PremiumsInterestEtc

July 12, 2020

1 Investment comparison calculator

1.1 Created by Nancy Aggarwal on Jul 12, 2020

```
In [1]: from scipy.optimize import minimize
```

2 Pay regular premium, gain fixed interest

2.1 Frequency of Compounding interest = frequency of payments

Say one pays an amount x regularly for n time-intervals. Say one wants to get one's investement out after N intervals ($N >= n$). Say the compound rate of interest (calculated at the same frequency) is i.

Now, the amount after n intervals is:
$Y_n = x(1+i)\sum_{j=0}^{n-1}(1+i)^j$
Finally, the return after N intervals is:
$Z_N = Y_n * (1+i)^{N-n}$

2.2 Frequency of compounding interest > frequency of payments

Say one pays an amount x regularly for n time-intervals. Say one wants to get one's investement out after N intervals ($N >= n$). Say the compound rate of interest (calculated at the a frequency m) is i.

Now, the amount after n intervals is:
$Y_n = x(1+i)^m \sum_{j=0}^{n-1}(1+i)^{mj}$
Finally, the return after N intervals is:
$Z_N = Y_n * (1+i)^{m(N-n)}$

2.3 Implementation

```
In [2]: def calcTotalReturn(i,x,n,N,m):
            # n is years of premium in some time unit (say year or quarter)
            # N is years of maturity in the same time uint
            # i is interest in percent per that time unit
            # x is premium per that time unit
            # m is the number of times interest is compounded between two payments
        #     print("interest = {}, premium = {}, years of premium = {}, years of maturity = {
            i = i/100
```

```
        seriesarray = [(1+i)**(j*m) for j in range(n)]
        Yn = x*((1+i)**m)*sum(seriesarray)
        ZN = Yn*(1+i)**(m*(N-n))
        return ZN
```

2.4 Example

8% yearly interest, premium of every 6 months at the rate of 55/yr for 16 years, interest compounded every month. Policy matured after 25 years.

```
In [3]: intervalfactor = 2 #(convert years to semesters)
        compoundingfactor = 6 #months in a semester

        IntervalsOfPremium = 16*intervalfactor
        IntervalsOfMaturity = 25*intervalfactor
        RateofInterest = 8/(intervalfactor*compoundingfactor) #percent
        IntervalPremium = 55/intervalfactor

In [4]: calcTotalReturn(RateofInterest,IntervalPremium,IntervalsOfPremium,IntervalsOfMaturity,c

Out[4]: 3722.659639968529

In [ ]:
```

3 Now back-calculate interest given final sum

```
In [5]: FinalSum = 75*25 + 7.5e2 + 1e3

In [6]: FinalSum

Out[6]: 3625.0

In [7]: def err(i,tup):
            Model = calcTotalReturn(i[0],tup[0],tup[1],tup[2],tup[3])
            Meas = tup[4]
            error = abs((Model-Meas)/Meas)
        #     print(i,Model,Meas,error)
            return error

In [8]: minimizeResult=minimize(err,1,[IntervalPremium,IntervalsOfPremium,IntervalsOfMaturity,c

In [9]: inferredInterest=minimizeResult.x*intervalfactor*compoundingfactor

In [10]: inferredInterest

Out[10]: 7.858347053111679

In [ ]:
```