Package ‘RTL’

May 19, 2022

Type Package

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

Version 1.1.0

Date 2022-05-18

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, htr, jsonlite, lubridate, magrittr, plotly, purrr, RCurl, readr, rlang, stringr, tibble, tidyr, timetk, xts, zoo, glue, Rcpp, lifecycle,

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

Encoding UTF-8

LazyData true

LazyDataCompression xz

RoxygenNote 7.2.0

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 2022-05-19 00:10:02 UTC
### R topics documented:

- bond .................................................. 3
- cancrudeaassays ................................... 4
- cancrudeaassayssum ................................. 4
- cancrudeprices ....................................... 5
- chart_eia_sd ........................................ 5
- chart_eia_steo ....................................... 6
- chart_fwd_curves .................................... 7
- chart_pairs ........................................... 8
- chart_PerfSummary ................................. 8
- chart_spreads ........................................ 9
- chart_zscore ......................................... 11
- CRReuro ............................................... 12
- crudeassaysBP ....................................... 13
- crudeassaysXOM ..................................... 13
- crudes ................................................. 13
- dflong ................................................. 14
- dfwide .................................................. 14
- distdescplot ......................................... 15
- eia2tidy ............................................... 15
- eiaStocks ............................................. 16
- eiaStorageCap ....................................... 17
- eurodollar ............................................ 17
- expiry_table ......................................... 18
- fitOU .................................................... 18
- fizdiffs ................................................. 19
- fxfwd ................................................... 19
- garch .................................................... 20
- getCurve ............................................... 20
- getGenscapePipeOil ................................. 22
- getGenscapeStorageOil ......................... 23
- getGIS .................................................. 24
- getPrice ............................................... 25
- getPrices .............................................. 28
- holidaysOil .......................................... 29
- npv ....................................................... 29
- planets ............................................... 30
- promptBeta .......................................... 31
- ref.opt.inputs ...................................... 32
- ref.opt.outputs ..................................... 32
- refineryLP ........................................... 32
- returns ............................................... 33
- rolladjust ............................................ 34
- simGBM ............................................... 34
- simOU .................................................. 35
- simOUJ ............................................... 36
- simOUt ............................................... 37
**bond**

**Description**

Compute bond price, cash flow table and duration

**Usage**

```r
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")

cancrudeassays | Data for Canadian crude assays reported by Crude Monitor

Description

Data set with historical Canadian Crude Assays.

Usage

cancrudeassays

Format

data frame

Source

https://crudemonitor.ca/

cancrudeassayssum | Summarized data for Canadian crude assays

Description

Data set with historical Canadian Crude Assays Statistics.

Usage

cancrudeassayssum

Format

data frame

Source

https://crudemonitor.ca/
**cancrudeprices**

*Randomized data for Canadian crude pricing.*

**Description**

Randomized data of Canadian Crude monthly prices versus WTI Calendar Month Average.

**Usage**

```r
cancrudeprices
```

**Format**

data frame

---

**chart_eia_sd**

*EIA weekly Supply Demand information by product group*

**Description**

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

**Usage**

```r
cart_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

```r
## 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

```r
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
**chart_fwd_curves**

**Examples**

```r
## 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**

```r
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**

```r
df <- dfwide %>%
dplyr::select(date, dplyr::starts_with("CL")) %>%
tidy::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) %>%
  tidy::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 Data frame of contract pairs - see example.
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

```r
## Not run:
cpairs <- dplyr::tibble(
  year = c("2014", "2019", "2020"),
  first = c("@HO4H", "@HO9H", "@HO0H"),
  second = c("@CL4J", "@CL9J", "@CL0J")
)
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**

```r
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"
- `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**

```r
# 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")
```
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")

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)
<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
<th>Usage</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>crudeassaysBP</td>
<td>Data for BP crude assays</td>
<td>crudeassaysBP</td>
<td>data frame</td>
</tr>
<tr>
<td>crudeassaysXOM</td>
<td>Data for ExxonMobil crude assays</td>
<td>crudeassaysXOM</td>
<td>data frame</td>
</tr>
<tr>
<td>crudes</td>
<td>Data for crude assays of 50+ types of crude oil.</td>
<td>crudes</td>
<td>data frame</td>
</tr>
</tbody>
</table>
Source

Canadian Crude Monitor and BP Crude Assays

---

**dflong**

*Data for commodity prices in a long dataframe format*

Description

Futures settlement data set.

Usage

dflong

Format

data frame

Source

Morningstar Commodities

---

**dfwide**

*Data for 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)

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 = " ")
eiaStocks

Data for EIA weekly stocks

Description
EIA weekly crude, NG, ULSD and RBOB stocks.

Usage
eiaStocks

Format
data frame
**eiaStorageCap**

*Data for working storage capacity in the US*

**Description**

EIA working storage capacity in kbs except NG in bcf.

**Usage**

`eiaStorageCap`

**Format**

`data frame`

---

**eurodollar**

*Data for Eurodollar futures contracts*

**Description**

ED futures contract for December 2024

**Usage**

`eurodollar`

**Format**

`data frame`

**Source**

Morningstar
## expiry_table

*Metadata for 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

```r
expiry_table
```

### Format

*data frame*

## fitOU

*Fits a Ornstein–Uhlenbeck process to a dataset*

### Description

Parameter estimation for Ornstein–Uhlenbeck process

### Usage

```r
fitOU(spread)
```

### Arguments

- **spread**
  
  Spread time series.

### Value

List of alpha, mu and sigma estimates

### Author(s)

Philippe Cote

### Examples

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

**Description**

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

**Usage**

fizdiffs

**Format**

data frame

---

fxfwd

**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

```r
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

```r
## 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"
)
```

```r
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"
)
```
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/GetPipelineFlowValuesV2/

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庭".
- 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.
getGenscapeStorageOil

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").
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

```r
## 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",
```
getPrice
  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
)
getPrices

getPrices(Morningstar Commodities API multiple calls)

Description

Multiple Morningstar API calls using getPrices 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**

*Metadata for 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

```r
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
```

Data for IR compounding exercises

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**

```r
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**

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

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

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

### Usage

```r
ref.opt.inputs
```

### Format

data frame

---

### Description

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

### Usage

```r
ref.opt.outputs
```

### Format

data frame

---

### Description

Plain vanilla refinery optimization LP model.

### Usage

```r
refineryLP(crudes = ref.opt.inputs, products = ref.opt.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**

```r
refineryLP(crudes = ref.opt.inputs, products = ref.opt.outputs)
```

---

**returns**

*Compute absolute, relative or log returns.*

**Description**

Computes periodic returns from a dataframe ordered by date

**Usage**

```r
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**

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

```r
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**

```r
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"))
```

---

**Description**

Simulates a Geometric Brownian Motion process
Usage

```r
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

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

---

**simOU**

*OU process simulation*

Description

Simulates an Ornstein–Uhlenbeck process

Usage

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

Value

A numeric vector of simulated values

Author(s)

Philippe Cote

Examples

```r
simOU(nsims = 5, S0 = 5, mu = 5, theta = .5, sigma = 0.2, T2M = 1, dt = 1 / 12)
```

Description

Simulates a Ornstein–Uhlenbeck process with Jumps

Usage

```r
simOUJ(
    nsims = 2,
    S0 = 5,
    mu = 5,
    theta = 10,
    sigma = 0.2,
    jump_prob = 0.05,
    jump_avesize = 2,
    jump_stdev = 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_stdev  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_stdev = 0.05,
    T2M = 1, dt = 1 / 12)
```

Description

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

Usage

```
simOUJ(  
    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)

Description

Cash and futures

Usage

spot2futConvergence

Format

data frame

Source

Morningstar, EIA
<table>
<thead>
<tr>
<th>spot2futCurve</th>
<th>Data for spot to futures convergence - forward curve</th>
</tr>
</thead>
</table>

**Description**
Forward Curve

**Usage**
spot2futCurve

**Format**
data frame

**Source**
Morningstar, EIA

<table>
<thead>
<tr>
<th>spy</th>
<th>Sample SPY ETF data set</th>
</tr>
</thead>
</table>

**Description**
Stock price and returns for SPY

**Usage**
spy

**Format**
data frame

**Source**
Yahoo Finance
Description

Commodity swap pricing from exchange settlement

Usage

\[
\text{swapCOM}( \\
\text{futures = futs,} \\
\text{futuresNames = c("CL0M", "CL0N"),} \\
\text{pricingDates = c("2020-05-01", "2020-05-30"),} \\
\text{contract = "cmewti",} \\
\text{exchange = "nymex"} \\
) 
\]

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 historical swap prices.

Author(s)

Philippe Cote

Examples

```r
## 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"  
) 
```
### Description

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

#### Usage

```r
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

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

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

Usage

```r
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
swapIRS

## Not run:
```r
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

```r
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 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
<table>
<thead>
<tr>
<th>Data Source</th>
<th>Description</th>
<th>Usage</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>tradeCycle</td>
<td>Data for Canadian and US physical crude trading calendars</td>
<td>tradeCycle</td>
<td>data frame</td>
</tr>
<tr>
<td>tradeHubs</td>
<td>GIS Data for Crude Oil Trading Hubs</td>
<td>tradeHubs</td>
<td>data frame</td>
</tr>
<tr>
<td>tradeprocess</td>
<td>Data for teaching the various ways to monetize a market call.</td>
<td>tradeprocess</td>
<td>data frame</td>
</tr>
</tbody>
</table>
**tradeStats**  
*Risk-reward statistics for quant trading*

**Description**  
Compute list of risk reward metrics

**Usage**  
```r
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**
```r
library(PerformanceAnalytics)
tradeStats(x = RTL::spy, Rf = 0)
```

---

**tsQuotes**  
*Interest Rate Curve Data for RQuantlib.*

**Description**  
USD IR curve input for RQuantlib::DiscountCurve

**Usage**
```r
tsQuotes
```

**Format**
- data frame
**usSwapCurves**

<table>
<thead>
<tr>
<th>Description</th>
<th>Data for US interest rate discounting using zero rates curve.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Usage</strong></td>
<td>usSwapCurves</td>
</tr>
<tr>
<td><strong>Format</strong></td>
<td>List # @source Morningstar and FRED</td>
</tr>
</tbody>
</table>

**usSwapCurvesPar**

<table>
<thead>
<tr>
<th>Description</th>
<th>Data for US interest rate discounting using zero rates parallel curve.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Usage</strong></td>
<td>usSwapCurvesPar</td>
</tr>
<tr>
<td><strong>Format</strong></td>
<td>data frame</td>
</tr>
</tbody>
</table>

**wtiSwap**

<table>
<thead>
<tr>
<th>Description</th>
<th>Data for WTI Calendar Month Average Swap pricing</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Usage</strong></td>
<td>wtiSwap</td>
</tr>
<tr>
<td><strong>Format</strong></td>
<td>data frame</td>
</tr>
<tr>
<td><strong>Source</strong></td>
<td>Morningstar</td>
</tr>
</tbody>
</table>
Index

* datasets
  cancrudeassays, 4
cancrudeassayssum, 4
cancrudeprices, 5
crudeassaysBP, 13
crudeassaysXOM, 13
crudes, 13
dflong, 14
dfwide, 14
eiaStocks, 16
eiaStorageCap, 17
eurodollar, 17
expiry_table, 18
fizdiffs, 19
fxfwd, 19
holidaysOil, 29
planets, 30
ref.opt.inputs, 32
ref.opt.outputs, 32
spot2futConvergence, 38
spot2futCurve, 39
spy, 39
tickers_eia, 44
tradeCycle, 45
tradeHubs, 45
tradeprocess, 45
tsQuotes, 46
usSwapCurves, 47
usSwapCurvesPar, 47
wtiSwap, 47

bond, 3
cancrudeassays, 4
cancrudeassayssum, 4
cancrudeprices, 5
chart_eia_sd, 5
chart_eia_steo, 6
chart_fwd_curves, 7
chart_pairs, 8

crudeassaysBP, 13
crudeassaysXOM, 13
crudes, 13
dflong, 14
dfwide, 14
distdescplot, 15
eia2tidy, 15
eiaStocks, 16
eiaStorageCap, 17
eurodollar, 17
expiry_table, 18
fitOU, 18
fizdiffs, 19
fxfwd, 19
garch, 20
getCurve, 20
getGenscapePipeOil, 22
getGenscapeStorageOil, 23
getGIS, 24
getPrice, 25
getPrices, 28

holidaysOil, 29

npv, 29
planets, 30
promptBeta, 31
ref.opt.inputs, 32
ref.opt.outputs, 32
refineryLP, 32
returns, 33
INDEX

rolladjust, 34
simGBM, 34
simOU, 35
simOUJ, 36
simOUt, 37
spot2futConvergence, 38
spot2futCurve, 39
spy, 39
swapCOM, 40
swapFutWeight, 41
swapInfo, 42
swapIRS, 43
tickers_eia, 44
tradeCycle, 45
tradeHubs, 45
tradeprocess, 45
tradeStats, 46
tsQuotes, 46
usSwapCurves, 47
usSwapCurvesPar, 47
wtiSwap, 47