Finlib

Versión beta

Riesgo Financiero

11 de octubre de 2023

Índice general

La librería FinRIskLib cuenta con los siguientes objetivos:

- Permitir un acceso rápido a la información oficial disponible dentro de los servidores.
- Permitir la replica de los calculos calculos oficiales en valorización.
- Estandarizar las fuentes y calculos realizados dentro del area de riesgo financiero.

Para cumplir estos objetivos la librería se compone de distintos paquetes, con objetivos y finalidades diferenciadas, los cuales pueden ser usadas de manera individual o conjunta.

Este documento cuenta con tres partes, la primera parte se dedica en mostrar la instalación de la librería, mientras que la segunda parte da una descripción detallada de los paquetes, módulos, clases y funciones que componen la librería. Finalmente la tercera parte muestra algunos ejemplos de uso de la librería.

CAPÍTULO 1

Contenido

1.1 FinRiskLib

1.1.1 Instalación

Para instalar la librería se deben seguir los siguientes pasos:

1.1.1.1 Creación de Entorno Virtual

Aunque la creación de un entorno virtual (Virtual Environment) no es obligatorio se recomienda hacerlo para evitar para mantener un mayor control sobre las versiones y los paquetes que se instalaran. El entorno virtual puede ser creado utilizando dos caminos:

1. Creación desde terminal:

   Se debe ingresar al terminal y utilizar el siguiente comando:

```
python -m venv /path/to/new/virtualenvironment/venv.
```

2. Creación desde Anaconda Navigator:

   Desde el navegador de Anaconda se puede ir a la sección de Enviroments donde se da la opciones de agregar un nuevo entorno de manera automática. Este es el acercamiento recomendado

   Una vez creado el entrono virtual este debe ser activado utilizando el siguiente comando:

```
source venv/bin/activate
```

1.1.1.2 Instalación de la librería

Desde el terminal y con el entorno virtual activado se debe correr el siguiente comando:

```
pip install .
```

Este comando instalara todas las librerías necesarias para el funcionamiento de la libreria, la ventaja de generar esta instalación dentro de un entorno virtual es que la instalación se vera limitada al entorno y no se generarán cambios sobre el entrono del sistema

1.1.1.3 Generación Kernel para utilización en Jupyter

Para poder habilitar la utilización del entorno virtual en jupyter es necesario crear un nuevo kernel utilizando el siguiente comando:

```
ipython kernel install --user --name=<nombre>
```

Debemos recordar que este comando debe ser ejecutado desde el entorno virtual que se desea copiar.

1.1.2 Paquetes

La librería se encuentra divide en los siguientes paquetes:

1.1.2.1 Módulos Base

Módulo de Enumeradores

Los enumeradores dentro de este módulo se utilizan para estandarizar la utilización de la librería entre los distintos usuarios y asi evitar diferencias en los nombres, códigos o parametrización utilizada.

A continuación se presenta un listado de los enumeradores contenidos en este módulo:

- *Localidad* (`Locality`): Enumerador de localizaciones.
- *Moneda* (`Currency`): Enumerador de monedas.
- *Tipos de Índices* (`IndexType`): Listado de tipos de índices financieros.
- *Índices Financieros* (`FinancialIndex`): Enumerador de Indices Financieros.
- *Máximo/Mínimo* (`MaxMin`): Enumerador de Máximo y Minimo.
- *Posición Financiera* (`FinancialPosition`): Enumerador de posición larga o corta.
- *Tipos de Flujo de Caja* (`CashflowType`): Enumerador de tipos de Flujo de Caja.
- *Fuente de Información* (`Source`): Enumerador de fuentes de información.
- *Tipos de Curva* (`CurveType`): Enumerador de tipos de curva.
- *Periodos* (`Period`): Enumerador de Periodos.
- *Periodicidad* (`Periodicity`): Enumerador de periodicidades.
- *Redondeo de Días* (`BusinessDay`): Enumerador de convenciones de redondeo de días hábiles.
- *Conteo de Días* (`DayCount`): Enumerador de convenciones de conteo de días.
- *Composición de tasas* (`Compounding`): Enumerador de convenciones de composición de tasas.

- *Métodos de Interpolación* (`InterpolationMethod`): Enumerador de métodos de interpolación.
- *Métodos de Extrapolación* (`ExtrapolationMethod`): Enumerador de métodos de extrapolación.

Localidad

class enums.**Locality**(*value*, *names=None*, ***, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Enumerador de localidades.

Lista las principales localidades financieras siguiendo el ISO 3166, cada localización tiene representación abreviada (letras) y un código numérico.

NDL = 0

Localidad No Determinada (Non Determined Location)

IMF = 999

Fondo Monetario Internacional(NON - ISO CODE)

USA = 840

Estados Unidos

CAN = 124

Canada

CHL = 152

Chile

PER = 604

Peru

COL = 170

Colombia

VEN = 862

Venezuela

ARG = 32

Argentina

BRA = 76

Brasil

MEX = 484

Mexico

EUR = 998

Union Europea

GBR = 826

Reino Unido

CHE = 756

Suiza

SWE = 752

Suecia

DNK = 208
Dinamarca

NOR = 578
Noruega

ZAF = 710
Sudáfrica

CN = 156
China

HKG = 344
Hong Kong

JPN = 392
Japón

KOR = 410
Corea del sur

AUS = 36
Australia

NZL = 554
Nueva Zelanda

Moneda

class enums.**Currency**(*value*, *names=None*, ***, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Enumerador de Moneda

Lista las monedas utilizando un Mnemotécnico de 3 letras.

NDC = 1
Non Determined Currency

USD = (2,)
Dólar Estadounidense

CAD = (3,)
Dólar Canadiense

DEG = (4,)
Derecho Especial de Giro

CLP = (5,)
Peso Chileno

CLD = (6,)
Peso Chileno a Dólar Observado

CLF = (7,)
Unidad de Fomento

PEN = (8,)
Sol Peruano

COP = (9,)
Peso Colombiano

BRL = (10,)
Real Brasilero

MXN = (11,)
Peso Mexicano

EUR = (12,)
Euro

GBP = (13,)
Libra Británica

CHF = (14,)
Franco Suizo

SEK = (15,)
Corona Sueca

DKK = (16,)
Corona Danesa

NOK = (17,)
Corana Noruega

JPY = (18,)
Yen Japonés

CNY = (19,)
Yuan Renminbi Chino

CNH = (20,)
Yuan Renminbi Chino

KRW = (21,)
Won Surcoreano

HKD = (22,)
Dólar Hong Kong

AUD = (23,)
Dólar Australiano

NZD = (24,)
Dólar Neozelandés

ZAR = (25,)
Rand Sudafricano

static from_str(*currency_str: str*)
Convertir String a Currency

Método estático que convierte un string en Currency

Parámetros
currency_str – String de moneda

Devuelve
Enumerador de moneda

Tipos de Índices

class enums.**IndexType**(*value*, *names=None*, ***, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Enumerador de tipos de índices

Enumera los distintos tipos de índices existentes

FIXED = 1
Fijo

MPR = 2
Monetary Policy Rate

ON = 3
Tasa Overnight

IBOR = 4
Interbancarias

CPI = 5
Indices de inflación

static from_str(*index_type_str: str*)
Convertir String a IndexType

Método estático que convierte un string en un enumerador de Tipo de Índice

Parámetros
index_type_str – String de Tipo de Índice

Devuelve
Enumerador de tipo de índice

Índices Financieros

class enums.**FinancialIndex**(*value*, *names=None*, ***, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Enumerador de Índices Financieros

Lista los principales índices financieros.

FIX = (1,)
Fijo

ICP = (2,)
Índice Camara Promedio

ICP_REAL = (3,)
Índice Camara Real

TAB1M = (4,)
TAB CLP 1M

TAB3M = (5,)
TAB CLP 3M

TAB6M = (6,)
TAB CLP 6M

TAB1Y = (7,)
TAB CLP 1Y

TABUF3M = (8,)
TAB UF 3M

TABUF6M = (9,)
TAB UF 6M

TABUF1Y = (10,)
TAB UF 1Y

OIS = (11,)
OIS

SOFR = (12,)
SOFR

LIBOR1M = (13,)
LIBOR 1M

LIBOR3M = (14,)
LIBOR 3M

LIBOR6M = (15,)
LIBOR 6M

LIBOR1Y = (16,)
LIBOR 1Y

TERM_SOFR1M = (17,)
TERM SOFR 1M

TERM_SOFR3M = (18,)
TERM SOFR 3M

TERM_SOFR6M = (19,)
TERM SOFR 6M

TERM_SOFR1Y = (20,)
TERM SOFR 1Y

EONIA = (21,)
EONIA

ESTR = (22,)
ESTR

EURIBOR1M = (23,)
EURIBOR 1M

EURIBOR3M = (24,)
EURIBOR 3M

EURIBOR6M = (25,)
EURIBOR 6M

EURIBOR1Y = (26,)
EURIBOR 1Y

IBR = (27,)
Índice Bancario de Referencia

TONA = (28,)
TONA Japonés

BBSW_3M = (29,)
Bank Bill 3M Australiano

SARON = (30,)
SARON Suizo

AONIA = 31
AONIA Australiano

static from_str(*index_str: str*)
Convertir String a Índice

Método estático que convierte un string en Índice Financiero

Parámetros
index_str – String de Índice Financiero

Devuelve
Enumerador de Índice Financiero

Muestra
IndexException – En caso no se reconozca el string

Máximo/Mínimo

class enums.**MaxMin**(*value*, *names=None*, ***, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)
Determinante de Máximo y Mínimo

Determina el Máximo (+1) y Minimo (-1)

MAX = (1,)
Máximo

MIN = -1
Minimo

Posición Financiera

class enums.**FinancialPosition**(*value*, *names=None*, ***, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Enumerador de posición financiera

Lista las posibles posiciones financieras, Long (1) y Short (-1)

LONG = 1

Largo

SHORT = -1

Corto

Tipos de Flujo de Caja

class enums.**CashflowType**(*value*, *names=None*, ***, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Enumerador de tipo de flujo de Caja

Lista los tipos de flujo de caja: Activo (1) y Pasivo (-1)

ACTIVE = 1

Activo

PASSIVE = -1

Pasivo

Fuente de Información

class enums.**Source**(*value*, *names=None*, ***, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Enumerador de Fuente de Información

Enumera las distintas fuentes de información

OFFICIAL = 1

Oficial

MUREX = 2

Murex

SANDBOX = 3

Sandbox

Tipos de Curva

class enums.**CurveType**(*value*, *names=None*, ***, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Enumerador de Tipos de Curva

Enumera los distintos tipos de curva.

DISCOUNT = 1

Curva de Descuento

PROJECTION = 2

Curva de Proyección

DUAL = 3

Curva de Descuento y Proyección

CREDIT_RISK = 4

Curva de Riesgo de Crédito

FIXED_INCOME = 5

Curva de Renta Fija

FUNDING_COST = 6

Curva de Costo de Fondo

NOT_USED = 7

Curva no usada

Periodos

class enums.**Period**(*value*, *names=None*, ***, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Enumerador de periodos

Enumera y estandariza la representación de los distintos tipos de periodo.

DAY = 1

Día

MONTH = 2

Mes

QUARTER = 3

Trimestre

SEMESTER = 4

Semestre

YEAR = 5

Año

Periodicidad

class enums.**Periodicity**(*value*, *names=None*, ***, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Enumerador de periodicidad

Enumera y estandariza la representación de las periodicidades.

UNIQUE = 0

Pago unico

ANNUAL = 1

Anual

BIANNUAL = 2

Semestral

QUARTERLY = 4

Trimestral

MONTHLY = 12

Mensual

WEEKLY = 52

Semanal

TN = 182

Cada dos días

OVERNIGHT = 365

Pago diario

ON = 365

Pago diario

TAILOR_MADE = 999

Pagos hechos a la medida

static from_str(*periodicity_str: str*)

Convertir String a Periodicity

Método estático que convierte un string en una Periodicidad

Parámetros
periodicity_str – String de periodicidad

Devuelve
Enumerador de periodicidad

Redondeo de Días

class enums.**BusinessDay**(*value*, *names=None*, ***, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Enumerador de redondeo de días hábiles

Enumera las distintas convenciones de redondeo de días hábiles.

UNADJUSTED = 1

Sin ajuste por fecha

FOLLOWING = 2

Convención Following

MODIFIED_FOLLOWING = 3

Convención Modified Following

PRECEDING = 4

Convención Preceding

MODIFIED_PRECEDING = 5

Convención Modified Preceding

END_OF_MONTH = 6

Fin de mes

PUBLISH_UF = 7

Convención UF

static **from_str**(*business_day_str: str*)

Convertir String a Business Day Convention

Método estático que convierte un string en un enumerador de convención de días laborales

Parámetros
business_day_str – String de Convención de días laborales

Devuelve
Enumerador de BusinessDay

Conteo de Días

class enums.**DayCount**(*value*, *names=None*, ***, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Enumerador de convenciones de conteo de días

Enumera las distintas convenciones de conteo de días.

DC_30_360 = 1

Convención 30/360

DC_30_365 = 2

Convención 30/365

DC_30E_360 = 3

Convención 30/360 Europeo

DC_30E_365 = 4

Convención 30/365 Europeo

DC_30E_360_ISDA = 5

Convención 30/360 Europeo ISDA

DC_30E_365_ISDA = 6

Convención 30/365 Europeo ISDA

DC_ACT_360 = 7

Convención Act/360

DC_ACT_365 = 8

Convención Act/365

static from_str(*day_count_str: str*)

Convertir String a DayCount

Método estático que convierte un string en un enumerador de conteo de días

Parámetros
day_count_str – String de Conteo de días

Devuelve
Enumerador de conteo de días

Composición de tasas

class enums.**Compounding**(*value*, *names=None*, ***, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Enumerador para composición de tasas

Enumera las distintas convenciones de composición de tasas

LINEAR = 1

Lineal

YIELD = 2

Compuesta

EXPONENTIAL = 3

Exponencial

static from_str(*compounding_str: str*)

Convertir String a IndexType

Método estático que convierte un string en un enumerador de Tipo de Índice

Parámetros
compounding_str – String de Tipo de Índice

Devuelve
Enumerador de Compounding

Métodos de Interpolación

class enums.**InterpolationMethod**(*value*, *names=None*, ***, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Enumerador para métodos de interpolación

Enumera los métodos de interpolación implementados.

LINEAR = 1

Lineal

LOG_LINEAR = 2

Log Lineal

static from_str(*interpolation_str: str*)

Convertir String a InterpolationMethod

Método estático que convierte un string en un enumerador de método de interpolación

Parámetros
interpolation_str – String de Método de interpolación

Devuelve
Enumerador de método de interpolación

Métodos de Extrapolación

class enums.**ExtrapolationMethod**(*value*, *names=None*, ***, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Enumerador para métodos de extrapolación

Enumera los métodos de extrapolación implementados

SLOPE = 1

Mantiene la pendiente

LOG_SLOPE = 2

Mantiene la pendiente del logaritmo

RATE_SLOPE = 3

Mantiene la pendiente convirtiendo a una tasa convención Yield ACT/360

FLAT = 4

Mantiene el valor mas cercano

FLAT_RATE = 5

Mantiene el valor mas cercano tratándolo como tasa

static from_str(*extrapolation_str: str*)

Convertir String a InterpolationMethod

Método estático que convierte un string en un enumerador de método de interpolación

Parámetros
extrapolation_str – String de Método de interpolación

Devuelve
Enumerador de método de extrapolación

Módulo de Excepciones

Las excepciones contenidas en este módulo fueron diseñadas para manejar los errores internos de la librería. Estos errores pueden ser causados por multiples razones entre ellas: tipo equivocado de parámetros, operación inválida, falta de información disponible, etc.

A continuación se presenta un listado de las excepciones contenidas en este módulo:

- *Excepciones de Enumeradores* (`EnumException`): Maneja errores asociados a Enumeradores.
- *Excepciones de Moneda* (`CurrencyException`): Maneja errores asociados a Monedas.
- *Excepciones de Tasa de Cambio* (`FxRateException`): Maneja errores asociados a Tasas de Cambio.
- *Excepciones de Calendario* (`CalendarException`): Maneja errores asociados a calendarios de negociación.
- *Excepciones de Curva* (`CalendarException`): Maneja errores asociados con curvas.
- *Excepciones de Índices* (`IndexException`): Maneja errores asociados con índices financieros.
- *Excepciones de Dataset* (`DatasetException`): Maneja errores asociados con Dataset.
- *Excepciones de Portafolio* (`PortfolioException`): Maneja errores asociados con Portafolio.
- *Excepciones de Lectura de Archivo* (`FileReadException`): Maneja errores asociados con Lectura de Archivos.
- *Excepciones de Consulta* (`QueryException`): Maneja errores asociados con consultas a base de datos.

Excepciones de Enumeradores

class exceptions.**EnumException**(*message: str*)

Excepción de Enumerador

Maneja los errores asociados con enumeradores.

Parámetros
message (*str*) – Mensaje

Excepciones de Moneda

class exceptions.**CurrencyException**(*message: str*, *currency: str = ''*)

Excepción de Moneda

Maneja los errores asociados con Monedas.

Parámetros
message (*str.*) – Mensaje.

Excepciones de Tasa de Cambio

class exceptions.**FxRateException**(*fx_pair_name: str*, *message: str*)

Excepción de tasa de cambio

Maneja errores asociados a la tasa de cambio.

Parámetros

- **fx_pair_name** (*str.*) – Tasa de Cambio.
- **message** (*str*) – Mensaje.

Excepciones de Calendario

class exceptions.**CalendarException**(*message: str*)

Excepción de Calendario

Maneja los errores en la aplicación de calendarios.

Parámetros
message (*str.*) – Mensaje.

Excepciones de Curva

class exceptions.**CurveException**(*curve_name: str*, *message: str*)

Excepción de Curva

Maneja los errores en la construcción o utilización de curvas.

Parámetros

- **curve_name** (*str.*) – Nombre de la curva.
- **message** (*str.*) – Mensaje.

Excepciones de Índices

class exceptions.**IndexException**(*idx_name: str*, *message: str*)

Excepción de Índice

Maneja errores asociados a índices.

Parámetros

- **idx_name** (*str.*) – Nombre del Índice.
- **message** (*str.*) – Mensaje.

Excepciones de Dataset

class exceptions.**DatasetException**(*message: str*)

Excepción de DataSet

Maneja los errores en Datasets.

Parámetros

- **message** – Mensaje.
- **message** – str.

Excepciones de Portafolio

class exceptions.**PortfolioException**(*message: str*)

Excepción de Portfolio

Maneja los errores en Portafolios.

Parámetros

- **message** – Mensaje.
- **message** – str.

Excepciones de Lectura de Archivo

class exceptions.**FileReadException**(*file_path: str*, *file_alias: str*, *message: str*)

Excepción de Lectura de archivos

Maneja los errores en la lectura de archivos.

Parámetros

- **file_path** (*str*) – Localización del archivo.
- **file_alias** (*str.*) – Nombre o Alias del archivo.
- **message** (*str.*) – Mensaje.

Excepciones de Consulta

class exceptions.**QueryException**(*query: str*, *message: str*)

Excepción de consulta

Maneja los errores de queries en la base de datos.

Parámetros

- **query** (*str.*) – Consulta SQL.
- **message** (*str*) – Mensaje.

Módulo de Configuración

Módulo dedicado contener las principales configuraciones de la librería

finriskconfig.csa_clients = {'0970060006': 'BANCO DE CREDITO E INVERSIONES', '0970180001': 'SCOTIABANK CHILE', '0970300007': 'BANCO DEL ESTADO DE CHILE', '097036000K': 'BANCO SANTANDER CHILE', '0970430008': 'JP MORGAN CHASE BANK', '200508258K': 'CHICAGO MERCANTILE EXCHANGE - CME', '4038302766': 'BARCLAYS BANK PLC LONDRES', '4042672403': 'BBVA ESPAÑA', '4076552687': 'BNP PARIBAS FRANCIA', '4078408281': 'BANK OF AMERICA NA', '408404268K': 'BNP PARIBAS - FRANCIA', '4159492767': 'DEUTSCHE BANK UK', '4525472766': 'JP MORGAN CHASE BANK NA UK', '4526122769': 'CITIBANK NA LONDON', '4531832766': 'HSBC BANK PLC', '453423276K': 'MERRILL LYNCH INTERNATIONAL', '4537958281': 'HSBC USA', '4563121365': 'THE BANK OF NOVA SCOTIA (SCOTIABANK)', '4586912769': 'GOLDMAN SACHS INTL LONDON', '4591652767': 'LCHCLEARNET LIMITED', '4631808285': 'GOLDMAN SACHS USA', '4661572766': 'MORGAN STANLEY COINTLPLC', '480256200K': 'UBS - SUIZA', '4959212762': 'WELLS FARGO BNAK LONDON', '828789007': 'COOPERATIVA DE AHORRO Y CREDITO COOPEUCH', '964890005': 'CREDICORP CAPITAL CORREDORES DE BOLSA SPA', '965096604': 'BANCO FALABELLA SA', '970060006': 'BANCO DE CREDITO E INVERSIONES', '970110003': 'BANCO INTERNACIONAL', '970180001': 'SCOTIABANK CHILE', '970300007': 'BANCO DEL ESTADO DE CHILE', '97036000K': 'BANCO SANTANDER CHILE', '970430008': 'JP MORGAN CHASE BANK', '995004100': 'BANCO CONSORCIO'}

Pares de Monedas

finriskconfig.fx_pair_config = {'CADUSD': {'calendar': [USA], 'discount days': 1, 'primary': 'CAD', 'secondary': 'USD'}, 'CLFCLP': {'calendar': [CHL], 'discount days': 0, 'primary': 'CLF', 'secondary': 'CLP'}, 'CLPCLF': {'calendar': [CHL], 'discount days': 0, 'primary': 'CLP', 'secondary': 'CLF'}, 'CLPUSD': {'calendar': [CHL, USA], 'discount days': 1, 'primary': 'CLP', 'secondary': 'USD'}, 'COPCLP': {'calendar': [CHL, COL], 'discount days': 2, 'primary': 'COP', 'secondary': 'CLP'}, 'EURUSD': {'calendar': [USA], 'discount days': 2, 'primary': 'EUR', 'secondary': 'USD'}, 'GBPUSD': {'calendar': [USA], 'discount days': 2, 'primary': 'GBP', 'secondary': 'USD'}, 'USDAUD': {'calendar': [USA], 'discount days': 2, 'primary': 'USD', 'secondary': 'AUD'}, 'USDBRL': {'calendar': [USA], 'discount days': 2, 'primary': 'USD', 'secondary': 'BRL'}, 'USDCAD': {'calendar': [USA], 'discount days': 1, 'primary': 'USD', 'secondary': 'CAD'}, 'USDCHF': {'calendar': [USA], 'discount days': 2, 'primary': 'USD', 'secondary': 'CHF'}, 'USDCLP': {'calendar': [CHL, USA], 'discount days': 1, 'primary': 'USD', 'secondary': 'CLP'}, 'USDCNH': {'calendar': [USA], 'discount days': 2, 'primary': 'USD', 'secondary': 'CNH'}, 'USDCNY': {'calendar': [USA], 'discount days': 2, 'primary': 'USD', 'secondary': 'CNY'}, 'USDCOP': {'calendar': [USA], 'discount days': 2, 'primary': 'USD', 'secondary': 'COP'}, 'USDDKK': {'calendar': [USA], 'discount days': 2, 'primary': 'USD', 'secondary': 'DKK'}, 'USDJPY': {'calendar': [USA], 'discount days': 2, 'primary': 'USD', 'secondary': 'JPY'}, 'USDKRW': {'calendar': [USA], 'discount days': 2, 'primary': 'USD', 'secondary': 'KRW'}, 'USDMXN': {'calendar': [USA], 'discount days': 2, 'primary': 'USD', 'secondary': 'MXN'}, 'USDNOK': {'calendar': [USA], 'discount days': 2, 'primary': 'USD', 'secondary': 'NOK'}, 'USDNZD': {'calendar': [USA], 'discount days': 2, 'primary': 'USD', 'secondary': 'NZD'}, 'USDPEN': {'calendar': [USA], 'discount days': 2, 'primary': 'USD', 'secondary': 'PEN'}, 'USDSEK': {'calendar': [USA], 'discount days': 2, 'primary': 'USD', 'secondary': 'SEK'}, 'USDZAR': {'calendar': [USA], 'discount days': 2, 'primary': 'USD', 'secondary': 'ZAR'}}

Indices Financieros

finriskconfig.index_bbg_codes = {ICP: 'CLICP INDEX', ICP_REAL: 'CLICREAL INDEX', TAB6M: 'CLTN180N INDEX', TAB1M: 'CLTN30DN INDEX', TAB1Y: 'CLTN360N INDEX', TAB3M: 'CLTN90DN INDEX', EURIBOR1M: 'EUR001M INDEX', EURIBOR3M: 'EUR003M INDEX', EURIBOR6M: 'EUR006M INDEX', EURIBOR1Y: 'EUR012M INDEX', TABUF6M: 'PCRR180D INDEX', TABUF1Y: 'PCRR360D INDEX', TABUF3M: 'PCRR90D INDEX', SOFR: 'SOFRRATE INDEX', TERM_SOFR1Y: 'TSFR12M INDEX', TERM_SOFR1M: 'TSFR1M INDEX', TERM_SOFR3M: 'TSFR3M INDEX', TERM_SOFR6M: 'TSFR6M INDEX', LIBOR1M: 'US0001M INDEX', LIBOR3M: 'US0003M INDEX', LIBOR6M: 'US0006M INDEX', LIBOR1Y: 'US0012M INDEX'}

Configuración de Curvas

```
finriskconfig.index_codes = {'CLF TAB 1Y': FinancialIndex.TABUF1Y, 'CLF TAB 3M':
FinancialIndex.TABUF3M, 'CLF TAB 6M': FinancialIndex.TABUF6M, 'CLF TRA':
FinancialIndex.ICP_REAL, 'CLICP INDEX': FinancialIndex.ICP, 'CLICREAL INDEX':
FinancialIndex.ICP_REAL, 'CLP TAB 1M': FinancialIndex.TAB1M, 'CLP TAB 1Y':
FinancialIndex.TAB1Y, 'CLP TAB 3M': FinancialIndex.TAB3M, 'CLP TAB 6M':
FinancialIndex.TAB6M, 'CLP TNA': FinancialIndex.ICP, 'CLTN180N INDEX':
FinancialIndex.TAB6M, 'CLTN30DN INDEX': FinancialIndex.TAB1M, 'CLTN360N INDEX':
FinancialIndex.TAB1Y, 'CLTN90DN INDEX': FinancialIndex.TAB3M, 'EUR EURIBOR 1M':
FinancialIndex.EURIBOR1M, 'EUR EURIBOR 1Y': FinancialIndex.EURIBOR1Y, 'EUR EURIBOR 3M':
FinancialIndex.EURIBOR3M, 'EUR EURIBOR 6M': FinancialIndex.EURIBOR6M, 'EUR001M INDEX':
FinancialIndex.EURIBOR1M, 'EUR003M INDEX': FinancialIndex.EURIBOR3M, 'EUR006M INDEX':
FinancialIndex.EURIBOR6M, 'EUR012M INDEX': FinancialIndex.EURIBOR1Y, 'EURIBOR 12M':
FinancialIndex.EURIBOR1Y, 'EURIBOR 180': FinancialIndex.EURIBOR6M, 'EURIBOR 1M':
FinancialIndex.EURIBOR1M, 'EURIBOR 3M': FinancialIndex.EURIBOR3M, 'EURIBOR 6M':
FinancialIndex.EURIBOR6M, 'EURIBOR180': FinancialIndex.EURIBOR6M, 'EURIBOR6M':
FinancialIndex.EURIBOR6M, 'FIJA': FinancialIndex.FIX, 'FIX': FinancialIndex.FIX, 'ICP':
FinancialIndex.ICP, 'ICP REAL': FinancialIndex.ICP_REAL, 'LIBOR 12M':
FinancialIndex.LIBOR1Y, 'LIBOR 180': FinancialIndex.LIBOR6M, 'LIBOR 1M':
FinancialIndex.LIBOR1M, 'LIBOR 30': FinancialIndex.LIBOR1M, 'LIBOR 360':
FinancialIndex.LIBOR1Y, 'LIBOR 3M': FinancialIndex.LIBOR3M, 'LIBOR 6M':
FinancialIndex.LIBOR6M, 'LIBOR 90': FinancialIndex.LIBOR3M, 'LIBOR US 12M':
FinancialIndex.LIBOR1Y, 'LIBOR US 1M': FinancialIndex.LIBOR1M, 'LIBOR US 3M':
FinancialIndex.LIBOR3M, 'LIBOR US 6M': FinancialIndex.LIBOR6M, 'LIBOR180':
FinancialIndex.LIBOR6M, 'LIBOR30': FinancialIndex.LIBOR1M, 'LIBOR360':
FinancialIndex.LIBOR1Y, 'LIBOR90': FinancialIndex.LIBOR3M, 'OIS': FinancialIndex.SOFR,
'PCRR180D INDEX': FinancialIndex.TABUF6M, 'PCRR360D INDEX': FinancialIndex.TABUF1Y,
'PCRR90D INDEX': FinancialIndex.TABUF3M, 'SOFR': FinancialIndex.SOFR, 'SOFRRATE INDEX':
FinancialIndex.SOFR, 'TAB 1M': FinancialIndex.TAB1M, 'TAB 1Y': FinancialIndex.TAB1Y, 'TAB
3M': FinancialIndex.TAB3M, 'TAB 6M': FinancialIndex.TAB6M, 'TAB CLF 180':
FinancialIndex.TABUF6M, 'TAB CLF 1Y': FinancialIndex.TABUF1Y, 'TAB CLF 360':
FinancialIndex.TABUF1Y, 'TAB CLF 3M': FinancialIndex.TABUF3M, 'TAB CLF 6M':
FinancialIndex.TABUF6M, 'TAB CLF 90': FinancialIndex.TABUF3M, 'TAB CLP 180':
FinancialIndex.TAB6M, 'TAB CLP 1M': FinancialIndex.TAB1M, 'TAB CLP 1Y':
FinancialIndex.TAB1Y, 'TAB CLP 30': FinancialIndex.TAB1M, 'TAB CLP 360':
FinancialIndex.TAB1Y, 'TAB CLP 3M': FinancialIndex.TAB3M, 'TAB CLP 6M':
FinancialIndex.TAB6M, 'TAB CLP 90': FinancialIndex.TAB3M, 'TAB30': FinancialIndex.TAB1M,
'TAB30.CLP': FinancialIndex.TAB1M, 'TAB360.CLF': FinancialIndex.TABUF1Y, 'TABUF 1Y':
FinancialIndex.TABUF1Y, 'TABUF 3M': FinancialIndex.TABUF3M, 'TABUF 6M':
FinancialIndex.TABUF6M, 'TERM SOFR 12M': FinancialIndex.TERM_SOFR1Y, 'TERM SOFR 1M':
FinancialIndex.TERM_SOFR1M, 'TERM SOFR 3M': FinancialIndex.TERM_SOFR3M, 'TERM SOFR 6M':
FinancialIndex.TERM_SOFR6M, 'TSFR12M INDEX': FinancialIndex.TERM_SOFR1Y, 'TSFR1M INDEX':
FinancialIndex.TERM_SOFR1M, 'TSFR3M INDEX': FinancialIndex.TERM_SOFR3M, 'TSFR6M INDEX':
FinancialIndex.TERM_SOFR6M, 'US0001M INDEX': FinancialIndex.LIBOR1M, 'US0003M INDEX':
FinancialIndex.LIBOR3M, 'US0006M INDEX': FinancialIndex.LIBOR6M, 'US0012M INDEX':
FinancialIndex.LIBOR1Y, 'USD LIBOR 1M': FinancialIndex.LIBOR1M, 'USD LIBOR 1Y':
FinancialIndex.LIBOR1Y, 'USD LIBOR 3M': FinancialIndex.LIBOR3M, 'USD LIBOR 6M':
FinancialIndex.LIBOR6M, 'USD SOFR 1M': FinancialIndex.TERM_SOFR1M, 'USD SOFR 1Y':
FinancialIndex.TERM_SOFR1Y, 'USD SOFR 3M': FinancialIndex.TERM_SOFR3M, 'USD SOFR 6M':
FinancialIndex.TERM_SOFR6M, 'USD SOFR CMP': FinancialIndex.SOFR, 'USD SOFR CMP -2BD':
FinancialIndex.SOFR, 'USD SOFR CMP 6M FALLBACK': FinancialIndex.SOFR}
```

Códigos BBG

finriskconfig.**index_config** = {FIX: {'type': 'ON', 'period': 'ON', 'count': 'DC_ACT_360', 'compound': 'LINEAR'}, ICP: {'type': 'ON', 'period': 'ON', 'count': 'DC_ACT_360', 'compound': 'LINEAR'}, ICP_REAL: {'type': 'ON', 'period': 'ON', 'count': 'DC_ACT_360', 'compound': 'LINEAR'}, TAB1M: {'type': 'IBOR', 'period': 'MONTHLY', 'count': 'DC_ACT_360', 'compound': 'LINEAR'}, TAB3M: {'type': 'IBOR', 'period': 'QUARTERLY', 'count': 'DC_ACT_360', 'compound': 'LINEAR'}, TAB6M: {'type': 'IBOR', 'period': 'BIANNUAL', 'count': 'DC_ACT_360', 'compound': 'LINEAR'}, TAB1Y: {'type': 'IBOR', 'period': 'ANNUAL', 'count': 'DC_ACT_360', 'compound': 'LINEAR'}, TABUF3M: {'type': 'IBOR', 'period': 'QUARTERLY', 'count': 'DC_ACT_360', 'compound': 'LINEAR'}, TABUF6M: {'type': 'IBOR', 'period': 'BIANNUAL', 'count': 'DC_ACT_360', 'compound': 'LINEAR'}, TABUF1Y: {'type': 'IBOR', 'period': 'ANNUAL', 'count': 'DC_ACT_360', 'compound': 'LINEAR'}, OIS: {'type': 'ON', 'period': 'ON', 'count': 'DC_ACT_360', 'compound': 'LINEAR'}, SOFR: {'type': 'ON', 'period': 'ON', 'count': 'DC_ACT_360', 'compound': 'LINEAR'}, LIBOR1M: {'type': 'IBOR', 'period': 'MONTHLY', 'count': 'DC_ACT_360', 'compound': 'LINEAR'}, LIBOR3M: {'type': 'IBOR', 'period': 'QUARTERLY', 'count': 'DC_ACT_360', 'compound': 'LINEAR'}, LIBOR6M: {'type': 'IBOR', 'period': 'BIANNUAL', 'count': 'DC_ACT_360', 'compound': 'LINEAR'}, LIBOR1Y: {'type': 'IBOR', 'period': 'ANNUAL', 'count': 'DC_ACT_360', 'compound': 'LINEAR'}, TERM_SOFR1M: {'type': 'IBOR', 'period': 'MONTHLY', 'count': 'DC_ACT_360', 'compound': 'LINEAR'}, TERM_SOFR3M: {'type': 'IBOR', 'period': 'QUARTERLY', 'count': 'DC_ACT_360', 'compound': 'LINEAR'}, TERM_SOFR6M: {'type': 'IBOR', 'period': 'BIANNUAL', 'count': 'DC_ACT_360', 'compound': 'LINEAR'}, TERM_SOFR1Y: {'type': 'IBOR', 'period': 'ANNUAL', 'count': 'DC_ACT_360', 'compound': 'LINEAR'}, EONIA: {'type': 'ON', 'period': 'ON', 'count': 'DC_ACT_360', 'compound': 'LINEAR'}, ESTR: {'type': 'ON', 'period': 'ON', 'count': 'DC_ACT_360', 'compound': 'LINEAR'}, EURIBOR1M: {'type': 'IBOR', 'period': 'MONTHLY', 'count': 'DC_ACT_360', 'compound': 'LINEAR'}, EURIBOR3M: {'type': 'IBOR', 'period': 'QUARTERLY', 'count': 'DC_ACT_360', 'compound': 'LINEAR'}, EURIBOR6M: {'type': 'IBOR', 'period': 'BIANNUAL', 'count': 'DC_ACT_360', 'compound': 'LINEAR'}, EURIBOR1Y: {'type': 'IBOR', 'period': 'ANNUAL', 'count': 'DC_ACT_360', 'compound': 'LINEAR'}, IBR: {'type': 'ON', 'period': 'ON', 'count': 'DC_ACT_360', 'compound': 'LINEAR'}, TONA: {'type': 'ON', 'period': 'ON', 'count': 'DC_ACT_360', 'compound': 'LINEAR'}, BBSW_3M: {'type': 'IBOR', 'period': 'QUARTERLY', 'count': 'DC_ACT_360', 'compound': 'LINEAR'}, SARON: {'type': 'ON', 'period': 'ON', 'count': 'DC_ACT_360', 'compound': 'LINEAR'}, AONIA: {'type': 'ON', 'period': 'ON', 'count': 'DC_ACT_360', 'compound': 'LINEAR'}}

Códigos de Indices

Módulo de Decoradores

decorators.**measure_performance**()

Medir performance

Mide la performance de una función

Devuelve

Performance de la función

Los módulos base se encuentran dentro del namespace de la librería, esto permite que sean fácilmente accesibles por todos los demás módulos y funciones.

Su principal función es la de estandarizar los nombres y conceptos, para esto contamos con 4 módulos:

Módulo de Enumeradores

Colección de enumeradores que estandarizan los conceptos y códigos utilizados dentro de la librería.

Módulo de Excepciones

Colección de excepciones hechas a la medida para manejar los distintos errores que puede generar la librería.

Módulo de Configuración

Colección de configuraciones utilizadas para leer archivos o bases de datos, conversión de curvas, etc.

Módulo de Decoradores

Colección de decoradores de funciones.

1.1.2.2 Paquete de Datos

El paquete de datos se encuentra dedicado al manejo y extracción de datos desde servidores como desde archivos con formato especifico.

La finalidad del paquete es la estandarización de la obtención de datos, al igual que centralizar y simplificar la traducción de archivos y consultas SQL en objetos nativos de la librería.

Las módulos que se se encuentran en el paquete son:

Módulo de Conexión

Clases dedicada a la conexión y obtención de datos abstractos desde servidores SQL.

Módulo Parser

Clase dedicada a la conversión de datos en objetos nativos de la librería.

Módulo de Lectura Archivos

Clase dedicada a la lectura de datos desde archivos con formato estandarizado.

Módulo Consultas SQL

Clase dedicada a la lectura de datos desde los servidores internos.

Módulo de Conexión

Este módulo está enfocado en la creación de conexiones a servidores SQL.

La principal clase de este módulo es:

- *Conexión SQL* (*SQLConn*): Clase dedicada al manejo de conexiones a servidores SQL.

Conexión SQL

class data.dbconnection.**SQLConn**(*ip: str*, *database: str*, *username: str = None*, *password: str = None*)

Conexión SQL

Se encarga de generar conexiones y ejecutar consultas a bases de datos

Parámetros

- **ip** (*str.*) – IP del servidor.
- **database** (*str.*) – Nombre de la base de datos.
- **username** (*str.*) – Nombre de usuario. (default None)
- **password** (*str.*) – Password. (default None)

get_conn_string() → str

Obtener cadena de conexión

Genera una cadena de conexión standard utilizando los datos provistos. En caso no se haya provisto un nombre de usuario se genera una cadena por Trusted Connection.

Devuelve
Cadena de conexión

execute_query(*query: str*) → DataFrame

Ejecutar Query

Ejecuta la query proporcionada utilizando la conexión interna.

Parámetros
query (*str.*) – Query a ser ejecutada.

Devuelve
Dataframe con resultado de la query

save_data(*table_name: str*, *data: DataFrame*)

Guardar Datos

Guarda los datos proporcionados en la tabla asignada.

Parámetros

- **table_name** (*str*) – Nombre de la tabla donde se guardará la información.
- **data** (*Dataframe*) – Dataframe con datos a guardar.

static get_b08_conn()

Obtener conexión a servidor B08

Obtiene la conexión a la base de Datos B08

Devuelve
SQLConn

Módulo Parser

Este módulo está dedicado a la conversión de datos con formatos establecidos a objetos propios de la librería financiera.

Cuenta con una clase estática principal:

- *Conversor de Data* (`DataParser`): Clase estática con funciones de conversión de datos.

Conversor de Data

class data.dataparser.**DataParser**

Clase estática dedicada a la conversión de datos (DataFrames) con estructuras definidas en objetos propios de la librería financiera

static parse_curve_data(*curve_data: DataFrame*) → CurveDataSet

Convertir data de curvas

Convierte un DataFrame con los datos

Tabla 1: **Formato Tabla de Curvas**

Fecha Proceso	Código Curva	Tenor	Factor de descuento
2022/01/03	CURVA_CLP_CL	1	0.998

Parámetros
curve_data (*DataFrame*) – Data de curvas

Devuelve
CurveDataSet

static parse_fx_data(*fx_data: DataFrame*) → FxDataSet

Convertir data de tipos de cambio

Convierte un DataFrame con los datos

Tabla 2: **Formato Tabla de Tipo de Cambio**

Fecha Proceso	Moneda Primaria	Moneda Secundaria	Tipo de Cambio
2022/01/03	USD	CLP	850

Parámetros
fx_data (*DataFrame*) – Data de curvas

Devuelve
FxDataSet

static parse_index_data(*index_data: DataFrame*) → IndexDataSet

Convertir data de índices financieros

Convierte un DataFrame con los datos

Tabla 3: **Formato Tabla de Índices Financieros**

Código Índice	Fecha Proceso	Valor
ICP REAL	2022/01/03	17854.15

Parámetros
index_data (*DataFrame*) – Data de índices

Devuelve
IndexDataSet

static parse_portfolio_data(*portfolio_data: DataFrame*) → Portfolio

Convertir data de portafolio

Convierte un dataframe con la estructura de cartera en un objeto Portfolio

Parámetros
portfolio_data (*DataFrame*) – Data de portfolio

Devuelve
Portfolio

static parse_operation_data(*operation_data: DataFrame*) → Derivative

Convertir data de operación

Convierte un dataframe con la estructura de la cartera en un objeto de instrumento.

Parámetros
operation_data (*DataFrame*) – Data de la operación

Devuelve
Derivative

static parse_grf_fwd_operation_data(*operation_data: DataFrame*, *compensation_currency: DataFrame*) → Derivative

Convertir data de operación forward desde interfaz GRF

Convierte un dataframe con la estructura de la cartera en un objeto de instrumento.

Parámetros

- **operation_data** (*DataFrame*) – Data de la operación
- **compensation_currency** (*DataFrame*) – Moneda de Compensación

Devuelve
Derivative

static parse_grf_fwd_portfolio_data(*portfolio_data: DataFrame*, *compensation_currency: DataFrame*) → Portfolio

Convertir data de portafolio

Convierte un dataframe con la estructura de cartera en un objeto Portfolio

Parámetros

- **portfolio_data** (*DataFrame*) – Data de portfolio
- **compensation_currency** (*DataFrame*) – Moneda de compensación

Devuelve
Portfolio

static parse_grf_swap_operation_data(*operation_data: DataFrame*) → Derivative

Convertir data de operación forward desde interfaz GRF

Convierte un dataframe con la estructura de la cartera en un objeto de instrumento.

Parámetros
operation_data (*DataFrame*) – Data de la operación

Devuelve
Derivative

static parse_grf_swap_portfolio_data(*portfolio_data: DataFrame*) → Portfolio

Convertir data de portafolio

Convierte un dataframe con la estructura de cartera en un objeto Portfolio

Parámetros
portfolio_data (*DataFrame*) – Data de portfolio

Devuelve
Portfolio

Módulo de Lectura Archivos

Este módulo está enfocado en la lectura de archivos que puedan ser entendidos y procesados por la librería. Los archivos deben tener un formato específico, el cual será especificado en la documentación de cada función.

La clase principal de este módulo es:

- *Lector Archivos Excel* (*XlsReader*): Lector de archivos Excel.

Lector Archivos Excel

class data.filereader.**XlsReader**

Clase dedicada a la lectura de datos en formato Excel.

static read_curve_data(*file_name: str*) → CurveDataSet

Leer de archivo de curvas

Lee un archivo excel con datos de una o más curvas y devuelve un diccionario con los nombres y las curvas inicializadas. El formato aceptado es:

Tabla 4: **Formato Tabla de Curvas**

Fecha Proceso	Código Curva	Tenor	Factor de descuento
2022/01/03	CURVA_CLP_CL	1	0.998

Parámetros
file_name (*str*) – Nombre del archivo a leer.

Devuelve
CurveDataSet

static read_fx_pair_data(*file_name: str*) → FxDataSet

Lector de archivo Excel de Tipos de cambio

Lee un archivo excel con datos de uno o más tipos de cambio para un día específico y devuelve una lista de objetos FxPair. El formato aceptado es:

Tabla 5: **Formato Tabla de Tipo de Cambio**

Fecha Proceso	Moneda Primaria	Moneda Secundaria	Tipo de Cambio
2022/01/03	USD	CLP	850

Parámetros
file_name (*str*) – Nombre del archivo a leer.

Devuelve
Diccionario de objetos FxPair

static read_index_data(*file_name: str*) → IndexDataSet

Lector de archivo Excel de Data de índices

Lee un archivo excel con datos de uno o más índices con información histórica y devuelve un diccionario con llave de objeto Index y valores históricos. El formato aceptado es:

Tabla 6: **Formato Tabla de Índices Financieros**

Código Indice	Fecha Proceso	Valor
ICP REAL	2022/01/03	17854.15

Parámetros
file_name (*str*) – Nombre del archivo a leer.

Devuelve
Diccionario de objetos Index

static read_dataset_data(*process_date: date*, *curve_file: str*, *fx_file: str*, *index_file: str*) → DataSet

Lector de dataset

Lee la información de mercado provista en los distintos archivos.

Parámetros

- **process_date** (*date.*) – Fecha del dataset.
- **curve_file** (*str.*) – Nombre del archivo con información de curvas.
- **fx_file** (*str.*) – Nombre del archivo con información de tipo de cambio.
- **index_file** (*str.*) – Nombre del archivo con información de índices.

Devuelve
Objeto Dataset.

static read_derivados_data(*portfolio_file: str*)

Leer Archivo de Operaciones

Lee un archivo con el formato de Cartera Derivados

Parámetros
portfolio_file – Dirección de archivo de portafolio

Devuelve
Portafolio

static read_derivados_operation_data(*operation_file: str*)

Leer Archivo de Operación Individual

Lee un archivo con el formato de Cartera Derivados de una sola operación.

Parámetros
operation_file – Dirección de archivo de portafolio

Devuelve
Derivative

static read_portfolio_data(*portfolio_file: str*)

Leer archivo de portafolio completo

Obtiene todos los productos de la cartera

Parámetros
portfolio_file – Dirección de archivo de portafolio

Devuelve
Portfolio

Módulo Consultas SQL

Este módulo se encuentra enfocado en estandarizar la obtención de datos desde los servidores productivos.

La clase principal de este módulo es:

- *Lector SQL* (`XlsReader`): Clase de lectura de datos SQL desde servidor productivo.

Lector SQL

class data.dbreader.**DBReader**(*conn: ~finrisklib.data.dbconnection.SQLConn = <finrisklib.data.dbconnection.SQLConn object>*)

Lector de base de datos

Clase dedicada a la lectura de información desde las bases de datos.

Parámetros
conn (SQLConn) – Conexión a BD. (Default: B08)

execute_query(*query: str*)

Ejecutar Query

Ejecuta la query solicitada

Parámetros
query – Query

Devuelve
Dataframe con datos de query ejecutada

get_curve_names(*process_date: date*, *source: Source = Source.OFFICIAL*) → list

Obtener nombre de curvas

Obtiene una lista con los nombres de curvas disponibles.

Parámetros

- **process_date** (*date*) – fecha de proceso.
- **source** (Source) – Fuente de información (Default = OFFICIAL).

Devuelve
Lista con nombres de curvas.

Muestra
QueryException – Cuando no se retorna valores con la query realizada.

get_single_curve_data(*curve_name: str*, *process_date: date*, *source: Source = Source.OFFICIAL*) → Curve

Obtener Curva

Obtiene un objeto curva en específico.

Parámetros

- **curve_name** (*str*) – Nombre de la curva.
- **process_date** (*date*) – fecha de proceso.
- **source** (Source) – Fuente de información (Default = OFFICIAL).

Devuelve
Objeto Curva con la información del servidor.

Muestra
QueryException – Cuando no se retorna valores con la query realizada.

get_curve_dataset(*process_date: date*, *source: Source = Source.OFFICIAL*) → CurveDataSet

Obtener todas las Curvas

Obtiene un diccionario con todas las curvas disponibles a una fecha determinada.

Parámetros

- **process_date** (*date.*) – Fecha de las curvas.
- **source** (Source) – Fuente de información (Default = OFFICIAL).

Devuelve
Diccionario con objetos Curva con la información del servidor.

Muestra
QueryException – Cuando no se retorna valores con la query realizada.

get_fx_rate(*primary_currency: str*, *secondary_currency: str*, *process_date: date*, *source: Source = Source.MUREX*) → float

Obtener tasa de cambio

Retorna el tipo de cambio solicitado para la fecha especificada

Parámetros

- **primary_currency** (*str.*) – Nombre de la moneda primaria.

- **secondary_currency** (*str.*) – Nombre de la moneda secundaria.
- **process_date** (*date.*) – Fecha de proceso.
- **source** (Source) – Fuente de Información (Default = MUREX).

Devuelve
Valor de la moneda

get_fx_dataset(*process_date: date*, *source: Source = Source.MUREX*) → FxDataSet

Obtener todas las tasas de cambio

Retorna los tipos de cambio para la fecha especificada

Parámetros

- **process_date** (*date*) – Fecha de proceso.
- **source** (Source) – Fuente de Información (Default = MUREX).

Devuelve
Dataset de tipo de cambio

get_index_rate(*index: FinancialIndex*, *process_date: date*) → float

Obtener valor de Índice Financiero

Obtiene el valor de un índice financiero a una fecha determinada

Parámetros

- **index** (FinancialIndex) – Índice Financiero.
- **process_date** (*date*) – fecha de proceso.

Devuelve
valor del índice.

Muestra
QueryException – En caso no se encuentre data o el índice no se reconozca.

get_single_index_dataset(*index: FinancialIndex*, *start_date: date*, *end_date: date*) → IndexDataSet

Obtener Valores Históricos de Índice Financiero

Obtiene el valor histórico de un índice financiero entre las fechas determinadas

Parámetros

- **index** (FinancialIndex) – Índice Financiero.
- **start_date** (*date*) – fecha de inicio.
- **end_date** (*date*) – fecha de término.

Devuelve
IndexDataSet con datos solicitados

Muestra
QueryException – En caso no se encuentre data o el índice no se reconozca.

get_index_dataset(*start_date: date*, *end_date: date*) → IndexDataSet

Obtener Valores Históricos de todos los Índices Financiero

Obtiene el valor histórico de todos los índices financieros entre las fechas determinadas

Parámetros

- **start_date** (*date.*) – Fecha de inicio.

- **end_date** (*date.*) – fecha de término.

Devuelve
IndexDataSet con datos solicitados

Muestra
QueryException – En caso no se encuentre data o el índice no se reconozca.

get_calendar_dataset() → CalendarDataSet

Obtener Dataset de Calendarios

Obtiene un dataset de Calendarios con los calendarios default

get_dataset(*process_date: date*, *source: Source = Source.OFFICIAL*) → DataSet

Obtener Dataset

Obtiene la información de mercado provista en los distintos archivos.

Parámetros

- **process_date** (*date.*) – Fecha del dataset.
- **source** (Source) – Fuente (Default: OFFICIAL)

Devuelve
Objeto Dataset.

get_operation(*id_number: int*, *process_date: date*) → Derivative

Obtener operación

Obtiene una operación desde la base de datos a partir del número de operación y la fecha de proceso

Parámetros

- **id_number** (*int*) – Número de operación.
- **process_date** (*date*) – Fecha de proceso.

Devuelve
Derivative

get_grf_fwd_operation(*id_number: int*, *process_date: date*) → Derivative

Obtener operación forward desde la interfaz GRF

Obtiene una operación desde la base de datos a partir del número de operación y la fecha de proceso

Parámetros

- **id_number** (*int*) – Número de operación.
- **process_date** (*date*) – Fecha de proceso.

Devuelve
Derivative

get_official_mtm(*id_number: int*, *process_date*) → Cash

Obtener Mark to Market Oficial

Obtiene el Mark to Market oficial de una operación

Parámetros

- **id_number** (*int*) – Número de operación.
- **process_date** (*date*) – Fecha de proceso.

Devuelve
Cash

`get_forward_portfolio`(*process_date: date*, *source: Source = Source.OFFICIAL*) → Portfolio

Obtener portafolio forward

Obtiene todos los forwards de la cartera

Parámetros

- **process_date** (*date*) – Fecha de Proceso.
- **source** (Source) – Fuente de Información (Default: Official)

Devuelve
Portfolio

`get_swap_portfolio`(*process_date: date*, *source: Source = Source.OFFICIAL*) → Portfolio

Obtener portafolio swap

Obtiene todos los swaps de la cartera

Parámetros

- **process_date** (*date*) – Fecha de Proceso.
- **source** (Source) – Fuente de Información (Default: Official)

Devuelve
Portfolio

`get_portfolio`(*process_date: date*, *source: Source = Source.OFFICIAL*) → Portfolio

Obtener el portafolio completo

Obtiene todos los productos de la cartera

Parámetros

- **process_date** (*date*) – Fecha de Proceso.
- **source** (Source) – Fuente de Información (Default: Official)

Devuelve
Portfolio

`get_grf_swap_operation`(*id_number: int*, *process_date: date*) → Derivative

Obtener operación Swap desde la interfaz GRF

Obtiene una operación desde la base de datos a partir del número de operación y la fecha de proceso

Parámetros

- **id_number** (*int*) – Número de operación.
- **process_date** (*date*) – Fecha de proceso.

Devuelve
Derivative

1.1.2.3 Paquete Matemático

El objetivo de este paquete es el de centralizar y estandarizar todos los calculos matemáticos realizados, lo que reduce la cantidad de código a utilizar y permite una estandarización de las formulas a ser utilizadas.

Los módulos que se encuentran en este paquete son:

Módulo de Interpolación

Contiene funciones para interpolación y extrapolación.

Módulo de Interpolación

Clases dedicadas a la interpolación y extrapolación de vectores

La principal clase de este módulo es:

- *Interpolación* (`Interpolator`): Clase dedicada a la interpolación de vectores.

Interpolación

class finmath.interpolator.**Interpolator**

Clase estática dedicada a la interpolación y extrapolación de vectores

Note

La clase se encuentra enfocada en la practicidad de uso, ya que puede ser llamada directamente sin necesidad de alguna inicialización, además el método *interpolate()* se encarga de apuntar a la interpolación requerida basándose en el método de interpolación o extrapolación especificado. Sin embargo, esto tiene un coste en eficiencia si se requieren hacer gran cantidad de interpolaciones, porque el proceso realizará siempre chequeos de seguridad (largo de vectores) y búsqueda de índices en el vértice x. Si es que se requiere hacer miles de interpolaciones sobre un mismo vector, lo recomendable es generar un nuevo vector con todos los puntos interpolados para que estos sean mantenidos en memoria y se reduzca el tiempo de ejecución. Esta decisión se deja al usuario.

static interpolate(*point*, *x_vector*, *y_vector*, *interpolation_method=InterpolationMethod.LINEAR*, *extrapolation_method=ExtrapolationMethod.SLOPE*) → float

Función de interpolación.

Parámetros

- **point** (*float*) – Punto a interpolar.
- **x_vector** (*list[]*) – Eje x.
- **y_vector** (*list[]*) – Eje y.
- **interpolation_method** (InterpolationMethod) – Método de interpolación (default: Lineal)
- **extrapolation_method** (ExtrapolationMethod) – Método de extrapolación (default: Pendiente)

Devuelve

Punto interpolado

Muestra
ValueException – Cuando la longitud de los vectores x e y no son iguales.

static interpolate_linear(*point*, *x_vector*, *y_vector*) → float

Interpolación Lineal

Genera una interpolación lineal para un punto especificado de acuerdo a la siguiente fórmula:

$$y = y_2 \cdot \alpha + y_1 \cdot (1 - \alpha)$$

Donde:

$$\alpha = \frac{(x - x_1)}{x_2 - x_1}$$

Parámetros

- **point** (*float*) – Punto a interpolar.
- **x_vector** (*List[]*) – Eje x.
- **y_vector** (*List[]*) – Eje y.

Devuelve
Punto interpolado

Muestra
ValueException – Cuando la longitud de los vectores x e y no son iguales.

static interpolate_log_linear(*point*, *x_vector*, *y_vector*) → float

Interpolación Log-lineal

Genera una interpolación Log-Lineal para un punto especificado de acuerdo a la siguiente fórmula:

$$y = y_2^{\alpha} \cdot y_1^{1-\alpha}$$

Donde:

$$\alpha = \frac{(x - x_1)}{x_2 - x_1}$$

Parámetros

- **point** (*float*) – Punto a interpolar.
- **x_vector** (*List[]*) – Eje x.
- **y_vector** (*List[]*) – Eje y.

Devuelve
Punto interpolado

Muestra
ValueException – Cuando la longitud de los vectores x e y no son iguales.

static extrapolate_slope(*point*, *x_vector*, *y_vector*) → float

Extrapolación por pendiente

Genera una extrapolación manteniendo la última pendiente para un punto especificado de acuerdo a la siguiente fórmula si el punto es mayor al último valor del eje x:

$$y = y_n + (x - x_n)\frac{y_n - y_{n-1}}{x_n - x_{n-1}}$$

En el caso en que el primer punto sea menor al primer punto del eje X se tiene la siguiente fórmula:

$$y = y_1 - (x_1 - x)\frac{y_2 - y_1}{x_2 - x_1}$$

Parámetros

- **point** (*float*) – Punto a extrapolar.
- **x_vector** (*List[]*) – Eje x.
- **y_vector** (*List[]*) – Eje y.

Devuelve
Punto extrapolado

Muestra
ValueException – Cuando la longitud de los vectores x e y no son iguales.

static extrapolate_log_slope(*point*, *x_vector*, *y_vector*) → float

Extrapolación por pendiente logarítmica

Genera una extrapolación manteniendo la última pendiente logarítmica para un punto especificado de acuerdo a la siguiente fórmula si el punto es mayor al último valor del eje x:

$$y = y_n \cdot e^{(x-x_n)\frac{ln(y_n)-ln(y_{n-1})}{x_n-x_{n-1}}}$$

En el caso en que el primer punto sea menor al primer punto del eje X se tiene la siguiente fórmula:

$$y = y_1 \cdot e^{-(x_1-x)\frac{ln(y_2)-ln(y_1)}{x_2-x_1}}$$

Parámetros

- **point** (*float*) – Punto a extrapolar.
- **x_vector** (*List[]*) – Eje x.
- **y_vector** (*List[]*) – Eje y.

Devuelve
Punto extrapolado

Muestra
ValueException – Cuando la longitud de los vectores x e y no son iguales.

static extrapolate_flat(*point*, *x_vector*, *y_vector*) → float

Extrapolación plana

Genera una extrapolación plana manteniendo el valor conocido más cercano.

Parámetros

- **point** (*float*) – Punto a extrapolar.
- **x_vector** (*List[]*) – Eje x.
- **y_vector** (*List[]*) – Eje y.

Devuelve
Punto extrapolado

Muestra
ValueException – Cuando la longitud de los vectores x e y no son iguales.

static extrapolate_flat_rate(*point*, *x_vector*, *y_vector*) → float

Extrapolación plana

Genera una extrapolación plana manteniendo el valor conocido más cercano tratándolo como una tasa YIELD ACT/360.

Parámetros

- **point** (*float*) – Punto a extrapolar.
- **x_vector** (*List[]*) – Eje x.
- **y_vector** (*List[]*) – Eje y.

Devuelve
Punto extrapolado

Muestra
ValueException – Cuando la longitud de los vectores x e y no son iguales.

static extrapolate_rate_slope(*point*, *x_vector*, *y_vector*) → float

Extrapolación por tasa YIELD Act/360

Genera una extrapolación manteniendo la última pendiente para un punto especificado bajo una tasa YIELD Act/360 de acuerdo a la siguiente fórmula si el punto es mayor al último valor del eje x:

$$r = r_n + (x - x_n)\frac{r_n - r_{n-1}}{x_n - x_{n-1}}$$
$$y = (1 + r)^{(-x/360)}$$

En el caso en que el primer punto sea menor al primer punto del eje X se tiene la siguiente fórmula:

$$r = r_1 - (x_1 - x)\frac{r_2 - r_1}{x_2 - x_1}$$
$$y = (1 + r)^{(-x/360)}$$

Parámetros

- **point** (*float*) – Punto a extrapolar.
- **x_vector** (*List[]*) – Eje x.
- **y_vector** (*List[]*) – Eje y.

Devuelve
Punto extrapolado

Muestra
ValueException – Cuando la longitud de los vectores x e y no son iguales.

1.1.2.4 Paquete de Mercado

El paquete de mercado se encuentra dedicado a almacenar clases que representen información de mercado. Dentro de este paquete contamos con los siguientes módulos:

Módulo Calendarios

Módulo dedicado a la generación, almacenamiento y manejo de días hábiles y feriados.

Módulo Curvas

Módulo dedicado a la generación, almacenamiento y construcción de curvas de valorización y proyección.

Módulos Tipo de cambio

Módulo dedicado al manejo de Pares de Monedas.

Módulos Índices Financieros

Módulo dedicado al manejo de indices de mercado.

Módulo Datasets

Módulo dedicado al almacenamiento, manejo y utilización conjunta de los datos de mercado.

Módulo Calendarios

El módulo de calendarios se encarga de generar clases que contengan la información de los días laborales de una localidad específica, al igual que permitir generar calculos relacionados con fechas.

La principal clase de este módulo es:

- *Calendario* (`TradingCalendar`): Clase dedicada al manejo de días hábiles.

Calendario

class market.tradingcalendar.**TradingCalendar**(*calendar_name: str*, *holidays: list*, *base_calendar: AbstractHolidayCalendar*)

Calendarios

Clase dedicada al almacenamiento de calendarios

Parámetros

- **calendar_name** (*str*) – Nombre del Calendario
- **holidays** (*list*) – listado de feriados.
- **base_calendar** (*AbstractHolidayCalendar*) – Calendario base.

adjust_workday(*workday: date*, *business_day_convention: BusinessDay*)

Ajustar día laborable

Ajusta el día laborable provisto de acuerdo a la convención solicitada.

Parámetros

- **workday** (*date*) – Día laborable inicial.
- **business_day_convention** (BusinessDay) – Convención de ajuste.

Devuelve

Fecha ajustada

offset_date(*start_date: date*, *n_periods: int*, *period: Period*, *convention: BusinessDay*) → date

Mover fecha

Mueve una fecha dentro de un periodo específico.

Parámetros

- **start_date** (*date*) – fecha de inicio.
- **n_periods** (*int*) – número de periodos.
- **period** (Period) – Periodo
- **convention** (BusinessDay) – Convención.

Devuelve
Fecha con offset

static is_weekend(*selected_date*)

Fin de Semana

Verifica si la fecha propuesta es un fin de semana (sábado y domingo).

Parámetros
selected_date (*date*) – Fecha.

Devuelve
Booleano

static get_year_fraction(*start_date: date*, *end_date: date*, *day_count: DayCount*)

Obtener Fracción de Año

Obtiene la fracción de año entre las fechas provistas utilizando la convención de conteo de días provista

Parámetros

- **start_date** (*date*) – Fecha de inicio.
- **end_date** (*date*) – Fecha de fin.
- **day_count** (DayCount) – Convención de conteo de días.

Devuelve
fracción de año

Módulo Curvas

Este módulo contiene los objetos necesarios para poder representar curvas de descuento y proyección.

La principal clase de este módulo es:

- *Curva* (Curve): Clase dedicada a representación de curvas de proyección y descuento.

Curva

class market.curve.**Curve**(*curve_name: str*, *process_date: date*, *tenors: list*, *discount_factors: list*, *curve_config: dict = None*)

Curva

Clase dedicada a la representación de curvas de descuento y de proyección

Parámetros

- **curve_name** (*str*) – Identificador de la curva.
- **process_date** (*date*) – Fecha de proceso de la curva actual.
- **tenors** (*list[int]*) – Lista de tenors asignados.
- **discount_factors** (*list[float]*) – Lista de factores de descuento mid.
- **curve_config** (*dict*) – Diccionario con configuración de curva (Default: None)

get_discount_factor(*t: int*) → float

Obtener factor de descuento

Obtiene el factor de descuento al tenor especificado

Parámetros
t (*int*) – tenor especificado.

Devuelve
Factor de descuento

get_cap_factor(*t: int*) → float

Obtener factor de capitalización

Obtiene el factor de capitalización a un tenor especificado

Parámetros
t (*int*) – tenor especificado

Devuelve
Factor de capitalización

get_forward_factor(*t_start*, *t_end*)

Obtener Factor Forward

Obtiene el factor forward entre los tenors especificados

Parámetros

- **t_start** – tenor de inicio
- **t_end** – tenor de final

Devuelve
Factor Forward

get_rate(*t: int*) → float

Obtener tasa de descuento

Obtiene la tasa de descuento del tenor especificado

Parámetros
t (*int*) – tenor especificado.

Devuelve
Tasa de descuento

`get_rate_curve()`

Obtener curva de tasas

Obtiene la curva expresada en tasa

Devuelve
Tuple con tenors y tasa.

`to_dict()`

Convertir en diccionario

Convierte los datos de la curva en un diccionario

Devuelve
Diccionario con datos de la curva

`rates_to_dict()`

Convertir a un diccionario de tasas

Convierte la curva en un diccionario de tasas

Devuelve
Diccionario de tasas

`to_dataframe()`

Convertir a dataframe

Convierte los datos de la curva a un dataframe

Devuelve
Dataframe con datos de la curva

`rates_to_dataframe()`

Convertir tasas a dataframe

Convierte las tasas de la curva a un dataframe

Devuelve
Dataframe con tasas de la curva

`static df_to_rate(`*discount_factor*, *start_date*, *end_date*, *rate_compounding*, *rate_day_count*`)`

Convertir Factor de Descuento a Tasa

Convierte el factor de descuento provisto a una tasa utilizando las convenciones de composición y conteo de días provistas.

Parámetros

- **discount_factor** (*float.*) – Factor de descuento.
- **start_date** (*date.*) – fecha de inicio.
- **end_date** (*date.*) – fecha de fin.
- **rate_compounding** (*Compounding.*) – Convención de composición de Tasa.
- **rate_day_count** (*DayCount.*) – Convención de conteo de días.

Devuelve
Tasa de interés.

static rate_to_df(*annual_rate: float*, *start_date: date*, *end_date: date*, *rate_compounding: Compounding*, *rate_day_count: DayCount*)

Convertir Tasa a Factor de Descuento

Convierte la tasa a factor de descuento utilizando las convenciones de composición y conteo de días provisto.

Parámetros

- **annual_rate** – Tasa Anual.
- **start_date** (*date.*) – fecha de inicio.
- **end_date** (*date.*) – fecha de fin.
- **rate_compounding** (*Compounding.*) – Convención de composición de Tasa.
- **rate_day_count** (*DayCount.*) – Convención de conteo de días.

Devuelve
Factor de descuento

static rate_to_cap_factor(*annual_rate*, *start_date*, *end_date*, *rate_compounding*, *rate_day_count*)

Convertir Tasa a Factor de Capitalización

Convierte la tasa a factor de capitalización utilizando las convenciones de composición y conteo de días provisto.

Parámetros

- **annual_rate** – Tasa Anual.
- **start_date** (*date*) – fecha de inicio.
- **end_date** (*date*) – fecha de fin.
- **rate_compounding** (Compounding) – Convención de composición de Tasa.
- **rate_day_count** (DayCount) – Convención de conteo de días.

Devuelve
Factor de capitalización

Módulos Tipo de cambio

El módulo de tipos de cambio permite la estandarización en la representación de los distintos pares de moneda.

La principal clase de este módulo es:

- *Tasa de Cambio* (*FxPair*): Clase dedicada a representación de tipos de cambio.

Tasa de Cambio

class market.fxrate.**FxPair**(*primary_currency: Currency*, *secondary_currency: Currency*, *discount_days: int = 2*, *calendar_locations=None*)

Representación de un par de mondas

Cuenta con una moneda primaria y otra moneda secundaria, siendo el tipo de cambio la cantidad de unidades de la moneda secundaria que se requieren para adquirir una unidad de la moneda primaria.

Parámetros

- **primary_currency** (*Currency.*) – Moneda primaria.
- **secondary_currency** (*Currency.*) – Moneda secundaria.
- **discount_days** (*Int.*) – Cantidad de días de descuento (default: 2).
- **calendar_locations** (*List*) – Lista de localizaciones.

inverse_pair()

Obtener par inverso

Invierte el par de monedas y retorna un nuevo par de monedas

Devuelve
FxPair con monedas invertidas

has_eq_currency()

Monedas Iguales

Verifica si las monedas primarias y secundarias sean iguales

Devuelve
Booleano

Módulos Índices Financieros

El módulo de índices se encarga de proveer la información y configuración de los principales índices de mercado.

La principal clase de este módulo es:

- *Índices Financieros* (*Index*): Clase dedicada a representación de tipos de cambio.

Índices Financieros

class market.index.**Index**(*name: str*, *idx_type: IndexType*, *periodicity: Periodicity*, *day_count: DayCount*, *compounding: Compounding*)

Clase base Índice

Representa un índice financiero

Parámetros

- **name** (*str*) – Nombre.
- **idx_type** (IndexType) – Tipo de índice.
- **periodicity** (Periodicity) – Periodicidad.
- **day_count** (*day_count*) – Convención de conteo de días.
- **compounding** (Compounding) – Convención de composición.

Módulo Datasets

Este módulo está enfocado en la creación de datasets, es decir objetos que contengan y ayuden a manejar la información dentro de los mismos.

Las principales clases de este módulo son:

- *Dataset de Mercado* (`DataSet`): Clase dedicada de datos de mercado.
- *Dataset de Curvas* (`CurveDataSet`): Clase dedicada al manejo de datos de curvas.
- *Dataset de Pares de Moneda* (`FxDataSet`): Clase dedicada al manejo de datos de pares de moneda.
- *Dataset de Índices Financieros* (`IndexDataSet`): Clase dedicada al manejo de datos de índices financieros.

Dataset de Mercado

class market.dataset.**DataSet**(*process_date: date*, *curve_dataset:* CurveDataSet, *fx_dataset:* FxDataSet, *index_dataset:* IndexDataSet, *calendar_dataset: CalendarDataSet = None*)

Clase dedicada al almacenamiento de información de mercado.

Parámetros

- **process_date** (*date*) – Fecha de la información de mercado.
- **curve_dataset** (CurveDataSet) – (Opcional) Set de datos de Curvas.
- **fx_dataset** (FxDataSet) – (Opcional) Set de datos de Tasas de Cambio.
- **index_dataset** (IndexDataSet) – (Opcional) Set de datos de Índices Financieros.

set_default_valuation_currency(*valuation_currency*)

Asigna la moneda default de valorización

Asigna una moneda default para generar los calculos de valorización

Parámetros
valuation_currency (Currency) – Moneda de valorización.

get_default_valuation_currency()

Obtener la moneda default de valorización

Obtiene la moneda asignada como moneda default para generar los calculos de valorización

set_curve_dataset(*curve_dataset:* CurveDataSet)

Asignar dataset de Curvas

Asigna un dataset de curvas

Parámetros
curve_dataset (CurveDataSet) – dataset de curvas

set_fx_dataset(*fx_dataset:* FxDataSet)

Asignar dataset de Tipos de Cambio

Asigna un dataset de Tipos de Cambio

Parámetros
fx_dataset (CurveDataSet) – dataset de tipos de cambio

set_index_dataset(*index_dataset:* IndexDataSet)

Asignar dataset de Índices

Asigna un dataset de Índices

Parámetros
index_dataset (CurveDataSet) – dataset de Índices

disable_discounted_spot()

Desactiva el uso de spot descontado.

enable_discounted_spot()

Activa el uso de spot descontado.

get_fx_rate(*fx_pair: FxPair*)

Obtener tipo de cambio

Obtiene tipo de cambio a la fecha del dataset.

Parámetros
fx_pair (FxPair) – Nombre de la tasa de cambio.

Muestra
DatasetException – En caso no encuentre datos.

Devuelve
Tipo de Cambio

get_fx_forward_rate(*fx_pair: FxPair*, *forward_date: date*)

Obtener tipo de cambio forward

Obtiene el tipo de cambio a una fecha futura, utilizando la siguiente fórmula:

$$S_t^{C_1-C_2} = S_0^{C_1-C_2} \cdot \frac{P_0^{C_1}}{P_0^{C_2}}$$

Parámetros

- **fx_pair** (FxPair) – Tasa de Cambio.
- **forward_date** (*date*) – Fecha Futura.

Muestra
DatasetException – En caso no se pueda realizar el cálculo solicitado.

Devuelve
Tipo de cambio a fecha futura

get_discount_curve(*associated_currency: Currency*, *associated_collateral: Currency = None*) → Curve

Obtener curva de descuento

Obtiene una curva de descuento de acuerdo la moneda y colateral asociada. En caso la curva no exista se intentará construir una curva sintética tomando como base la moneda default de las curvas. Si no es posible obtener un sintético se levanta una excepción.

Parámetros

- **associated_currency** (*Currency.*) – Moneda asociada.
- **associated_collateral** (Currency) – Colateral asociado.

Muestra
DatasetException – En caso la curva no pueda ser obtenida

Devuelve
Objeto Curve

`get_projection_curve`(*associated_index: Index*)

Obtener curva de proyección

Obtiene una curva de proyección para el índice señalado

Parámetros
associated_index (Index) – Índice asociado.

Devuelve
Curva de proyección

`get_index_data`(*index: FinancialIndex*, *idx_date: date*)

Obtener data de índices

Obtiene la data de un índice a una fecha especificada.

Parámetros

- **index** (Index) – Indice.
- **idx_date** (*date*) – Fecha requerida.

Muestra
DatasetException – En caso no se cuente con la información requerida.

Devuelve
Información del índice a la fecha especificada.

`get_index_data_between`(*index: FinancialIndex*, *start_date: date*, *end_date: date*)

Obtener data de índices entre las fechas

Parámetros

- **index** (FinancialIndex) – Índice.
- **start_date** (*date*) – Fecha de inicio.
- **end_date** (*date*) – Fecha de fin.

Devuelve
Información entre las fechas designadas

`get_calendar`(*localities: list*)

Obtiene un calendario dada una lista de localidades

Parámetros
localities (*list*) – Listado de localidades

Devuelve
TradingCalendar

Dataset de Curvas

class market.dataset.**CurveDataSet**(*process_date: date*, *default_collateral_currency: Currency = Currency.USD*)

Set de Datos de Curvas

Clase especializada en el almacenamiento y tratamiento de datos de curvas.

Parámetros

- **process_date** (*date*) – Fecha de Proceso.
- **default_collateral_currency** (Currency) – Moneda de colateral (default: USD).

set_default_collateral_currency(*collateral_currency: Currency*)

Asignar moneda default de colateral

Asigna una moneda como colateral default para todos los calculos internos, incluyendo la generación de curvas sintéticas.

Parámetros
collateral_currency (Currency) – Moneda de Colateral

get_default_collateral_currency()

Obtener moneda default de colateral

Obtiene la moneda asignada como colateral default para todos los calculos internos.

Devuelve
Moneda de colateral default

add_discount_curve(*curve*, *associated_currency: Currency*, *associated_collateral: Currency = None*)

Agregar curva de descuento

Agrega una curva de descuento considerando una moneda y un colateral asociado

Parámetros

- **curve** (Curve) – Curva.
- **associated_currency** (Currency) – Moneda asociada a la Curva.
- **associated_collateral** (Currency) – Colateral asociado a la Curva.

get_discount_curve(*associated_currency: Currency*, *associated_collateral: Currency = None*)

Obtener curva de descuento

Obtiene una curva de descuento de acuerdo la moneda y colateral asociada. En caso la curva no exista se intentará construir una curva sintética tomando como base la moneda default de las curvas. Si no es posible obtener un sintético se levanta una excepción.

Parámetros

- **associated_currency** (*Currency.*) – Moneda asociada.
- **associated_collateral** (Currency) – Colateral asociado.

Muestra
DatasetException – En caso la curva no pueda ser obtenida

Devuelve
Objeto Curve

add_projection_curve(*curve: Curve*, *associated_index: FinancialIndex*)

Agregar curva de proyección

Agrega una curva de proyección

Parámetros

- **curve** (Curve) – Curva
- **associated_index** – Indice asociado
- **associated_index** – Index

get_projection_curve(*associated_index: Index*) → Curve

Obtener curva de proyección

Obtiene una curva de proyección para el índice señalado

Parámetros
associated_index (Index) – Índice asociado.

Devuelve
Curva de proyección

Dataset de Pares de Moneda

class market.dataset.**FxDataSet**(*process_date: date*)

Set de Datos de Pares de Moneda

Clase especializada en el almacenamiento y tratamiento de pares de monedas.

Parámetros
process_date (*date*) – Fecha de Proceso.

add_fx_rate(*fx_pair: FxPair*, *fx_mid: float*, *discounted_fx: float = None*)

Agregar tipo de cambio

Agrega un objeto tipo de cambio al dataset.

Parámetros

- **fx_pair** (FxPair) – Par de monedas.
- **fx_mid** (*float*) – Valor mid.
- **discounted_fx** (*float*) – Valor descontado (Opcional)

contains(*fx_pair: FxPair*)

Verificar contención de fx pair

Verifica si el dataset contiene un par de moneda especifico

Parámetros
fx_pair (FxPair) – Par de moneda

Devuelve
bool

contains_discounted(*fx_pair: FxPair*)

Verificar si contiene Spot Descontado

Verifica si el dataset contienen la moneda y el spot descontado respectivo

Parámetros
fx_pair (FxPair) – Par de moneda

Devuelve
bool

set_fx_rate(*fx_pair: FxPair*, *fx_rate: float*)

Asigna nuevo valor a tasa de cambio

Asigna nuevo valor a tasa de cambio

Parámetros

- **fx_pair** (FxPair) – Par de monedas.
- **fx_rate** (*float*) – Valor mid.

get_fx_rate(*fx_pair: FxPair*)

Obtener valor de tasa de cambio

Obtiene el valor almacenado de tasa de cambio

Parámetros
fx_pair (FxPair) – Par de monedas.

set_discounted_fx_rate(*fx_pair: FxPair*, *discounted_fx_rate: float*)

Asigna nuevo valor a tasa de cambio

Asigna nuevo valor a tasa de cambio

Parámetros

- **fx_pair** (FxPair) – Par de monedas.
- **discounted_fx_rate** (*float*) – Valor descontado.

get_discounted_fx_rate(*fx_pair: FxPair*)

Obtener valor descontado de tasa de cambio

Obtiene el valor almacenado de tasa de cambio descontada

Parámetros
fx_pair (FxPair) – Par de monedas.

set_projection(*fx_pair: FxPair*, *projection_date: date*, *projected_rate: float*)

Asigna nuevo valor a tasa de cambio

Asigna nuevo valor a tasa de cambio

Parámetros

- **fx_pair** (FxPair) – Par de monedas.
- **projection_date** (*date*) – Fecha de proyección
- **projected_rate** (*float*) – Tasa Proyectada.

get_projection(*fx_pair: FxPair*, *projection_date: date*)

Asigna nuevo valor a tasa de cambio

Asigna nuevo valor a tasa de cambio

Parámetros

- **fx_pair** (FxPair) – Par de monedas.
- **projection_date** (*date*) – Fecha de proyección

Dataset de Índices Financieros

class market.dataset.**IndexDataSet**

Set de Datos de Índices Financieros

Clase especializada en el almacenamiento y tratamiento de índices financieros.

set_index_data(*index*, *historical_data*)

Asignar data de índices

Asigna la información histórica correspondiente a un índice.

Parámetros

- **index** (Index) – Indice.
- **historical_data** (*dict*) – Data histórica.

get_index_data(*index: FinancialIndex*, *idx_date: date*)

Obtener data de índices

Obtiene la data de un índice a una fecha especificada.

Parámetros

- **index** (Index) – Indice.
- **idx_date** (*date*) – Fecha requerida.

Muestra
DatasetException – En caso no se cuente con la información requerida.

Devuelve
Información del índice a la fecha especificada.

get_index_data_between(*index: FinancialIndex*, *start_date: date*, *end_date: date*) → dict

Obtener data de índices entre las fechas

Parámetros

- **index** (FinancialIndex) – Indice.
- **start_date** (*date*) – Fecha de inicio.
- **end_date** (*date*) – Fecha de fin.

Devuelve
Información entre las fechas designadas

1.1.2.5 Paquete de Instrumentos

El paquete de instrumentos esta dedicado a la generación de objetos estandarizados que posibiliten el trabajo con los distintos instrumentos de mercado. Esto permite la valorización y obtención de riesgos de manera rápida y estandarizada.

Dentro de este paquete contamos con los siguientes módulos:

Módulo Flujos

Módulo dedicado a la representación de Flujos de Caja.

Módulo Instrumento

Módulo con clases abstractas para la representación de instrumentos financieros

Módulo Forwards

Módulo con clases enfocadas en la representación de forwards.

Módulo Swaps

Módulo con clases enfocadas en la representación de swaps.

Módulo Portafolio

Módulo con clases enfocadas en la representación de portafolios.

Módulo Flujos

Las clases en este módulo están dedicadas a la representación de cajas y flujos de caja.

Las clases dentro de este módulo son:

- *Cash* (`Cash`): Clase dedicada a la representación de caja financiera.
- *Cash Flow* (`CashFlow`): Clase dedicada a la representación de flujos de caja.

Cash

class instruments.flow.**Cash**(*amount: float*, *currency: Currency*)

Clase Cash

Representa una caja con moneda y monto específico. La clase soporta la suma de otro instrumento cash y su comparación.

Parámetros

- **amount** (*float*) – Monto
- **currency** (Currency) – Numerario o Moneda

Muestra

- **TypeError** – Cuando se intenta sumar con un objeto no Cash
- ***CurrencyException*** – Cuando se intenta sumar con un objeto Cash con moneda distinta

Ejemplo de uso

```
from finlib.instruments.flow import Cash
from finlib.enums import Currency

cash_1 = Cash(amount = 100, currency = Currency.USD)
cash_2 = Cash(amount = 150, currency = Currency.CLP)
cash_3 = Cash(amount = 100, currency = Currency.USD)

cash_fail = cash_1 + "100 USD" # Genera una excepción TypeError
cash_fail = cash_1 + cash_2 # Genera una excepción CurrencyException
cash_4 = cash_1 + cash_3 # Cash(200, Currency.USD)

result = cash_1 == "100 USD" # false
result = cash_1 == cash_2 # false
result = cash_1 == cash3 # true
```

Cash Flow

class instruments.flow.**Cashflow**(*flow_date: date*, *amount: float*, *currency: Currency*)

Clase Cashflow

Representa un flujo futuro con moneda, monto y fecha determinada. La clase soporta la suma de otro instrumento cashflow, al igual que su comparación.

Parámetros

- **flow_date** (*date*) – Fecha del flujo
- **amount** (*float*) – Monto
- **currency** (Currency) – Numerario o Moneda

Muestra

- **TypeError** – Cuando se intenta sumar con un objeto no Cashflow
- ***CurrencyException*** – Cuando se intenta sumar con un objeto Cashflow con moneda distinta
- **ValueError** – Cuando se intenta sumar con un objeto Cashflow con fecha distinta

```
from finlib.instruments.flow import Cashflow
from finlib.enums import Currency
from datetime import date

cashflow_1 = Cashflow(flow_date=date(year=2022, month=1, day=30), amount = 100,␣
↪currency = Currency.USD)
cashflow_2 = Cashflow(flow_date=date(year=2022, month=1, day=30), amount = 150,␣
↪currency = Currency.CLP)
cashflow_3 = Cashflow(flow_date=date(year=2022, month=1, day=30), amount = 100,␣
↪currency = Currency.USD)
cashflow_4 = Cashflow(flow_date=date(year=2022, month=3, day=30), amount = 100,␣
↪currency = Currency.USD)

cashflow_fail = cashflow_1 + "100 USD" # Genera una excepción TypeError
cashflow_fail = cashflow_1 + cashflow_2 # Genera una excepción CurrencyException
cashflow_fail = cashflow_1 + cashflow_4 # Genera una excepción ValueError
```

(continué en la próxima página)

(proviene de la página anterior)

```
cash_5 = cash_1 + cash_3 # Cashflow(flow_date=date(year=2022, month=1, day=30), 200,
↪ Currency.USD)

result = cash_1 == "100 USD" # false
result = cash_1 == cash_2 # false
result = cash_1 == cash3 # true
```

Módulo Instrumento

Este módulo cuenta con una clase abstract base para los instrumentos derivados

- *Derivado Financiero* (`Derivative`): Clase abstracta dedicada a la representación de un derivado financiero.

Derivado Financiero

class instruments.instrument.**Derivative**

add_metadata(*parameter*, *value*)

Agregar Metadata

Agrega un parámetro y valor a la metadata del instrumento.

Parámetros

- **parameter** (*str*) – Nombre del parámetro.
- **value** (*Any*) – Valor del parámetro.

Devuelve

get_metadata(*parameter*)

Obtener Metadata

Obtiene un parámetro de la metadata del instrumento.

Parámetros
parameter (*str*) – Nombre del parámetro.

Devuelve
valor de la metadata

add_active_cashflow(*active_cashflow: Cashflow*)

Agregar flujo de caja activo

Agrega un flujo de caja activo.

Parámetros
active_cashflow (Cashflow) – Flujo de caja

add_passive_cashflow(*passive_cashflow: Cashflow*)

Agregar flujo de caja pasivo

Agrega un flujo de caja pasivo.

Parámetros
passive_cashflow (Cashflow) – Flujo de caja

get_active_cashflows()

Obtener flujos de caja activos

Obtiene los flujos de caja activos.

get_passive_cashflows()

Obtener flujos de caja pasivos

Obtiene los flujos de caja pasivos.

get_cashflows()

Obtener flujos de caja

Obtiene los flujos de caja.

calculate_cashflows()

Calcular flujos de caja (Abstracto)

Método abstracto para el cálculo de flujos de caja

get_valuated_cashflows(*dataset: DataSet*)

Valorizar flujos de caja (Abstracto)

Método abstracto para la valorización de flujos de caja

valuate(*dataset: DataSet*)

Valorizar (Abstracto)

Método abstracto para la valorización del instrumento

Módulo Forwards

Este módulo se encuentra enfocado en representar productos forward, para esto se cuenta con las siguientes clases.

La clase Forward que representa un forward Vanilla.

Las clases dentro de este módulo son:

- *Forward* (*Forward*): Clase dedicada a la representación de Forwards de Moneda.

Forward

class instruments.forward.**Forward**(*start_date: date*, *end_date: date*, *fixing_date: date*, *pay_date: date*, *active_notional: Cash*, *passive_notional: Cash*, *associated_collateral: Currency*, *payment_currency: Currency = Currency.CLP*)

Instrumento Forward

Clase dedicada a la representación de forwards vanilla.

Parámetros

- **start_date** (*date*) – Fecha de inicio.
- **end_date** (*date*) – Fecha de termino.
- **fixing_date** (*date*) – Fecha de fijación.
- **pay_date** (*date*) – Fecha de pago.
- **active_notional** (Cash) – Nocional activo.

- **passive_notional** (Cash) – Nocional pasivo.
- **associated_collateral** (Currency) – Colateral asociado.

valuate(*dataset: DataSet*)

Valorizar

Obtiene el mtm, en caso no se especifique la fecha se utilizara la fecha del data set

Parámetros
dataset (DataSet) – Dataset de mercado

Devuelve
Objeto Cash con valor presente del instrumento

get_strike()

Obtener strike

Obtiene el strike del forward

Devuelve
Strike

get_currency_pair()

Obtener par de moneda

Obtiene el par de moneda asociada con el instrumento

Devuelve
FxPair

Módulo Swaps

Este módulo se encuentra enfocado en representar productos swap.

Las clases dentro de este módulo son:

- *Cupón* (*Coupon*): Clase dedicada a la representación de Cupones.
- *Swap Leg* (*Leg*): Clase dedicada a la representación de Swap Legs.
- *Swap* (*Swap*): Clase dedicada a la representación de Swap Legs.

Cupón

class instruments.swap.**Coupon**(*associated_index: Index*, *reference_rate: float*, *spread: float*, *start_date: date*, *end_date: date*, *start_fixing_date: date*, *end_fixing_date: date*, *interest_payment_date: date*, *principal_payment_date: date*, *outstanding_notional: Cash*, *amortization: Cash*, *associated_collateral: Currency*)

Cupón

Clase enfocada en la representación de cupones con interés y/o amortizaciones.

Parámetros

- **associated_index** (Index) – Índice Asociado.
- **reference_rate** (*float*) – Tasa de referencia.
- **spread** (*float*) – Spread sobre la tasa de interés.

- **start_date** (*date*) – Fecha de inicio del cupón.
- **end_date** (*date*) – Fecha de término del cupón.
- **start_fixing_date** (*date*) – Fecha de inicio de fijación del cupón.
- **end_fixing_date** (*date*) – Fecha de término de fijación del cupón.
- **interest_payment_date** (*date*) – Fecha de pago del interés.
- **principal_payment_date** (*date*) – Fecha de pago de la amortización.
- **outstanding_notional** (Cash) – Nocional remanente.
- **amortization** (Cash) – Amortización.
- **associated_collateral** (Currency) – Colateral asociado.

to_dict()

Convertir a diccionario

Convierte el cupón a un diccionario

Devuelve
Diccionario con valores del cupón

get_valuation_details(*dataset: DataSet*) → dict

Detalle de valorización

Valoriza y detalla el instrumento utilizando la información de mercado provista

Parámetros
dataset (DataSet) – Información de mercado.

Devuelve
Diccionario de datos

valuate(*dataset: DataSet*)

Valorizar

Valoriza el instrumento utilizando la información de mercado provista

Parámetros
dataset (DataSet) – Información de mercado.

Devuelve
Objeto Cash con valor presente del instrumento

to_dataframe()

Convertir a dataframe

Convierte a Dataframe

Devuelve
DataFrame con valores del cupón

Swap Leg

class instruments.swap.**Leg**(*position: FinancialPosition*, *associated_collateral*)

Swap Leg

Clase enfocada en la representación de Swap Legs.

Parámetros

- **position** (FinancialPosition) – Posición financiera.
- **associated_collateral** (Currency) – Colateral asociado.

add_coupon(*coupon:* Coupon)

Agregar Cupón

Agrega una cupón al swap leg.

Parámetros
coupon (Coupon) – Cupón

get_coupon(*i: int*)

Obtener cupón

Obtiene el cupón en la posición i

Parámetros
i (*int*) – Posición

Devuelve
Coupon

to_dataframe()

Convertir a dataframe

Convierte a Dataframe

Devuelve
DataFrame con valores del cupón

valuate(*dataset: DataSet*)

Valorizar

Valoriza el instrumento utilizando la información de mercado provista

Parámetros
dataset (DataSet) – Información de mercado.

Devuelve
Objeto Cash con valor presente del instrumento

Swap

class instruments.swap.**Swap**(*active_leg*, *passive_leg*, *associated_collateral*)

Instrumento Swap

Clase base dedicada a la representación de instrumentos Swap.

Parámetros
associated_collateral (Currency) – Colateral asociado.

add_leg(*leg:* Leg)

Agregar Leg

Agrega un objeto Leg al Swap

Parámetros

- **leg** – Swap Leg
- **leg** – Leg

to_dataframe()

Convertir a dataframe

Convierte a Dataframe

Devuelve
DataFrame con valores del cupón

get_flow(*dataset: DataSet*)

Obtener Flujo

Obtiene los flujos proyectados a la fecha de pago

Parámetros
dataset (DataSet) – Set de datos de mercado.

Devuelve
Lista de Cashflow a la fecha de pago.

valuate(*dataset: DataSet*)

Valorizar

Valoriza el instrumento utilizando la información de mercado provista

Parámetros
dataset (DataSet) – Información de mercado.

Devuelve
Objeto Cash con valor presente del instrumento

valuate_by_leg(*dataset: DataSet*)

Valorizar por leg

Valoriza las legs del instrumento utilizando la información de mercado provista

Parámetros
dataset (DataSet) – Información de mercado.

Devuelve
lista con objetos Cash con valor presente de las legs del instrumento

Módulo Portafolio

Este módulo se encuentra enfocado en representar portafolios de productos.

Las clases dentro de este módulo son:

- *Portafolio* (`Portfolio`): Clase dedicada a la representación de un portafolio genérico.

Portafolio

class instruments.portfolio.**Portfolio**(*portfolio_name: str*)

Portafolio

Clase enfocada en la representación de un portafolio de productos genérico.

Parámetros
portfolio_name (*str*) – Nombre del portafolio

add_operation(*id_number: float*, *instrument: Derivative*)

Agregar Operación

Agrega una operación al portafolio

Parámetros

- **id_number** (*float*) – Numero de identificación
- **instrument** (Derivative) – Instrumento

valuate(*dataset: DataSet*) → dict

Valorizar

Valoriza el portafolio utilizando el dataset provisto

Parámetros
dataset (DataSet) – Dataset

valuate_to_excel(*dataset: DataSet*, *file_name: str*)

Valorizar a Excel

Valoriza el portafolio y muestra los resultados en formato excel

Parámetros

- **dataset** (DataSet) – Dataset.
- **file_name** (*str*) – Nombre del archivo (debe incluir terminación .xlsx)

Devuelve

merge(*other_portfolio*)

Combinar Portafolios

Combina dos portafolios en uno

Parámetros
other_portfolio (Portfolio) – Otro Portafolio

1.2 Ejemplos de Uso

1.2.1 Uso de Conexión a Base de datos

La conexión a base de datos está pensada en permitir la extracción rápida de datos desde cualquier base de datos utilizando queries definidas por el usuario.

1.2.1.1 Carga de Librerías

[1]:
```
from finrisklib.data.dbconnection import SQLConn
from IPython.display import display # Formateo
from datetime import date
```

1.2.1.2 ¿Como ejecutar una query?

1. Definir la consulta como un string

[2]:
```
query_simple = """SELECT * FROM Banco.dbo.tipocambio WHERE fecha = '20220823'""" # Se␣
↪puede definir la query de forma directa

process_date = date(year=2022, month=8, day=22)
query_template = f"SELECT * FROM Banco.dbo.tipocambio WHERE fecha = '{process_date.
↪strftime('%Y%m%d')}'" # También se puede definir de forma parametrizada
```

2. Inicializar lector

[3]:
```
reader = SQLConn.get_b08_conn() # Inicializando lector
```

3. Ejecutar query

[4]:
```
result_simple = reader.execute_query(query=query_simple)
result_template = reader.execute_query(query=query_template)
```

4. Mostrar resultados

[5]:
```
print("Resultado Query Simple: ")
display(result_simple)
```

```
Resultado Query Simple:
```

	fecha	localidad	moneda	valor
0	2022-08-23	BR	BRL	180.960000
1	2022-08-23	CL	AUD	636.726400
2	2022-08-23	CL	BRL	180.454000
3	2022-08-23	CL	CAD	708.457200

(continué en la próxima página)

(proviene de la página anterior)

```
4  2022-08-23          CL    CHF    952.468700
5  2022-08-23          CL    CLD    945.470000
6  2022-08-23          CL    CLF  33715.330000
7  2022-08-23          CL    CNY    134.202600
8  2022-08-23          CL    COP      0.210000
9  2022-08-23  CL            DEG   1277.010000
10 2022-08-23          CL    DKK    123.068800
11 2022-08-23          CL    EUR    915.394600
12 2022-08-23          CL    GBP   1086.970900
13 2022-08-23          CL    JPY      6.723400
14 2022-08-23          CL    KRW      0.681700
15 2022-08-23          CL    MXN     45.879100
16 2022-08-23          CL    NOK     94.526400
17 2022-08-23  CL            NZD    571.140000
18 2022-08-23          CL    PEN    237.780800
19 2022-08-23          CL    SEK     86.521900
20 2022-08-23          CL    USD    917.275000
21 2022-08-23  CL            ZAR     54.060000
22 2022-08-23          US    AUD      0.694200
23 2022-08-23          US    BRL      5.083200
24 2022-08-23          US    CAD      1.294800
25 2022-08-23          US    CHF      0.963100
26 2022-08-23          US    CNY      6.835000
27 2022-08-23          US    COP   4367.030000
28 2022-08-23          US    DKK      7.453400
29 2022-08-23          US    EUR      0.998000
30 2022-08-23          US    GBP      1.185000
31 2022-08-23          US    JPY    136.430000
32 2022-08-23          US    KRW   1345.615000
33 2022-08-23          US    MXN     19.993300
34 2022-08-23          US    NOK      9.703900
35 2022-08-23          US    NZD      0.622648
36 2022-08-23          US    PEN      3.857700
37 2022-08-23          US    SEK     10.601700
38 2022-08-23          US    USD    917.275000
39 2022-08-23          US    ZAR      0.058935
```

[6]:
```
print("Resultado Query Template:")
display(result_template)
```

```
Resultado Query Template:
```

```
        fecha   localidad moneda         valor
0  2022-08-22  BR            BRL    181.420000
1  2022-08-22          CL    AUD    647.971200
2  2022-08-22          CL    BRL    182.851500
3  2022-08-22          CL    CAD    722.310800
4  2022-08-22          CL    CHF    977.325500
5  2022-08-22          CL    CLD    945.350000
6  2022-08-22          CL    CLF  33700.210000
7  2022-08-22          CL    CNY    137.615900
8  2022-08-22          CL    COP      0.214900
9  2022-08-22  CL            DEG   1312.040000
```

(continué en la próxima página)

(proviene de la página anterior)

```
10 2022-08-22          CL    DKK    125.856000
11 2022-08-22          CL    EUR    936.073600
12 2022-08-22          CL    GBP   1109.623000
13 2022-08-22          CL    JPY      6.856800
14 2022-08-22          CL    KRW      0.703400
15 2022-08-22          CL    MXN     46.806900
16 2022-08-22          CL    NOK     95.832900
17 2022-08-22  CL              NZD    581.150000
18 2022-08-22          CL    PEN    243.199600
19 2022-08-22          CL    SEK     87.926400
20 2022-08-22          CL    USD    942.435000
21 2022-08-22  CL              ZAR     55.410000
22 2022-08-22          US    AUD      0.687600
23 2022-08-22          US    BRL      5.154100
24 2022-08-22          US    CAD      1.304800
25 2022-08-22          US    CHF      0.964300
26 2022-08-22          US    CNY      6.848300
27 2022-08-22          US    COP   4385.330000
28 2022-08-22          US    DKK      7.488200
29 2022-08-22          US    EUR      0.993300
30 2022-08-22          US    GBP      1.177400
31 2022-08-22          US    JPY    137.445000
32 2022-08-22          US    KRW   1339.800000
33 2022-08-22          US    MXN     20.134600
34 2022-08-22          US    NOK      9.834200
35 2022-08-22          US    NZD      0.616647
36 2022-08-22          US    PEN      3.875200
37 2022-08-22          US    SEK     10.718500
38 2022-08-22          US    USD    942.435000
39 2022-08-22          US    ZAR      0.058794
```

1.2.1.3 ¿Como manipular los datos?

Las queries directas retornan un objeto dataframe del paquete Pandas por lo que son manipuladas utilizando los comandos especificados por el proveedor.

a) Extracción de una o mas columnas

[7]:
```
selected_data = result_template.loc[:, ["valor"]]
```

[8]:
```
result_template.to_excel(r'prueba.xlsx')
```

[9]:
```
display(selected_data)
```

```
          valor
0    181.420000
1    647.971200
2    182.851500
3    722.310800
```

(continué en la próxima página)

(proviene de la página anterior)

```
4     977.325500
5     945.350000
6   33700.210000
7     137.615900
8       0.214900
9    1312.040000
10    125.856000
11    936.073600
12   1109.623000
13      6.856800
14      0.703400
15     46.806900
16     95.832900
17    581.150000
18    243.199600
19     87.926400
20    942.435000
21     55.410000
22      0.687600
23      5.154100
24      1.304800
25      0.964300
26      6.848300
27   4385.330000
28      7.488200
29      0.993300
30      1.177400
31    137.445000
32   1339.800000
33     20.134600
34      9.834200
35      0.616647
36      3.875200
37     10.718500
38    942.435000
39      0.058794
```

b) Búsqueda de un dato en especifico

[10]:
```
selected_data = result_template.loc[result_template['moneda'] == 'USD']
```

[11]:
```
display(selected_data)
```

```
        fecha localidad moneda    valor
20 2022-08-22        CL    USD  942.435
38 2022-08-22        US    USD  942.435
```

c) Obtener data de acuerdo a localización

[12]:
```
selected_data = result_template.loc[1][3]
```

[13]:
```
display(selected_data)
```
```
647.9712
```

1.2.2 Uso de lector de base de datos

El lector de base de datos permite obtener datos directamente de la base de datos oficial utilizando queries predefinidas y transformando los datos en objetos propios de la librería

1.2.2.1 Carga de librerías

[1]:
```
from finrisklib.data.dbreader import DBReader # Lector de Base de datos
from finrisklib.enums import FinancialIndex
from finrisklib.enums import Source
from IPython.display import display # Formateo
from datetime import date
```

1.2.2.2 Inicialización de lector

[2]:
```
reader = DBReader()
```

1.2.2.3 Datos de Curvas

[3]:
```
curve_name = 'CLP fx'
process_date = date(year=2022, month=8, day=23)

curve = reader.get_single_curve_data(curve_name=curve_name, process_date=process_date,␣
↪source=Source.SANDBOX)
```

a) Mostrar como factores de descuento

[4]:
```
df = curve.to_dataframe()
display(df)
```
```
      Tenor  Discount Factor
0       8.0         0.998063
1      15.0         0.995990
2      34.0         0.991161
3      63.0         0.981989
4      94.0         0.973400
5     126.0         0.964291
6     155.0         0.955910
```

(continué en la próxima página)

(proviene de la página anterior)

```
7     188.0         0.947108
8     275.0         0.926399
9     367.0         0.906843
10    552.0         0.879410
11    736.0         0.849496
12   1100.0         0.806704
13   1465.0         0.769472
14   1830.0         0.730337
15   2198.0         0.695461
16   2563.0         0.666686
17   2927.0         0.634250
18   3291.0         0.605523
19   3657.0         0.578067
20   4389.0         0.533318
21   5483.0         0.474120
22   7309.0         0.408970
23  10963.0         0.337719
```

b) Mostrar como tasas

[5]:
```
df = curve.rates_to_dataframe()
display(df)
```

```
      Tenor     Rates
0       8.0  0.091191
1      15.0  0.101231
2      34.0  0.098571
3      63.0  0.109446
4      94.0  0.108770
5     126.0  0.109480
6     155.0  0.110409
7     188.0  0.109667
8     275.0  0.105259
9     367.0  0.100672
10    552.0  0.087419
11    736.0  0.083052
12   1100.0  0.072828
13   1465.0  0.066513
14   1830.0  0.063770
15   2198.0  0.061288
16   2563.0  0.058600
17   2927.0  0.057598
18   3291.0  0.056410
19   3657.0  0.055434
20   4389.0  0.052915
21   5483.0  0.050220
22   7309.0  0.045023
23  10963.0  0.036290
```

1.2.2.4 Datos de tipos de cambio

a) Obtención de tipo de cambio definiendo dos monedas

[6]:
```
primary_currency = 'EUR'
secondary_currency = 'USD'

process_date = date(year=2022, month=8, day=23)

tc = reader.get_fx_rate(primary_currency=primary_currency, secondary_currency=secondary_
↪currency,
                        process_date=process_date, source=Source.SANDBOX)

print(tc)
```

```
0.99795
```

1.2.2.5 Datos de índices

[7]:
```
sofr = FinancialIndex.SOFR
icp = FinancialIndex.ICP

start_date = date(year=2022, month=8, day=1)
end_date = date(year=2022, month=8, day=23)

icp_data = reader.get_single_index_dataset(index=icp, start_date=start_date, end_
↪date=end_date)
sofr_data = reader.get_single_index_dataset(index=sofr, start_date=start_date, end_
↪date=end_date)
```

[8]:
```
idx_date = date(year=2022, month=8, day=10)
display(icp_data.get_index_data(index=icp, idx_date=idx_date))
```

```
20040.63
```

[9]:
```
idx_date = date(year=2022, month=8, day=9)
display(sofr_data.get_index_data(index=sofr, idx_date=idx_date))
```

```
0.0229
```

1.2.3 Uso y manipulación de curvas

En este archivo se muestra como:

- Obtener curvas desde la base de datos.
- Graficar la curva con factores de descuento y tasa.
- Calcular una curva sintética.
- Crear una curva con configuración personalizada.

1.2.3.1 Configuración Inicial

Comenzamos cargando las librerías necesarias

[1]:
```
from finrisklib.data.dbreader import DBReader
from finrisklib.market.curve import generate_curve_config
from finrisklib.market.curve import Curve
from finrisklib.enums import InterpolationMethod
from finrisklib.enums import ExtrapolationMethod
from finrisklib.enums import Compounding
from finrisklib.enums import DayCount
from finrisklib.enums import Source
from datetime import date
import matplotlib.pyplot as plt
```

1.2.3.2 Carga desde base de datos

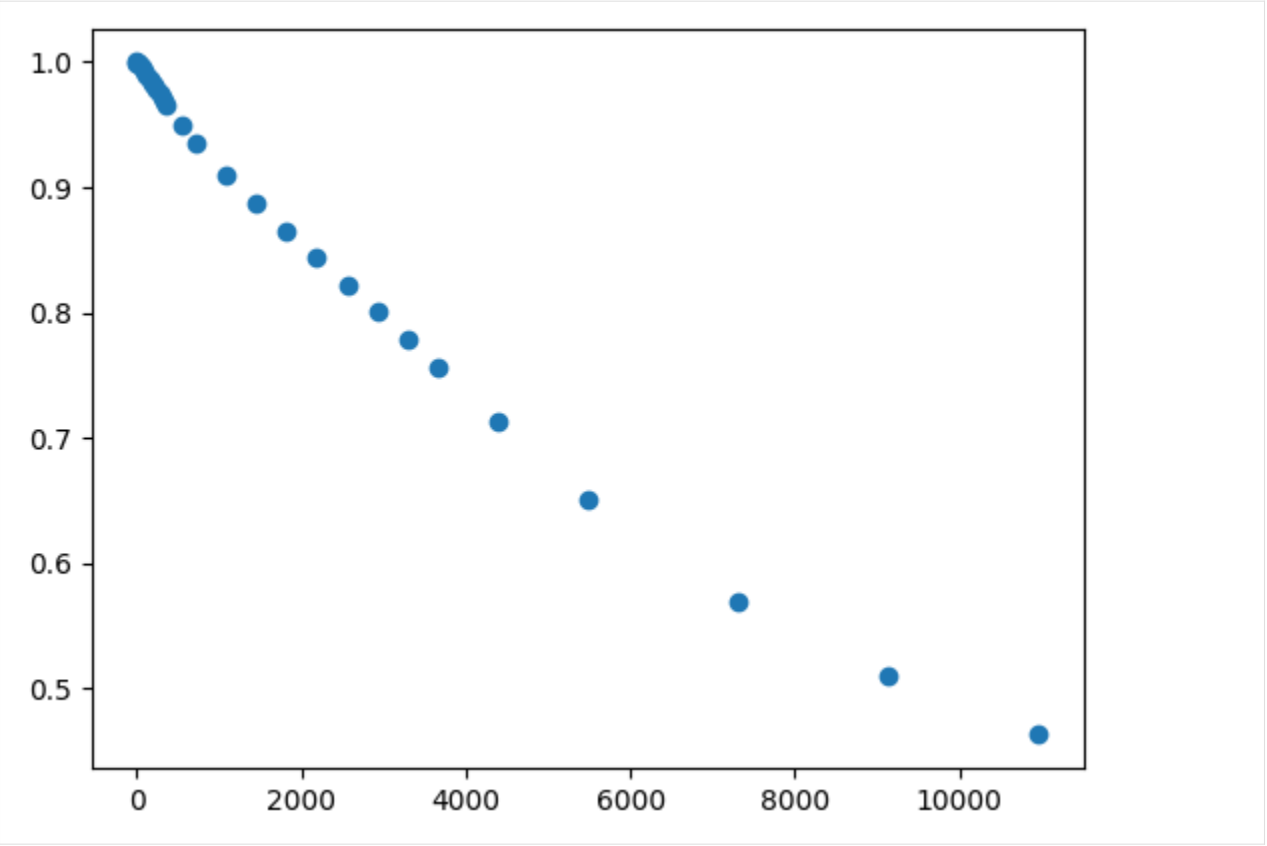
Para cargar una curva desde base de datos son necesarios dos datos, la fecha y el código de la curva. Adicionalmente, se puede elegir entre cargar las curvas de Murex (Default) o las curvas oficiales.

[2]:
```
# Definimos la fecha a la que se cargarán los datos
curve_date = date(year=2022, month=8, day=23)

# Códigos de las curvas a cargar
murex_code = 'USD SOFR'
bac_code = 'CURVA_USD_USA'

# Inicializamos el lector de base de datos
reader = DBReader()

# Obtención de curvas
murex_curve = reader.get_single_curve_data(curve_name=murex_code, process_date=curve_
↪date, source=Source.SANDBOX)
bac_curve = reader.get_single_curve_data(curve_name=bac_code, process_date=curve_date,␣
↪source=Source.OFFICIAL)
```

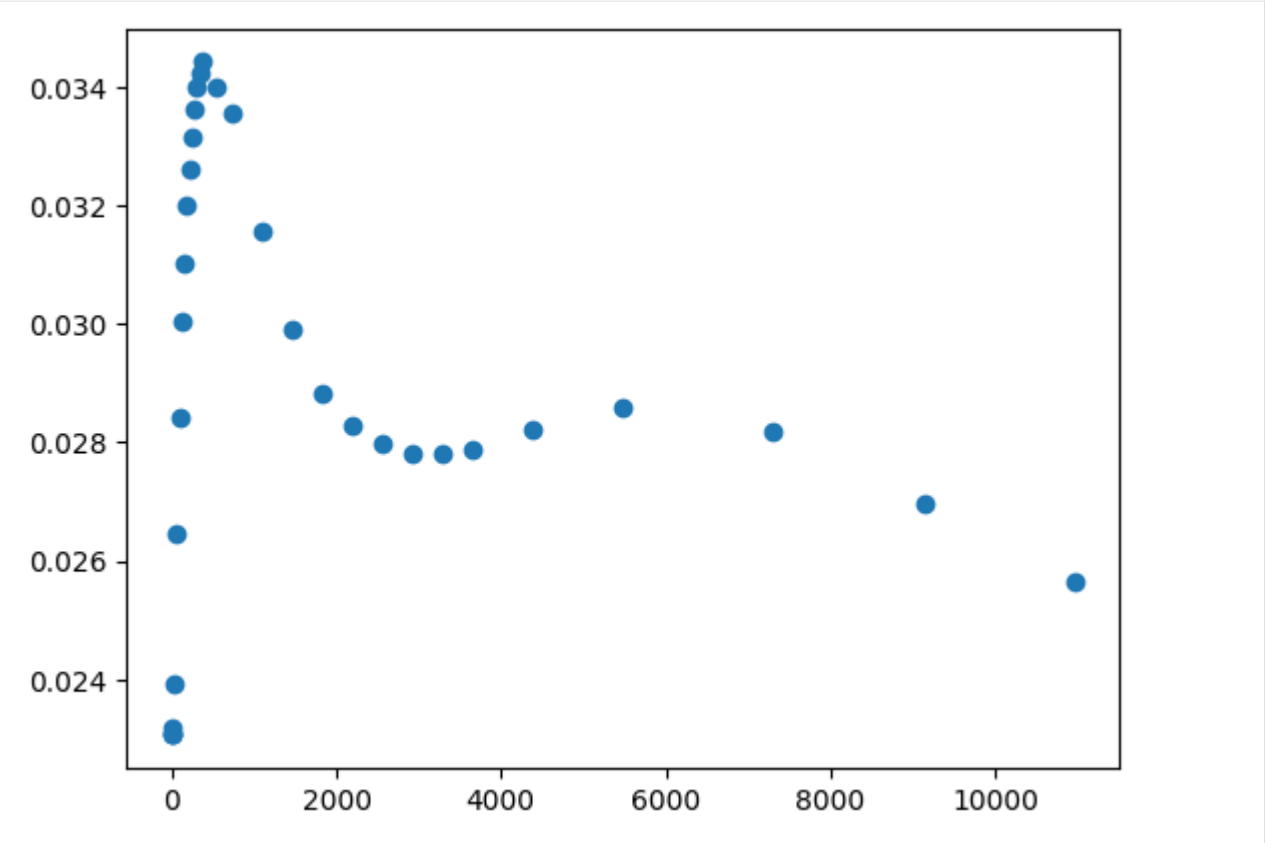
1.2.3.3 Graficar curva

Habiendo obtenido las curvas desde base de datos procedemos a graficarlas:

[3]:
```
# Gráfico de factores de descuento

curve_dict = murex_curve.to_dict()

tenors = curve_dict.keys()
dfs = curve_dict.values()
plt.scatter(tenors, dfs)
plt.show()
```



[4]:
```
# Gráfico de tasas

rates_dict = murex_curve.rates_to_dict()

tenors = rates_dict.keys()
rates = rates_dict.values()

plt.scatter(tenors, rates)
plt.show()
```



1.2.3.4 Calculo de curva sintética

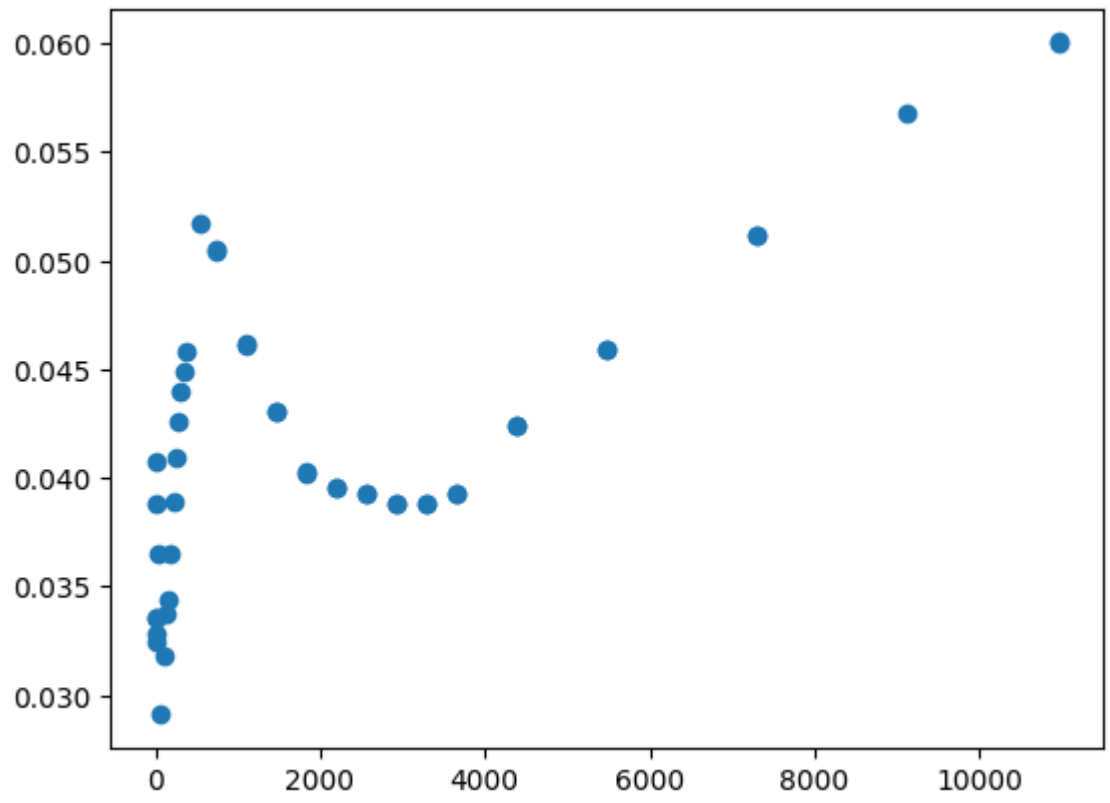
Las curvas sintéticas son calculadas automáticamente cuando se genera cualquier tipo de operación entre curvas, en este caso calcularemos la curva USD Local de manera sintética.

[5]:
```
# Necesitamos 3 curvas para
clp_us_code = 'CLP FX'
usd_us_code = 'USD SOFR'
clp_cl_code = 'CLP_CAM'

# Carga de datos
reader.mx_origin = True
CLP_US = reader.get_single_curve_data(curve_name=clp_us_code, process_date=curve_date,␣
↪source=Source.SANDBOX)
USD_US = reader.get_single_curve_data(curve_name=usd_us_code, process_date=curve_date,␣
↪source=Source.SANDBOX)
CLP_CL = reader.get_single_curve_data(curve_name=clp_cl_code, process_date=curve_date,␣
↪source=Source.SANDBOX)

# Creación de curva sintética
USD_CL = (USD_US/CLP_US)*CLP_CL

# Graficamos resultado
tenors, rates = USD_CL.get_rate_curve()
```

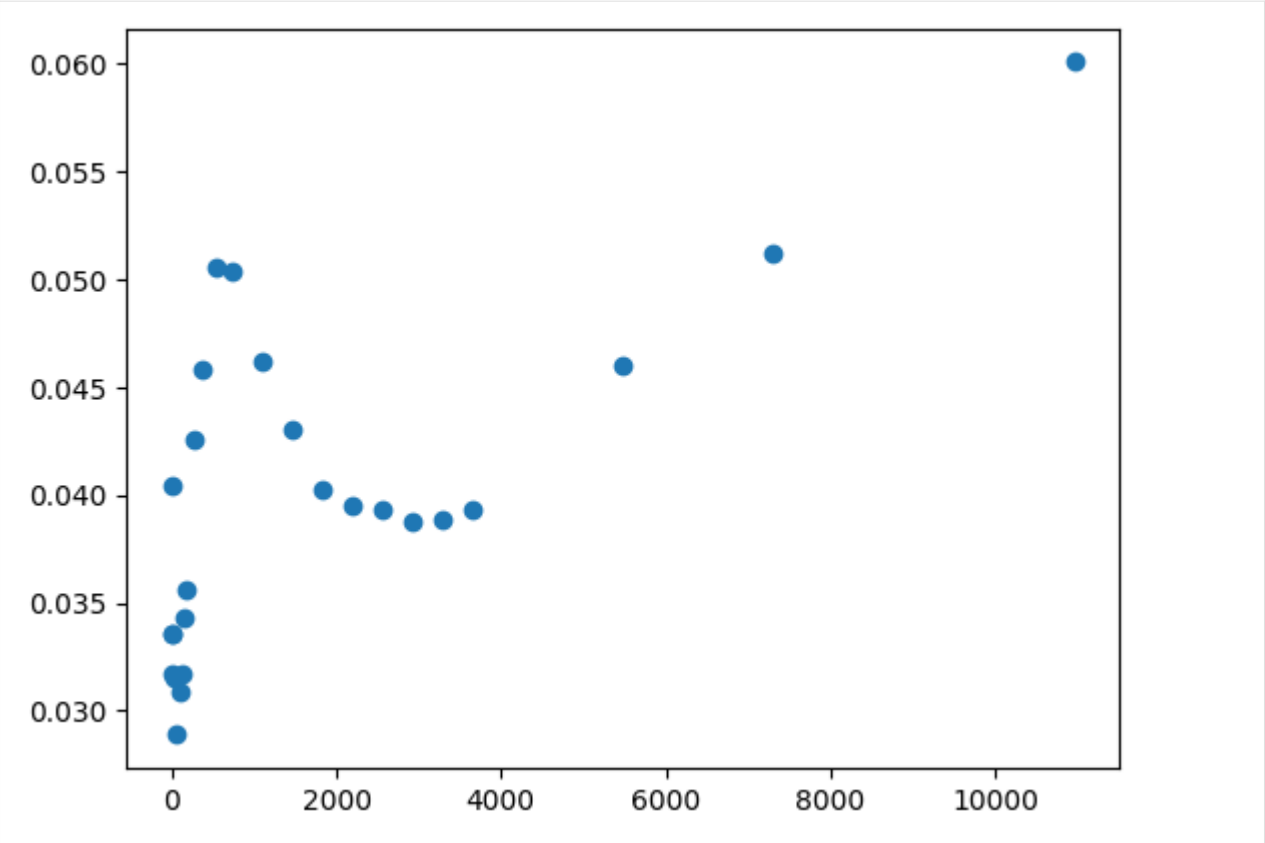
(continué en la próxima página)

(proviene de la página anterior)

```
plt.scatter(tenors, rates)
plt.show()
```



[6]:
```
# Comparación con Curva Oficial

USD_CL = reader.get_single_curve_data(curve_name="CURVA_USD_CL", process_date=curve_date,
↪ source=Source.OFFICIAL)
# Graficamos resultado
tenors, rates = USD_CL.get_rate_curve()

plt.scatter(tenors, rates)
plt.show()
```



1.2.3.5 Curva con Configuración personalizada

Para generar una curva es necesario proveer un vector de tenors y factores de descuento con una fecha y nombre asociado, adicionalmente se puede proveer una configuración personalizada.

[7]:
```
# Datos a utilizar
tenors = [8, 15, 32, 63, 94, 126, 156, 185, 275, 367, 550, 735, 1099, 1463, 1828, 2194,␣
↪2559, 2926, 3290, 3655, 4385, 5481, 7308, 10962]
dfs = [0.998310470000, 0.997016240000, 0.993143450000, 0.986344690000, 0.979086630000, 0.
↪971138990000, 0.964269270000, 0.958115730000,
0.938463220000, 0.923226950000, 0.898902260000, 0.873595740000, 0.834128950000, 0.
↪793526060000, 0.749730130000, 0.711597520000,
0.671697820000, 0.635942680000, 0.603079810000, 0.570977400000, 0.527430220000, 0.
↪462406640000, 0.389123040000, 0.306809400000]

# Configuración personalizada
custom_config = generate_curve_config(interpolation_method=InterpolationMethod.LINEAR,
                                      low_point_extrapolation_method=ExtrapolationMethod.
↪SLOPE,
                                      high_point_extrapolation_
↪method=ExtrapolationMethod.FLAT,
                                      rate_compounding=Compounding.YIELD,
                                      rate_day_count=DayCount.DC_30_360)

# Creación de curvas
```

(continué en la próxima página)

(proviene de la página anterior)

```
default_curve = Curve(curve_name="Default Config Curve", process_date=curve_date,␣
↪tenors=tenors, discount_factors=dfs)
custom_curve = Curve(curve_name="Default Config Curve", process_date=curve_date,␣
↪tenors=tenors, discount_factors=dfs, curve_config=custom_config)
```

[8]:
```
# Comparación 1

d1 = default_curve.get_discount_factor(32)
d2 = custom_curve.get_discount_factor(32)

print("default:{d1} - custom:{d2}".format(d1=d1, d2=d2))
```

```
default:0.99314345 - custom:0.99314345
```

[9]:
```
# Comparación 2

d1 = default_curve.get_rate(5)
d2 = custom_curve.get_rate(5)

print("default:{d1} - custom:{d2}".format(d1=d1, d2=d2))
```

```
default:0.07906308306385257 - custom:0.08519133793064726
```

[10]:
```
# Comparación 3

d1 = default_curve.get_discount_factor(35*365)
d2 = custom_curve.get_discount_factor(35*365)

print("default:{d1} - custom:{d2}".format(d1=d1, d2=d2))
```

```
default:0.2891814894739244 - custom:0.3068094
```

1.2.4 Uso de calendario

La clase calendario permite la manipulación de fechas y la obtención de distancia entre las mismas, la clase se encuentra basada en los calendarios default de la libería pandas

1.2.4.1 Carga de librerías

[1]:
```
from finrisklib.market.tradingcalendar import TradingCalendar # Clase base de calendario
from finrisklib.enums import Locality
from finrisklib.enums import BusinessDay
from finrisklib.enums import Period
from finrisklib.enums import DayCount
from finrisklib.data.dbreader import DBReader
from datetime import date
from pandas.tseries.holiday import AbstractHolidayCalendar, Holiday, GoodFriday, nearest_
↪workday # Calendario Pandas
```

1.2.4.2 Creación de un calendario personalizado

[2]:
```
locations = [Locality.CHL, Locality.COL, Locality.USA]

rules = [GoodFriday,
         Holiday('NewYearsDay', month=1, day=1, observance=nearest_workday),
         Holiday('Christmas', month=12, day=25, observance=nearest_workday)]

holidays = [date(year=2022, month=9, day=16)]

base_calendar = AbstractHolidayCalendar(name='STG-BGT-NY', rules=rules)
custom_calendar = TradingCalendar(calendar_name='STG-BGT-NY', holidays=holidays, base_
↪calendar=base_calendar)
```

a) Ajustar fecha laborable

[3]:
```
date1 = date(year=2022, month=9, day=16)

adjusted_date = custom_calendar.adjust_workday(workday=date1, business_day_
↪convention=BusinessDay.MODIFIED_FOLLOWING)
print(adjusted_date)
```

```
2022-09-19
```

1.2.4.3 Utilizar Calendarios Default

[4]:
```
reader = DBReader()
calendar_dataset = reader.get_calendar_dataset()

nyse_calendar = calendar_dataset.get_calendar([Locality.USA])
stg_calendar = calendar_dataset.get_calendar([Locality.CHL])
```

[5]:
```
asuncion_virgen = date(year=2022, month=8, day=15)

adjusted_date = nyse_calendar.adjust_workday(workday=asuncion_virgen, business_day_
↪convention=BusinessDay.MODIFIED_FOLLOWING)
print(adjusted_date)
```

```
2022-08-15
```

[6]:
```
asuncion_virgen = date(year=2022, month=8, day=15)

adjusted_date = stg_calendar.adjust_workday(workday=asuncion_virgen, business_day_
↪convention=BusinessDay.MODIFIED_FOLLOWING)
print(adjusted_date)
```

```
2022-08-16
```

1.2.4.4 Funcionalidades Estáticas

a) Offset de fechas

[7]:
```
start_date = date(year=2022, month=8, day=23)
n_periods = 6
period = Period.MONTH
end_date = nyse_calendar.offset_date(start_date=start_date, n_periods=n_periods,␣
↪period=period,
                                     convention=BusinessDay.MODIFIED_FOLLOWING)

print(f'Start Date: {start_date} - End Date {end_date}')
```

```
Start Date: 2022-08-23 - End Date 2023-02-23
```

b) Obtener fracción de año

[8]:
```
yf = TradingCalendar.get_year_fraction(start_date=start_date, end_date=end_date, day_
↪count=DayCount.DC_ACT_360)
print(f'year fraction: {yf}')
```

```
year fraction: 0.5111111111111111
```

1.2.5 Uso de Dataset

La clase dataset contiene toda la información e inteligencia de mercado

1.2.5.1 Carga de librerías

[1]:
```
from finrisklib.market.fxrate import FxPair
from finrisklib.data.dbreader import DBReader
from finrisklib.enums import FinancialIndex
from finrisklib.enums import Currency

from datetime import date
from IPython.display import display # Formateo
```

1.2.5.2 Carga de Dataset desde base de datos

[2]:
```
market_date = date(year=2022, month=8, day=24)

reader = DBReader()
dataset = reader.get_dataset(process_date=market_date)
```

1.2.5.3 Funcionalidades Fx

a) Obtención de spot descontado

[3]:
```
fx = FxPair.generate('USDCLP')
value = dataset.get_fx_rate(fx_pair=fx)

print(value)
```

```
916.8005004691394
```

b) Obtención de spot no descontado

[4]:
```
dataset.disable_discounted_spot()
value = dataset.get_fx_rate(fx_pair=fx)

print(value)
dataset.enable_discounted_spot()
```

```
916.965
```

c) Obtención de tipo de cambio proyectado

[5]:
```
forward_date = date(year=2022, month=9, day=22)

value = dataset.get_fx_forward_rate(fx_pair=fx, forward_date=forward_date)
print(value)
```

```
922.2345691064787
```

1.2.5.4 Funcionalidades Curvas

a) Obtención de curva de acuerdo a colateral

[6]:
```
currency = Currency.CLP
collateral = Currency.USD

curve = dataset.get_discount_curve(associated_currency=currency, associated_
↪collateral=collateral)

display(curve.rates_to_dataframe())
```

```
      Tenor     Rates
0       1.0  0.091203
1       2.0  0.091226
2      96.0  0.109755
3     187.0  0.110720
4     275.0  0.105625
5     369.0  0.098712
```

(continué en la próxima página)

(proviene de la página anterior)

```
6     551.0  0.088595
7     733.0  0.083693
8    1098.0  0.073127
9    1463.0  0.066843
10   1828.0  0.064083
11   2196.0  0.061675
12   2560.0  0.059306
13   2924.0  0.057613
14   3289.0  0.056305
15   3655.0  0.055658
16   4387.0  0.052918
17   5481.0  0.050245
18   7307.0  0.045263
19  10960.0  0.036617
```

[7]:
```
currency = Currency.CLP
collateral = Currency.CLP

curve = dataset.get_discount_curve(associated_currency=currency, associated_
↪collateral=collateral)

display(curve.rates_to_dataframe())
```

```
      Tenor     Rates
0       1.0  0.102367
1       2.0  0.102393
2      96.0  0.112523
3     187.0  0.114707
4     275.0  0.115283
5     369.0  0.113394
6     551.0  0.106755
7     733.0  0.101230
8    1098.0  0.088235
9    1463.0  0.080234
10   1828.0  0.075787
11   2196.0  0.073229
12   2560.0  0.070940
13   2924.0  0.068950
14   3289.0  0.067660
15   3655.0  0.067222
16   4387.0  0.067559
17   5481.0  0.067897
18   7307.0  0.068521
19  10960.0  0.071195
```

b) Obtener curva de proyección

[8]:
```
idx = FinancialIndex.ICP

curve = dataset.get_projection_curve(associated_index=idx)
display(curve.rates_to_dataframe())
```

```
      Tenor     Rates
0       1.0  0.102367
1       2.0  0.102393
2      96.0  0.112523
3     187.0  0.114707
4     275.0  0.115283
5     369.0  0.113394
6     551.0  0.106755
7     733.0  0.101230
8    1098.0  0.088235
9    1463.0  0.080234
10   1828.0  0.075787
11   2196.0  0.073229
12   2560.0  0.070940
13   2924.0  0.068950
14   3289.0  0.067660
15   3655.0  0.067222
16   4387.0  0.067559
17   5481.0  0.067897
18   7307.0  0.068521
19  10960.0  0.071195
```

a) Obtención de índice a una fecha específica

[9]:
```
idx = FinancialIndex.ICP
idx_date = date(year=2022, month=8, day=19)

idx_value = dataset.get_index_data(index=idx, idx_date=idx_date)
print(idx_value)
```

```
20089.53
```

b) Obtención de data histórica de indice

[10]:
```
idx = FinancialIndex.ICP
start_date = date(year=2022, month=8, day=1)
end_date = date(year=2022, month=8, day=23)

idx_value = dataset.get_index_data_between(index=idx, start_date=start_date, end_
↪date=end_date)
display(idx_value)
```

```
{datetime.date(2022, 8, 1): 19991.85,
 datetime.date(2022, 8, 2): 19997.26,
```

(continué en la próxima página)

(proviene de la página anterior)

```
 datetime.date(2022, 8, 3): 20002.68,
 datetime.date(2022, 8, 4): 20008.1,
 datetime.date(2022, 8, 5): 20013.52,
 datetime.date(2022, 8, 8): 20029.78,
 datetime.date(2022, 8, 9): 20035.2,
 datetime.date(2022, 8, 10): 20040.63,
 datetime.date(2022, 8, 11): 20046.06,
 datetime.date(2022, 8, 12): 20051.49,
 datetime.date(2022, 8, 16): 20073.21,
 datetime.date(2022, 8, 17): 20078.65,
 datetime.date(2022, 8, 18): 20084.09,
 datetime.date(2022, 8, 19): 20089.53,
 datetime.date(2022, 8, 22): 20105.85,
 datetime.date(2022, 8, 23): 20111.3}
```

1.2.6 Construcción y valorización de producto Forward

1.2.6.1 Carga de Librerías

[1]:
```
from finrisklib.instruments.forward import Forward
from finrisklib.instruments.flow import Cash
from finrisklib.enums import Currency
from finrisklib.data.dbreader import DBReader
from datetime import date
```

1.2.6.2 Construcción

[2]:
```
# Nocionales (Operación 6812159)
active_notional = Cash(amount=225000, currency=Currency.USD)
passive_notional = Cash(amount=195558750, currency=Currency.CLP)

# Fechas
start_date = date(year=2022, month=6, day=14)
end_date = date(year=2022, month=7, day=18)
fixing_date = date(year=2022, month=7, day=18)
pay_date = date(year=2022, month=7, day=18)

# Construcción del instrumento

op_6812159 = Forward(active_notional=active_notional, passive_notional=passive_notional,
                     start_date=start_date, end_date=end_date, pay_date=pay_date, fixing_
↪date=fixing_date, associated_collateral=Currency.USD)
```

1.2.6.3 Valorización

a) Valorización con moneda default

[3]:
```
# Fecha de valorización
valuation_date = date(year=2022, month=7, day=11)

# Inicializamos el lector de base de datos
reader = DBReader()

dataset = reader.get_dataset(process_date=valuation_date)

val = op_6812159.valuate(dataset=dataset)

print(val)
```

```
28,536,787.95 CLP
```

b) Valorización con moneda especificada

[4]:
```
dataset.set_default_valuation_currency(valuation_currency=Currency.USD)

val = op_6812159.valuate(dataset=dataset)
print(val)

dataset.set_default_valuation_currency(valuation_currency=Currency.CLP)
```

```
28,688.56 USD
```

1.2.6.4 Obtención desde base de datos

a) Obtención de datos

[10]:
```
id_number = 6812159

db_op = reader.get_operation(id_number=id_number, process_date=valuation_date)
val = db_op.valuate(dataset=dataset)

print(val)
```

```
28,533,606.60 CLP
```

b) Obtención de MtM Oficial

[11]:
```
mtm = reader.get_official_mtm(id_number=id_number, process_date=valuation_date)

print(mtm)
```

```
28,541,754.30 CLP
```

1.2.7 Construcción y valorización de Swaps

1.2.7.1 Carga de librerías

[1]:
```
from finrisklib.enums import Currency
from finrisklib.data.dbreader import DBReader
from datetime import date
```

1.2.7.2 Obtención desde base de datos

a) Obtención de Instrumento

[2]:
```
# Fecha de valorización
valuation_date = date(year=2022, month=8, day=23)

# Inicializamos el lector de base de datos
reader = DBReader()

id_number = 1001268
op = reader.get_operation(id_number=id_number, process_date=valuation_date)
```

1.2.7.3 Valorización

a) Valorización independiente

[3]:
```
dataset = reader.get_dataset(process_date=valuation_date)

val = op.valuate(dataset=dataset)

print(val)
```

```
-453,219,506.79 CLP
```

[4]:
```
# Fecha de valorización
valuation_date2 = date(year=2022, month=8, day=24)
dataset2 = reader.get_dataset(process_date=valuation_date2)
```

[5]:
```
dataset2.set_default_valuation_currency(Currency.USD)

val2 = op.valuate(dataset=dataset2)

print(val2)
```

```
-495,699.55 USD
```

b) Valorización de sistema

[6]:
```
sys_val = reader.get_official_mtm(id_number=id_number, process_date=valuation_date)
print(sys_val)
```

```
-461,698,346.00 CLP
```

[7]:
```
dif = sys_val-val
print(dif)
```

```
-8,478,839.21 CLP
```

CAPÍTULO 2

Indices y tablas

- genindex
- modindex
- search

Índice de Módulos Python