

Swap_Pricing

January 16, 2018

1 Get discount factor for JPY

- input: MoneyMarket (short term interest rate), Swap rate.
- output: discount factors for each tenor listed by MoneyMarket and Swap rate.

1.1 Pricing

1.1.1 Swap pricing formula

The value of the exchange between a float and a fixed side is given by

$$V = \sum_{i=1}^N L(t_{i-1}, t_i) \times DF(t_i) \times \delta_i - \sum_{i=1}^N SwapRate \times DF(t_i) \times \delta_i,$$

where $L(t_{i-1}, t_i)$ is the float interest rate between t_{i-1} and t_i , $DF(t_i)$ is a discount factor, δ_i is a day-count-fraction and $SwapRate$ is a Swap rate which means a par rate for a swap trade.

1.1.2 Bootstrap method for getting discount factors

Discount factors as of today can be estimated from a par swap trade which corresponds to $V = 0$ under swap pricing formula. For example, let us consider a swap trade with maturity of 1.5 year. The discount factor for 1.5 year $DF(t_{1.5Y})$ is calculated by solving the following equation:

$$\sum_{i=1}^3 L(t_{i-1}, t_i) \times DF(t_i) \times \delta = \sum_{i=1}^3 SwapRate(1.5Y) \times DF(t_i) \times \delta$$

where a quoted swap rate is used for $SwapRate(1.5Y)$, the day-count-fraction δ is assumed 6 month and the float side interest rate is assumed that a following model expressed as

$$L(t_{i-1}, t_i) = \frac{1}{\delta} \left(\frac{DF(t_{i-1})}{DF(t_i)} - 1 \right).$$

The above equation can be solved by using $DF(t_{0.5Y})$, $DF(t_{1.0Y})$ and the float interest rate which is defined as above equation. As a result, the discount factor $DF(t_{1.5Y})$ is given by

$$DF(t_{1.5Y}) = \frac{1}{(1 + \delta \times SwapRate(1.5Y))} \left(DF(t_0) - SwapRate(1.5Y) \times \delta \times (DF(t_{0.5Y}) + DF(t_{1.0Y})) \right),$$

where $DF(t_{0.5Y})$ and $DF(t_{1.0Y})$ is calculated by using a quoted LIBOR (the rate of Money Market). The short rate of Money Market means spot rate, where the cashflows is expressed as only two terms. For example, $DF(t_{0.5Y})$ is given by

$$DF(t_{0.5Y}) = \frac{1}{(1 + \delta \times L(0.0Y, 0.5Y))},$$

where $L(0.0Y, 0.5Y)$ is the LIBOR rate between today and 6 month later. Discount factors after $t_{1.5Y}$ can be calculated by the same way as the derivation of $DF(t_{1.5Y})$. This method of getting discount factors gradually is called Bootstrap method.

```
In [3]: import matplotlib.pyplot as plt
import numpy as np
import datetime

class getDF_moneymarket:
    ''' def __init__(self, libor_rate, start_day, end_day):
        self.libor_rate = libor_rate
        self.start_day = start_day
        self.end_day = end_day
        self.datetime_obj_start = datetime.datetime.strptime(start_day, '%Y/%m/%d')
        self.datetime_obj_end = datetime.datetime.strptime(end_day, '%Y/%m/%d')
        self.daycount = (self.datetime_obj_end - self.datetime_obj_start).days / 360
        self.discount_factor = 0
    '''

    def __init__(self, today, array_ccy):
        self._start_day = today

    def getDF(self, seq_moneymarket):

        File "<ipython-input-3-180ce8381581>", line 14
        ...

^
IndentationError: expected an indented block
```

```
In [4]: DF = getDF_moneymarket(0.2, '2017/12/18', '2019/12/30')
print(DF.discount_factor)
print(DF.getDF())
print(DF.discount_factor)
```

NameError

Traceback (most recent call last)

```

<ipython-input-4-20a95f09654c> in <module>()
----> 1 DF = getDF_moneymarket(0.2, '2017/12/18', '2019/12/30')
      2 print(DF.discount_factor)
      3 print(DF.getDF())
      4 print(DF.discount_factor)

```

NameError: name 'getDF_moneymarket' is not defined

```

In [5]: DF1 = getDF_moneymarket(0.3, '2017/12/18', '2018/3/20')
        DF1.getDF()

```

NameError Traceback (most recent call last)

```

<ipython-input-5-1243af6ebe67> in <module>()
----> 1 DF1 = getDF_moneymarket(0.3, '2017/12/18', '2018/3/20')
      2 DF1.getDF()

```

NameError: name 'getDF_moneymarket' is not defined

```

In [9]: %matplotlib inline
import numpy as np
import csv
import time
import datetime
import matplotlib.pyplot as plt

with open('sample_moneymarket.csv', 'r') as csvfile:
    reader_obj = csv.reader(csvfile)
    # rewritten header_obj by using next method(???)
    header_obj = next(reader_obj)
    mm_list = []
    for row in reader_obj:
        mm_list.append(row)

mm_list

```

```

def get_DF_MM(money_market_list):

```

```

list_len = len(money_market_list)
discount_factor = np.zeros(list_len*2).reshape(list_len, 2)
discount_factor = [["",0.0] for i in range(list_len)]
day_count_fraction = np.zeros(list_len)
# substitution the kind of trade
for i in range(0,list_len):
    discount_factor[i][0] = money_market_list[i][0]
# calc daycount-fraction
convention = 360.0
for i in range(0, len(day_count_fraction)):
    day_count_fraction[i] = calc_daycount(money_market_list[i][1], money_market_list[i][2])
# calculate DF of 0/N
discount_factor[0][1] = 1.0 / (1.0 + day_count_fraction[0] * float(money_market_list[0][3]))
# calculate DF of T/N
discount_factor[1][1] = discount_factor[0][1] / (1.0 + day_count_fraction[1]*float(money_market_list[1][3]))
# calculate DF after 1W
for i in range(2, list_len):
    discount_factor[i][1] = discount_factor[1][1] / (1.0 + day_count_fraction[i] * float(money_market_list[i][3]))
return discount_factor

def calc_daycount(start_day, end_day, convention):
    datetime_obj_start = datetime.datetime.strptime(start_day, '%Y/%m/%d')
    datetime_obj_end = datetime.datetime.strptime(end_day, '%Y/%m/%d')
    daycount = (datetime_obj_end - datetime_obj_start).days / convention
    return daycount

def draw_DF(seq_discount_factor):
    list_len = len(seq_discount_factor)
    seq_DF = np.zeros(list_len)
    for i in range(0, list_len):
        seq_DF[i] = seq_discount_factor[i][1]
    plt.plot(seq_DF)
    plt.ylim([0,1.0])

def get_DF_SW(money_market_list, swap_rate_list):
    DF_moneymarket = get_DF_MM(money_market_list)

list_discountfactor = get_DF(mm_list)
list_discountfactor
# draw_DF(list_discountfactor)

```

NameError

Traceback (most recent call last)

```

<ipython-input-9-4f4face75369> in <module>()
    60
    61
---> 62 list_discountfactor = get_DF(mm_list)
    63 list_discountfactor
    64 # draw_DF(list_discountfactor)

```

NameError: name 'get_DF' is not defined

```

In [82]: with open('sample_moneymarket.csv', 'r') as csvfile:
        reader_obj = csv.reader(csvfile)
        # rewritten header_obj by using next method(???)
        header_obj = next(reader_obj)
        mm_list = []
        for row in reader_obj:
            mm_list.append(row)

mm_list

```

```

Out[82]: [['O/N', '2017/12/23', '2017/12/24', '0.014348'],
          ['T/N', '2017/12/24', '2017/12/25', '0.014348'],
          ['1W', '2017/12/25', '2018/1/1', '0.014876'],
          ['2W', '2017/12/25', '2018/1/8', '0.015'],
          ['1M', '2017/12/25', '2018/1/24', '0.01563'],
          ['2M', '2017/12/25', '2018/2/23', '0.01616'],
          ['3M', '2017/12/25', '2018/3/25', '0.01685'],
          ['6M', '2017/12/25', '2018/6/23', '0.01833'],
          ['1Y', '2017/12/25', '2018/12/20', '0.021']]

```

2 1/15

- データの加工
 - 小数点表記 (" {:.1f} ".format())
 - 文字列の結合 (+でできる)
- 空のリスト作成
 - 内包表記 -> [5 for i in range(10)] -> 5 が 10 個のリスト

```

In [256]: with open('sample_swaprate.csv', 'r') as csvfile:
        reader_obj = csv.reader(csvfile)
        # rewritten header_obj by using next method(???)
        header_obj = next(reader_obj)
        swap_rate_list = []
        for row in reader_obj:

```

```

swap_rate_list.append(row)
temp_num = [[] for i in range(len(swap_rate_list))] # comprehension expression for
### proceccing the expression for the type of 1Y to 1.0Y.
for i in range(len(swap_rate_list)):
    if (len(swap_rate_list[i][0]) == 2):
        temp_num[i] = "{:.1f}".format(int(swap_rate_list[i][0][0])) + swap_rate_li
        swap_rate_list[i][0] = temp_num[i]
    elif (len(swap_rate_list[i][0]) == 3):
        temp_num[i] = "{:.1f}".format(int(swap_rate_list[i][0][0:2])) + swap_rate_
        swap_rate_list[i][0] = temp_num[i]
    else:
        break

```

swap_rate_list

```

Out[256]: [['1.0Y', '2017/12/25', '2018/12/25', '0.01904'],
           ['2.0Y', '2017/12/25', '2019/12/25', '0.02086'],
           ['3.0Y', '2017/12/25', '2020/12/24', '0.02187'],
           ['4.0Y', '2017/12/25', '2021/12/24', '0.02248'],
           ['5.0Y', '2017/12/25', '2022/12/24', '0.02295'],
           ['6.0Y', '2017/12/25', '2023/12/24', '0.02337'],
           ['7.0Y', '2017/12/25', '2024/12/23', '0.02376'],
           ['8.0Y', '2017/12/25', '2025/12/23', '0.02411'],
           ['9.0Y', '2017/12/25', '2026/12/23', '0.02444'],
           ['10.0Y', '2017/12/25', '2027/12/23', '0.02475'],
           ['15.0Y', '2017/12/25', '2032/12/21', '0.02582'],
           ['20.0Y', '2017/12/25', '2037/12/20', '0.02632'],
           ['30.0Y', '2017/12/25', '2047/12/18', '0.02646']]

```

```

In [220]: "{:.1f}".format(132)

```

```

Out[220]: '132.0'

```

```

In [90]: [[1] for i in range(4)]

```

```

Out[90]: [[1], [1], [1], [1]]

```

```

In [6]: calc_daycount(mm_list[0][1], mm_list[0][2], 360)
        calc_daycount(mm_list[5][1], mm_list[5][2], 360)

```

```

Out[6]: 0.16666666666666666

```

```

In [8]: float(mm_list[0][3])

```

```

Out[8]: 0.014348

```

3 1/15

- エクセルの Vlookup 風の作業

- 半年置きでテナーで、空の swap rate のリストを作成
- 外部データとして存在する、加工済みの (1Y->1.0Y) データとマッチする行はそのまま置き換え
- マッチしない行は据え置きでデフォルトの0を代入したままのリストを作成

```
In [276]: def get_DF_SW(money_market_list, swap_rate_list, tenor):
    seq_len_of_swap_rate = int(30/tenor - 1)
    array_swap_rate = [["", 0, 0, 0] for i in range(seq_len_of_swap_rate)]
    for i in range(2, seq_len_of_swap_rate + 2):
        array_swap_rate[i-2][0] = "{}Y".format(i*0.5)
    ## for sentence is nested...
    ## I wanna reviese code, but I have not an idea. Please tell me better coding if y
    for i in range(len(array_swap_rate)):
        for j in range(len(swap_rate_list)):
            if (array_swap_rate[i][0] in swap_rate_list[j][0]):
                array_swap_rate[i] = swap_rate_list[j]
                break

    print(array_swap_rate)
    return array_swap_rate
```

```
In [277]: get_DF_SW(mm_list, swap_rate_list, 1/2)
```

```
[['1.0Y', '2017/12/25', '2018/12/25', '0.01904'], ['1.5Y', 0, 0, 0], ['2.0Y', '2017/12/25', '2018/12/25', '0.02086'],
```

```
Out[277]: [['1.0Y', '2017/12/25', '2018/12/25', '0.01904'],
            ['1.5Y', 0, 0, 0],
            ['2.0Y', '2017/12/25', '2019/12/25', '0.02086'],
            ['2.5Y', 0, 0, 0],
            ['3.0Y', '2017/12/25', '2020/12/24', '0.02187'],
            ['3.5Y', 0, 0, 0],
            ['4.0Y', '2017/12/25', '2021/12/24', '0.02248'],
            ['4.5Y', 0, 0, 0],
            ['5.0Y', '2017/12/25', '2022/12/24', '0.02295'],
            ['5.5Y', 0, 0, 0],
            ['6.0Y', '2017/12/25', '2023/12/24', '0.02337'],
            ['6.5Y', 0, 0, 0],
            ['7.0Y', '2017/12/25', '2024/12/23', '0.02376'],
            ['7.5Y', 0, 0, 0],
            ['8.0Y', '2017/12/25', '2025/12/23', '0.02411'],
            ['8.5Y', 0, 0, 0],
            ['9.0Y', '2017/12/25', '2026/12/23', '0.02444'],
            ['9.5Y', 0, 0, 0],
            ['10.0Y', '2017/12/25', '2027/12/23', '0.02475'],
            ['10.5Y', 0, 0, 0],
            ['11.0Y', 0, 0, 0],
            ['11.5Y', 0, 0, 0],
            ['12.0Y', 0, 0, 0],
```

```

['12.5Y', 0, 0, 0],
['13.0Y', 0, 0, 0],
['13.5Y', 0, 0, 0],
['14.0Y', 0, 0, 0],
['14.5Y', 0, 0, 0],
['15.0Y', '2017/12/25', '2032/12/21', '0.02582'],
['15.5Y', 0, 0, 0],
['16.0Y', 0, 0, 0],
['16.5Y', 0, 0, 0],
['17.0Y', 0, 0, 0],
['17.5Y', 0, 0, 0],
['18.0Y', 0, 0, 0],
['18.5Y', 0, 0, 0],
['19.0Y', 0, 0, 0],
['19.5Y', 0, 0, 0],
['20.0Y', '2017/12/25', '2037/12/20', '0.02632'],
['20.5Y', 0, 0, 0],
['21.0Y', 0, 0, 0],
['21.5Y', 0, 0, 0],
['22.0Y', 0, 0, 0],
['22.5Y', 0, 0, 0],
['23.0Y', 0, 0, 0],
['23.5Y', 0, 0, 0],
['24.0Y', 0, 0, 0],
['24.5Y', 0, 0, 0],
['25.0Y', 0, 0, 0],
['25.5Y', 0, 0, 0],
['26.0Y', 0, 0, 0],
['26.5Y', 0, 0, 0],
['27.0Y', 0, 0, 0],
['27.5Y', 0, 0, 0],
['28.0Y', 0, 0, 0],
['28.5Y', 0, 0, 0],
['29.0Y', 0, 0, 0],
['29.5Y', 0, 0, 0],
['30.0Y', '2017/12/25', '2047/12/18', '0.02646']]

```

3.0.1 エラーメッセージ

3.0.2 解決策

- 数値と文字列が混ざっているのをどちらかに統一すべし