuous
- CME_NymexFutures_EOD and CME_NymexFutures_EOD_continuous
- 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_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_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

contracts	Symbols vector
from	From date as character string
iuser	Morningstar user name as character - sourced locally in examples.
ipassword	Morningstar user password as character - sourced locally in examples.

Value

wide data frame

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

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

Value

List of NPV and NPV Data frame

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

Source

CME

---

planets *dataset: IR compounding*

Description

Planet metrics from NASA

Usage

```
planets
```

Format

data frame

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).
period	"all" or numeric period of time in last n periods as character eg "100".
betatype	"all" "bull" "bear".
output	"betas" or "chart"

Value

betas data frame or plotly chart of betas

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
products	Data frame of product outputs and max outputs.

Value

Optimal crude slate and profits

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

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")
retType	"abs" for absolute, "rel" for relative, or "log" for log returns.
period.return	Number of rows over which to compute returns.
spread	TRUE if you want to spread into a long dataframe.

Value

A dataframe object of returns.

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
S0	Spot price at t=0
drift	Drift term in percentage
sigma	Standard deviation

T2M	Maturity in years
dt	Time step in period e.g. 1/250 = 1 business day.
vec	Vectorized implementation. Defaults to TRUE

Value

A tibble of simulated values

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
x	Wide data frame of prices with date as first column.
s0	Vector of starting value for each variables. Defaults to NULL with zero.

Value

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

Author(s)

Philippe Cote

Examples

```
simMultivariates(nsims = 10, x = RTL::fizdiffs, s0 = NULL)
```

`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
`S0`	S at t=0
`mu`	Mean reversion level
`theta`	Mean reversion speed
`sigma`	Standard deviation
`T2M`	Maturity in years
`dt`	Time step size e.g. 1/250 = 1 business day.
`epsilon`	Defaults to NULL function generates its own. OPTIONAL: Array of epsilons for nsims = 1, if you want to feed your own e.g. in a multivariate context.

Value

A numeric vector of simulated values

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

Value

A numeric vector of simulated values

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

Value

A numeric vector of simulated values

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

Source

Morningstar, EIA

spot2futCurve *dataset: spot to futures convergence curve*

Description

Forward Curve

Usage

```
spot2futCurve
```

Format

data frame

Source

Morningstar, EIA

steo *dataset: EIA Short Term Energy Outlook*

Description

Short Term Energy Outlook from the EIA.

Usage

```
steo
```

Format

plotly object

Source

eia

stocks *dataset: Yahoo Finance data sets*

Description

Traded equity prices and returns

Usage

```
stocks
```

Format

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

Value

Data frame of histocial swap prices.

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

Value

What you defined in outputs. 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 = "2020-05-06",
  feeds = dplyr::tibble(feed = c("Crb_Futures_Price_Volume_And_Open_Interest",
    "CME_NymexFutures_EOD_continuous"), ticker = c("CL", "CL_001_Month")),
  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.
feeds	Feeds for Morningstar getCurve() and getPrice().
contract	Contract code in data(expiry_table). sort(unique(expiry_table$cmdty)) for options.
exchange	Exchange code in data(holidaysOil). Defaults to "nymex".
iuser	Morningstar user name as character - sourced locally in examples.
ipassword	Morningstar user password as character - sourced locally in examples.
output	"chart" or "all"

Value

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

Author(s)

Philippe Cote

Examples

```
## Not run:
feeds <- dplyr::tibble(
  feed = c(
    "Crb_Futures_Price_Volume_And_Open_Interest",
    "CME_NymexFutures_EOD_continuous"
  ),
  ticker = c("CL", "CL_001_Month")
)
swapInfo(
  date = "2020-05-06", feeds = feeds, 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().
eff.date	Date object. Defaults to today() + 2 days.
mat.date	Date object. Defaults to today() + 2 years.
notional	Numeric value of notional. Defaults to 1,000,000.

PayRec	"Pay" or "Rec" fixed.
fixed.rate	Numeric fixed interest rate. Defaults to 0.05.
float.curve	List of interest rate curves. Defaults to data("usSwapCurves").
reset.freq	Numeric where 1 = "monthly", 3 = quarterly, 6 = Semi annual 12 = yearly.
disc.curve	List of interest rate curves. Defaults to data("usSwapCurves").
convention	Vector of convention e.g. c("act",360) c(30,360),...
bus.calendar	Banking day calendar. Not implemented.
output	"price" for swap price or "all" for price, cash flow data frame, duration.

Value

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

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

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

Description

Crude Trading Trade Cycles

Usage

```
tradeCycle
```

Format

data frame

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

Description

Trading Hubs

Usage

```
tradeHubs
```

Format

data frame

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

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.
Rf	Risk-free rate

Value

List of risk/reward metrics.

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
par1value	Value of first parameter e.g. short MA
par2value	Value of second parameter e.g. long MA

Value

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

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
`par1value`	Value of first parameter e.g. short MA
`par2value`	Value of second parameter e.g. long MA

Value

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

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

usSwapCurves *dataset: US bootstrapped interest rate curve.*

Description

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

Usage

```
usSwapCurves
```

Format

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

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

Description

WTI Crude futures

Usage

```
wtiSwap
```

Format

data frame

Source

Morningstar

Index