Effective interest rate using `R` and `Python`

Andrija Djurovic*

*www.linkedin.com/in/andrija-djurovic

The following example illustrates the calculation of the effective interest rate (EIR) using `R` and `Python`. We will begin by specifying the loan characteristics.

```
#loan amount
amount <- 10e3
#yearly nominal interest rate
ir.y <- 0.0599
#monthly nominal interest rate
ir.m <- ir.y / 12
#maturity (in months)
maturity <- 60
```

Let's further assume that the borrower has an initial cost, which is immediately deducted from the loan amount and proceed with EIR calculation.

```
#initial loan costs
cost <- 150
#annuity
annuity <- amount * ir.m /(1 - (1 + ir.m) ^ (-maturity))
annuity
```

```
## [1] 193.2815
```

```
#check npv of the cash flow
sum(rep(annuity, maturity) / cumprod(1 + rep(ir.m, maturity)))
```

```
## [1] 10000
```

```
#optimization function
eir.opt <- function(amount, annuity, maturity, ir, cost) {
      #amount - loan amount
      #annuity - annuity
      #ir - effective interest rate (monthly)
      #cost - initial cost

      #cash flow
      cf <- rep(annuity, maturity)
      #discounted cash flow
      df <- cumprod(1 + rep(ir, maturity))
      #optimization value
      opt.array <- sum(cf / df) - (amount - cost)
return(opt.array)
}
```

```r
#calculate the effective interest rate
eir <- uniroot(f = eir.opt,
               amount = amount,
               annuity = annuity,
               maturity = maturity,
               cost = cost,
               interval = c(0, 1))$root
#monthly effective interest rate
eir
```

```
## [1] 0.005515214
```

```r
#yearly effective interest rate
eir * 12
```

```
## [1] 0.06618257
```

```r
#check the cash flow under the effective interest rate
sum(rep(annuity, maturity) / cumprod(1 + rep(eir, maturity)))
```

```
## [1] 9850.453
```

Let's now replicate the same process in `Python`.

```python
import numpy as np
from scipy.optimize import root_scalar

#loan amount
amount = 10e3
#yearly nominal interest rate
ir_y = 0.0599
#monthly nominal interest rate
ir_m = ir_y / 12
#maturity (in months)
maturity = 60
#initial loan costs
cost = 150
#monthly annuity
annuity = amount * ir_m / (1 - (1 + ir_m) ** (-maturity))
annuity
```

```
## 193.28151998435652
```

```
#check npv of the cash flow
np.sum(np.repeat(annuity, maturity) /
np.cumprod(1 + np.repeat(ir_m, maturity)))
```

```
## 9999.999999999876
```

```
#optimization function
def eir_opt(ir, amount, annuity, maturity, cost):
    #amount - loan amount
    #annuity - annuity
    #ir - effective interest rate (monthly)
    #cost - initial cost

    #cash flow
    cf = np.repeat(annuity, maturity)
    #discounted cash flow
    df = np.cumprod(1 + np.repeat(ir, maturity))
    #optimization value
    opt_array = np.sum(cf / df) - (amount - cost)

    return opt_array

#calculate the effective interest rate
eir = root_scalar(f = eir_opt,
                  args = (amount, annuity, maturity, cost),
                  bracket = [0, 1],
                  x0 = ir_m).root
#monthly effective interest rate
eir
```

```
## 0.005516816331682868
```

```
#yearly effective interest rate
eir * 12
```

```
## 0.0662017959801944
```

```
#check the cash flow under the effective interest rate
np.sum(np.repeat(annuity, maturity) /
np.cumprod(1 + np.repeat(eir, maturity)))
```

```
## 9849.999999925732
```

Both `R` and `Python` offer numerous methods for computing the effective interest rate. Generally, this computation is analogous to calculating the internal rate of return, a functionality provided by various packages in each programming language.