Aprenda com quem faz

Fundamentos em Ciências de Dados de Renda Fixa

Marcelo Garcia

2022

SUMÁRIO

> Capítulo 1

Capítulo 1. Taxas de Juros

Juros

Quando você faz um empréstimo no banco, é exigido que você pague um valor maior do que o valor emprestado ao final do determinado prazo de empréstimo; este valor a mais que o banco cobra é conhecido como juros. Em outras palavras, juros é a remuneração pelo uso do capital.

$$M = P + J$$

P é o capital inicial ou principal, J é o juros e M é o montante ou capital final.

Taxa de Juros

No mercado financeiro, os juros são expressos como uma fração do capital inicial em uma determinada unidade de tempo:

$$J = i \times P$$

P é o capital inicial ou principal, J é o juros e i é a taxa de juros.

Exemplos de aplicação em python:

Um indivíduo faz um investimento de R$10.000,00 pelo prazo de cinco meses. No final do prazo, ele recebe R$10.800,00. Determine:

- a) Os Juros recebidos
- b) A taxa de juros do negócio

a)

```python
P = 10_000 # reais
M = 10_800 # reais
J = M - P
print(f' Juros recebidos: R$ {J} reais')
```

```
Juros recebidos: R$ 800 reais
```

```python
i = 100*(M/P - 1)
print(f' Taxa de juros: {round(i,2)} % a.p.')
```

```
Taxa de juros: 8.0 % a.p.
```

Regime de Capitalização

Regime de capitalização é o nome dado ao processo de formação de capital ao longo do tempo. O regime de capitalização se divide em dois tipos: discreto e contínuo. Nesse módulo, nós nos restringimos ao regime de capitalização discreta.

Regime de Capitalização Discreta

No regime de capitalização discreta, os juros são adicionados ao capital apenas no fim de determinado intervalo de tempo. O regime de capitalização discreta é comumente dividido em dois tipos, simples e composto.

Regime de Capitalização Discreta Simples

O regime de capitalização discreta simples é aquele no qual os juros incidem apenas sobre o capital inicial.

Seja M_0 o capital inicial de determinada aplicação, ao final do primeiro período de capitalização temos:

$$M_1 = M_0 + iM_0 = (1 + i)M_0$$

Ou seja, o montante ao final do primeiro período de capitalização M_1 é igual ao montante inicial M_0 acrescido de juros incidindo sobre o capital inicial iM_0. Ao final do segundo período temos:

$$M_2 = M_1 + iM_0 = (1 + 2i)M_0$$

Veja que os juros continuam incidindo sobre o montante inicial. Ao final do terceiro período de capitalização temos:

$$M_3 = M_2 + iM_0 = (1 + 3i)M_0$$

E, generalizando, ao final do n-ésimo período de capitalização, temos:

$$M_n = (1 + ni)M_0$$

Exemplos de aplicação em Python

```python
import handcalcs.render as render
from handcalcs import handcalc
```

```python
#montante simples
@handcalc(jupyter_display=True)
def montante_simples(P, i, N):
    MS = (1+N*i/100)*P
    return MS
```

Um indivíduo faz uma aplicação de R$20.000,00 por 5 anos numa instituição financeira que paga uma taxa de 7% a.a. (ao ano) no regime de capitalização simples. Qual é o montante da operação?

```python
P = 20_000 # reais
N = 5 # anos
i = 7 # % a.a.
MS = montante_simples(P, i, N)
print(f' O montante da operação é R$ {MS}')
```

[

$$\mathrm{MS} = \left(1 + N \cdot \frac{i}{100}\right) \cdot P = \left(1 + 5 \cdot \frac{7}{100}\right) \cdot 20000 \quad = 27000.000$$

]

```
O montante da operação é R$ 27000.0
```

Regime de Capitalização Discreta Composta

Já no regime de capitalização discreta composto, os juros incidem sobre o montante do período de capitalização anterior.

No primeiro período de capitalização, o montante é exatamente como no regime de capitalização discreta. Temos:

$$M_1 = M_0 + iM_0 = (1 + i)M_0$$

Já ao final do segundo período, os juros incidem sobre o montante anterior, gerando:

$$M_2 = M_1 + iM_1 = (1 + i)M_1 = (1 + i)^2 M_0$$

Veja como a fórmula muda; para o terceiro período temos:

$$M_3 = M_2 + iM_2 = (1 + i)M_2 = (1 + i)^3 M_0$$

E finalmente, generalizando, temos:

$$M_n = (1 + i)^n M_0$$

Exemplos de aplicação em Python

```python
# montante composto
@handcalc(jupyter_display=True)
def montante_composto(P, i, N):
    MC = P*(1 + i/100)**N
    return MC
```

Python

Um indivíduo toma um empréstimo de R$20.000,00 por um período de quatro meses. A taxa do empréstimo é de 0,65% a.m. (ao mês). Considerando o regime de juros compostos, determine o montante do empréstimo no final do período.

```python
P = 20_000 # reais
N = 4 # meses
i = 0.65 # % a.m.
MC = montante_composto(P, i, N)
print(f' O montante da operação é {MC}')
```

Python

[

$$MC = P \cdot \left(1 + \frac{i}{100}\right)^{N} = 20000 \cdot \left(1 + \frac{0.650}{100}\right)^{4} = 20525.092$$

]

```
O montante da operação é 20525.092005701248
```

Taxas Proporcionais

No regime de capitalização simples, taxas proporcionais são duas taxas em unidades de tempo distintas que, incidindo sobre um mesmo principal e pelo mesmo prazo, produzem o mesmo montante:

$$M = P(1 + n_1 i_1)$$

$$M = P(1 + n_2 i_2)$$

O que nos leva à seguinte relação entre as taxas:

$$n_1 i_1 = n_2 i_2$$

Exemplos de aplicação em Python

```python
#taxas proporcionais
@handcalc(jupyter_display=True)
def taxas_proporcionais(i1, n1,n2):
    i2 = n1*i1/n2
    return i2
```

Python

+ Code + Markdown

Qual é a taxa trimestral proporcional à taxa de 6% a.a. ?

```python
i1 = 6# % a.a.
n1 = 1 # ano
n2 = 4 # trimestres
i2 = taxas_proporcionais(i1, n1, n2)
print(f'A taxa trimestral proporcional à taxa de 6% a.a. é {i2} % a.t.')
```

Python

[

$$i2 = n1 \cdot \frac{i1}{n2} = 1 \cdot \frac{6}{4} = 1.500$$

]

```
A taxa trimestral proporcional à taxa de 6% a.a. é 1.5 % a.t.
```

Determinar a taxa mensal proporcional à taxa de 7,2% a.a.

```python
i1 = 7.2 # % a.a.
n1 = 1 # ano
n2 = 12 # meses
i2 = taxas_proporcionais(i1, n1, n2)
print(f'A taxa mensal proporcional à taxa de 7.2% a.a. é {i2} % a.m.')
```

Python

[

$$\mathrm{i2} = \mathrm{n1} \cdot \frac{\mathrm{i1}}{\mathrm{n2}} = 1 \cdot \frac{7.200}{12} = 0.600$$

]

```
A taxa mensal proporcional à taxa de 7.2% a.a. é 0.6 % a.m.
```

Determinar a taxa diária proporcional à taxa de 0,9% a.m.

```python
i1 = 0.9#% a.m.
n1 = 1 # mês
n2 = 30 # dias corridos (mês comercial)
i2 = taxas_proporcionais(i1, n1, n2)
print(f'A taxa diária proporcional à taxa de 0.9% a.m. é {round(i2, 2)} % a.d.c.')
```

Python

[

$$\mathrm{i2} = \mathrm{n1} \cdot \frac{\mathrm{i1}}{\mathrm{n2}} = 1 \cdot \frac{0.900}{30} = 0.030$$

]

```
A taxa diária proporcional à taxa de 0.9% a.m. é 0.03 % a.d.c.
```

Determinar a taxa anual proporcional à taxa de 0,0053% ao dia (a.d.).

```python
i1 = 0.0053 # % a.d.
n1 = 360 # dias corridos (mês comercial)
n2 = 1 # ano
i2 = taxas_proporcionais(i1, n1, n2)
print(f'A taxa anual proporcional à taxa de 0.0053% a.d. é {i2} % a.a.')
```

Python

[

$$\mathrm{i2} = \mathrm{n1} \cdot \frac{\mathrm{i1}}{\mathrm{n2}} = 360 \cdot \frac{0.005}{1} = 1.908$$

]

```
A taxa anual proporcional à taxa de 0.0053% a.d. é 1.908 % a.a.
```

Taxas Equivalentes

No regime de capitalização composta, taxas equivalentes são duas taxas em unidades de tempo distintas que incidindo sobre um mesmo principal e pelo mesmo prazo produzem o mesmo montante

$$M = P(1 + i_1)^{n_1}$$

$$M = P(1 + i_2)^{n_2}$$

O que nos leva à seguinte relação entre as taxas:

$$(1 + i_1)^{n_1} = (1 + i_2)^{n_2}$$

Exemplos de aplicação em Python

```python
#taxas equivalentes
@handcalc(jupyter_display=True)
def taxas_equivalentes(i1, n1, n2):
    i2 = (1 + i1/100)**(n1/n2) - 1
    return i2*100
```

Determinar a taxa trimestral equivalente à taxa de 6,5% a.a.

```python
i1 = 6.5 #% a.a.
n1 = 1 # ano
n2 = 4 # trimestres
i2 = taxas_equivalentes(i1, n1, n2)
print(f' A taxa trimestral equivalente à taxa de 6,5% a.a. é {round(i2,2)} % a.t.')
```

[

$$i2 = \left(1 + \frac{i1}{100}\right)^{\left(\frac{n1}{n2}\right)} - 1 = \left(1 + \frac{6.500}{100}\right)^{\left(\frac{1}{4}\right)} - 1 \quad = 0.016$$

]

```
A taxa trimestral equivalente à taxa de 6,5% a.a. é 1.59 % a.t.
```

Determinar a taxa anual equivalente à taxa de 0,8% a.m.

```python
i1 = 0.8 # % a.m.
n1 = 12 # meses
n2 = 1 # ano
i2 = taxas_equivalentes(i1, n1, n2)
print(f' A taxa anual equivalente à taxa de 0,8% a.m. é {round(i2,2)} % a.a.')
```

[

$$i2 = \left(1 + \frac{i1}{100}\right)^{\left(\frac{n1}{n2}\right)} - 1 = \left(1 + \frac{0.800}{100}\right)^{\left(\frac{12}{1}\right)} - 1 \quad = 0.100$$

]

```
A taxa anual equivalente à taxa de 0,8% a.m. é 10.03 % a.a.
```

Determinar a taxa diária equivalente à taxa de 0,65% a.m.

```python
i1 = 0.65 # % a.m.
n1 = 1 # mês
n2 = 30 # dias corridos (mês comercial)
i2 = taxas_equivalentes(i1, n1, n2)
print(f' A taxa diária equivalente à taxa de 0,65% a.m. é {round(i2,2)} % a.d.')
```

[

$$i2 = \left(1 + \frac{i1}{100}\right)^{\left(\frac{n1}{n2}\right)} - 1 = \left(1 + \frac{0.650}{100}\right)^{\left(\frac{1}{30}\right)} - 1 \quad = 0.000$$

]

```
A taxa diária equivalente à taxa de 0,65% a.m. é 0.02 % a.d.
```

Determine a taxa por dia útil equivalente à taxa de 5,3% a.m., considerando o mês comercial com 21 dias úteis.

```python
i1 = 5.3 # % a.m.
n1 = 1 # mês
n2 = 21 # dias úteis
i2 = taxas_equivalentes(i1, n1, n2)
print(f' A taxa diária equivalente à taxa de 0,65% a.m. é {i2} % a.d.u')
```

[

$$i2 = \left(1 + \frac{i1}{100}\right)^{\left(\frac{n1}{n2}\right)} - 1 = \left(1 + \frac{5.300}{100}\right)^{\left(\frac{1}{21}\right)} - 1 \quad = 0.002$$

]

```
A taxa diária equivalente à taxa de 0,65% a.m. é 0.2462227895126734 % a.d.u
```

Em um determinado investimento a taxa auferida foi de 18,7% ao período (a.p.), considerando o período de 67 dias úteis. Determine a taxa por dia útil equivalente.

```python
i1 = 18.7 # a.p.
n1 = 1 # período
n2 = 67 # dias úteis
i2 = taxas_equivalentes(i1, n1, n2)
print(f' A taxa diária equivalente à taxa de 18,7% a.p. é {i2} % a.d.u')
```

Python

[

$$i2 = \left(1 + \frac{i1}{100}\right)^{\left(\frac{n1}{n2}\right)} - 1 = \left(1 + \frac{18.700}{100}\right)^{\left(\frac{1}{67}\right)} - 1 \quad = 0.003$$

]

```
A taxa diária equivalente à taxa de 18,7% a.p. é 0.2561919638698784 % a.d.u
```

Dada a taxa de 26% a.a., determine a taxa equivalente no período de 92 dias corridos do ano comercial. O ano comercial possui 360 dias corridos.

```python
i1 = 26 # % a.a.
n1 = 92/360 #anos
n2 = 1 # período
i2 = taxas_equivalentes(i1, n1, n2)
print(f' A taxa ao período equivalente à taxa de 26,0% a.a. é {i2} % a.p.')
```

Python

[

$$i2 = \left(1 + \frac{i1}{100}\right)^{\left(\frac{n1}{n2}\right)} - 1 = \left(1 + \frac{26}{100}\right)^{\left(\frac{0.256}{1}\right)} - 1 \quad = 0.061$$

]

```
A taxa ao período equivalente à taxa de 26,0% a.a. é 6.08408880475646 % a.p.
```

Taxa Nominal

Quando a taxa é expressa em uma unidade de tempo que não é o mesmo período no qual os juros são capitalizados, essa taxa é chamada de taxa nominal.

Exemplo: Taxa de 6% a.a. com capitalização mensal ou ainda 2% a.a. com capitalização diária.

Taxa Efetiva

Quando a taxa é expressa em uma unidade de tempo, que é o mesmo período no qual os juros são capitalizados, essa taxa é chamada de taxa efetiva.

Exemplo: Taxa de 6% a.a. com capitalização anual ou ainda 2% a.m. com capitalização mensal.

Relação da Taxa Nominal com a Taxa Efetiva

Por convenção, dada uma taxa nominal i_N, a taxa efetiva i_E será aquela que lhe é proporcional.

$$n_1 i_N = n_2 i_E$$

Ou ainda:

$$i_N = \frac{n_2}{n_1} i_E$$

Exemplos de aplicação em Python

Dada a taxa nominal de 8% a.a., capitalizada mensalmente, determinar a taxa efetiva mensal e anual.

```python
i1 = 8# % a.a.
n1 = 1 # ano
n2 = 12 # meses
i2 = taxas_proporcionais(i1, n1, n2)
print(f' A taxa efetiva mensal é {i2} % a.m.')
```

[

$$i2 = n1 \cdot \frac{i1}{n2} = 1 \cdot \frac{8}{12} \quad = 0.66667$$

]

```
A taxa efetiva mensal é 0.6666666666666666 % a.m.
```

```python
i_em = i2
N1 = 12 # meses
N2 = 1 # ano
i_ea = taxas_equivalentes(i_em, N1, N2)
print(f' A taxa efetiva anual é {round(i_ea,2)} % a.a.')
```

[

$$i2 = \left(1 + \frac{i1}{100}\right)^{\left(\frac{n1}{n2}\right)} - 1 = \left(1 + \frac{0.66667}{100}\right)^{\left(\frac{12}{1}\right)} - 1 \quad = 0.08300$$

]

```
A taxa efetiva anual é 8.3 % a.a.
```

Dada a taxa nominal de 7,0% a.a., capitalizada trimestralmente, determine a taxa efetiva trimestral e anual.

```python
i_n = 7 # % a.a.
n1 = 1 # ano
n2 = 4 # trimestres
i_et = taxas_proporcionais(i_n, n1, n2)
print(f' A taxa efetiva trimestral é {i_et} % a.t.')
```

[

$$i2 = n1 \cdot \frac{i1}{n2} = 1 \cdot \frac{7}{4} \quad = 1.75000$$

]

```
A taxa efetiva trimestral é 1.75 % a.t.
```

```python
N1 = 4 # trimestres
N2 = 1 # ano
i_ea = taxas_equivalentes(i_et, N1, N2)
print(f' A taxa efetiva anual é {round(i_ea,2)} % a.a.')
```

[

$$i2 = \left(1 + \frac{i1}{100}\right)^{\left(\frac{n1}{n2}\right)} - 1 = \left(1 + \frac{1.75000}{100}\right)^{\left(\frac{4}{1}\right)} - 1 \quad = 0.07186$$

]

```
A taxa efetiva anual é 7.19 % a.a.
```

Dada a taxa nominal de 0,23% a.m., capitalizados anualmente, determine a taxa efetiva anual e mensal.

```python
i_n = 0.23 #% a.m.
n1 = 12 # meses
n2 = 1 # ano
i_ea = taxas_proporcionais(i_n, n1, n2)
print(f' A taxa efetiva anual é {round(i_ea,2)} % a.a.')
```

[

$$i2 = n1 \cdot \frac{i1}{n2} = 12 \cdot \frac{0.23000}{1} \quad = 2.76000$$

]

```
A taxa efetiva anual é 2.76 % a.a.
```

```python
N1 = 1 # ano
N2 = 12 # meses
i_em = taxas_equivalentes(i_ea, N1, N2)
print(f' A taxa efetiva mensal é {round(i_em,2)} % a.m.')
```

[

$$i2 = \left(1 + \frac{i1}{100}\right)^{\left(\frac{n1}{n2}\right)} - 1 = \left(1 + \frac{2.76000}{100}\right)^{\left(\frac{1}{12}\right)} - 1 \quad = 0.00227$$

]

```
A taxa efetiva mensal é 0.23 % a.m.
```

Dada a taxa nominal de 6% a.a., capitalizados mensalmente, determine a taxa efetiva anual.

```python
i_n = 6 # % a.a.
n1 = 1 # ano
n2 = 12 # meses
i_em = taxas_proporcionais(i_n, n1, n2)
print(f' A taxa efetiva mensal é {i_em} % a.m.')
```

[

$$i2 = n1 \cdot \frac{i1}{n2} = 1 \cdot \frac{6}{12} \quad = 0.50000$$

]

```
A taxa efetiva mensal é 0.5 % a.m.
```

```python
N1 = 12 # meses
N2 = 1 # ano
i_ea = taxas_equivalentes(i_em, N1, N2)
print(f' A taxa efetiva anual é {round(i_ea,2)} % a.a.')
```

[

$$i2 = \left(1 + \frac{i1}{100}\right)^{\left(\frac{n1}{n2}\right)} - 1 = \left(1 + \frac{0.50000}{100}\right)^{\left(\frac{12}{1}\right)} - 1 \quad = 0.06168$$

]

```
A taxa efetiva anual é 6.17 % a.a.
```

Taxas Variáveis

Vimos que, para o regime de capitalização composta, a relação entre o principal, o montante, a taxa efetiva e o tempo é expressa por:

$$M = P(1 + i)^n$$

Essa expressão é válida para o caso em que a taxa i seja constante para todos os períodos de capitalização. Para o caso mais geral, em que a taxa é variável nos períodos de capitalização, valem as expressões abaixo:

$$M_1 = (1 + i_1)M_0$$

Ao final do segundo período de capitalização temos:

$$M_2 = (1 + i_2)M_1 = (1 + i_2)(1 + i_1)M_0$$

Ao final do terceiro período:

$$M_3 = (1 + i_3)M_2 = (1 + i_3)(1 + i_2)(1 + i_1)M_0$$

E finalmente, generalizando, ao final do n-ésimo período de capitalização, temos:

$$M_n = (1 + i_1)(1 + i_2) \dots (1 + i_n)M_0$$

Exemplos de aplicação em Python

```python
def montante(list_rent, P):
    M = P
    for i in list_rent:
        M *= (1 + i/100)
    return M
```

Um investidor obtem as seguintes rentabilidades efetivas mensais Mês 1 : 0.8% Mês 2 : 0.3% Mês 3 : 0.2% ao investir R$100.00,00 no mercado financeiro por três meses. Qual é o montante do resgate?

```python
list_rent = [0.8, 0.3, 0.2]
P = 100_00
M = montante(list_rent, P)
print(f' O montante é dado por ${round(M,2)}')
```

```
O montante é dado por $10130.46
```

Por um período de quatro dias úteis é realizado uma operação interbancária com um principal de R$ 100.000,00 é realizada por quatro dias úteis. As taxas over mês da operação são as seguintes:

1º dia: 0,748% a.m.o.

2º dia: 0,742% a.m.o.

3º dia: 0,756% a.m.o.

4º dia: 0,753% a.m.o.

Determine o montante da operação

```python
P = 100_00 # reais
list_rent = [0.748/30, 0.742/30, 0.756/30, 0.753/30]
M = montante(list_rent, P)
print(f' O montante é dado por ${round(M,2)}')
```

```
O montante é dado por $10010.0
```

Taxa Acumulada

Vimos no tópico anterior que o montante ao final do n-ésimo período de capitalização é dado por:

$$M_n = (1 + i_1)(1 + i_2)\ldots(1 + i_n)M_0$$

Podemos também pensar em termos da taxa efetiva no período, também conhecida como taxa acumulada:

$$M_n = (1 + i_{ac})M_0$$

Dessa forma, igualando as duas equações anteriores, temos

$$i_{ac} = (1 + i_1)(1 + i_2)\ldots(1 + i_n) - 1$$

Exemplos de aplicação em Python

```python
def taxa_ac(list_rent):
    import numpy as np
    list_fat = [1 + i/100 for i in list_rent]
    i_ac = (np.prod(list_fat) - 1)
    return 100*i_ac
```

Python

Um indivíduo, deixou aplicado R$ 20.000,00 por um período de 5 meses na bolsa de valores e obteve as seguintes rentabilidades efetivas mensais:

Mês 1: 1,5%

Mês 2: 6,2%

Mês 3: 2,7%

Mês 4: -5,2%

Mês 5: -3,8%

Determinar o montante do investimento e a taxa acumulada no período.

```python
P = 20_000 # reais
list_rent = [1.5, 6.2, 2.7, -5.2, -3.8]
M = montante(list_rent, P)
print( f' O montante acumulado é {M} reais')
i_ac  = taxa_ac(list_rent)
print(f' A taxa acumulada é {round(i_ac,2)} % a.p.')
```

[38] Python

```
O montante acumulado é 20191.770790027193 reais
A taxa acumulada é 0.96 % a.p.
```

```python
# Outra forma
i_ac = 100*(M/P - 1)
print(f' A taxa acumulada é {round(i_ac,2)} % a.p.')
```

[39] Python

```
A taxa acumulada é 0.96 % a.p.
```

Durante três dias úteis, uma operação interbancária foi realizada com as seguintes taxas over mês

1º dia: 0,732% a.m.o.

2º dia: 0,756% a.m.o.

3º dia: 0,748% a.m.o.

Determine a taxa efetiva no período da operação

```python
list_rent = [0.732/30, 0.756/30, 0.748/30]
i_ac = taxa_ac(list_rent)
print(f' A taxa acumulada é {round(i_ac,5)} % a.p.')
```

[40] Python

```
A taxa acumulada é 0.07455 % a.p.
```

Taxa Média

No regime de capitalização composta, com taxas flutuantes, vale a expressão abaixo:

$$M_n = (1 + i_1)(1 + i_2)\dots(1 + i_n)M_0$$

Admitindo que haja uma taxa efetiva constante em todos os períodos unitários de capitalização que, incidindo sobre o mesmo principal e mesmo prazo produz o mesmo montante, temos

$$M_n = (1 + \underline{i})^n M_0$$

Dessa forma, igualando as duas expressões anteriores, temos:

$$(1 + \underline{i})^n = (1 + i_1)(1 + i_2)\dots(1 + i_n)$$

Ou ainda:

$$1 + \underline{i} = [(1 + i_1)(1 + i_2)\dots(1 + i_n)]^{1/n} = \sqrt[n]{(1 + i_1)(1 + i_2)\dots(1 + i_n)}$$

A taxa efetiva $\underline{i}$ é denominada taxa média geométrica.

Exemplos de aplicação em Python

```python
def taxa_media(list_rent):
    import numpy as np
    list_fat = [1 + x/100 for x in list_rent]
    i_m = ((np.prod(list_fat))**(1/len(list_fat)) - 1)
    return 100*i_m
```

Python

Um individuo investiu durante 4 meses e obteve as seguintes rentabilidades efetivas:

Mês 1: 0,5%

Mês 2: 3,2%

Mês 3: -2,5%

Mês 4: 5,7%

Determine a rentabilidade mensal média

```python
list_rent = [0.5, 3.2, -2.5, 5.7]
i_med = taxa_media(list_rent)
print(f'A rentabilidade mensal média nesse período foi de {round(i_med,2)} % a.m.')
```

[13] Python

```
A rentabilidade mensal média nesse período foi de 1.68 % a.m.
```

Durante 5 dias úteis uma operação interbancária é realizada com as seguintes taxas over:

1º dia: 0,626% a.m.o.

2º dia: 0,632% a.m.o.

3º dia: 0,648% a.m.o.

4º dia: 0,637% a.m.o.

5º dia: 0,649% a.m.o.

Determine a taxa over média do período.

```python
list_rent = [0.626/30, 0.632/30, 0.648/30, 0.637/30, 0.649/30]
i_med = taxa_media(list_rent)
print(f'A taxa efetiva diária útil média é {round(i_med,5)} % a.d.u')

i_med_mo = i_med*30
print(f'A taxa over mês média é {round(i_med_mo,5)} % a.m.o.')
```

[15] Python

```
A taxa efetiva diária útil média é 0.02128 % a.d.u
A taxa over mês média é 0.6384 % a.m.o.
```

Taxa Real

A fórmula de Fisher estabelece o efeito da inflação sobre as taxas de juros e é expressa por meio da relação:

$$1 + i = (1 + \theta)(1 + r)$$

$$i = taxa\ efetiva$$

$$\theta = taxa\ de\ inflação\ obtida\ por\ meio\ de\ um\ índice\ de\ preços$$

$$r = taxa\ real$$

Exemplos de aplicação em Python

```python
def taxas(i=0, theta=0, r=0):
    if i == 0:
        return 100*((1 + theta/100)*(1 + r/100) - 1)
    elif theta == 0:
        return 100*((1 + i/100)/(1 + r/100) - 1)
    elif r == 0:
        return 100*((1 + i/100)/(1 + theta/100) - 1)
```

Um indivíduo fez uma aplicação em um mês cuja a rentabilidade foi de 1,5 % a.m. Sabendo que a inflação no período foi de 0,5% a.m. Determine sua rentabilidade real:

```python
i = 1.5 # % a.m.
theta = 0.5 #% a.m.
r = taxas(i, theta, 0)
print(f'A rentabilidade real foi de {round(r,3)} % a.m.')
```

```
A rentabilidade real foi de 0.995 % a.m.
```

```python
theta = 0.5 #% a.m.
r = 0.995 # % a.m.
i = taxas(0, theta, r)
print(f'A taxa deve ser de  {round(i,2)} % a.m. ')
```

```
A taxa deve ser de  1.5 % a.m.
```

```python
i = 1.5 # % a.m.
r = 0.995 # % a.m.
theta = taxas(i, 0, r)
print(f'A inflação foi de  {round(theta,2)} % a.m. ')
```

```
A inflação foi de  0.5 % a.m.
```

Um indivíduo deixou a quantia de $100.000,00 aplicada por 4 meses e resgatou o montante de $106.000,00. As taxas de inflação mensal do período foram as seguintes:

Janeiro: 0,6%

Fevereiro: 0,8%

Março: 0,4%

Abril: 0,32%

Determine:

a) a taxa efetiva obtida pelo indivíduo no período da aplicação

```python
P = 100_000 # reais
M = 106_000 # reais
i_efp = 100*(M/P - 1)
print(f' A taxa efetiva ao período é {round(i_efp,2)} % a.p.')
```

```
A taxa efetiva ao período é 6.0 % a.p.
```

b) A taxa de inflação acumulada no período da aplicação

```python
def taxa_ac(list_rent):
    import numpy as np
    list_fat = [1 + x/100 for x in list_rent]
    i_ac = (np.prod(list_fat) - 1)
    return 100*i_ac
```

```python
list_rent = [0.6, 0.8, 0.4, 0.32]
theta_ac = taxa_ac(list_rent)
print(f'A inflação acumulada no período vale {round(theta_ac,2)} % a.p.')
```

```
A inflação acumulada no período vale 2.14 % a.p.
```

c) A taxa real de retorno do indivíduo no período da aplicação

```python
i_realp = taxas(i_efp, theta_ac, 0)
print(f' A taxa real no período é {round(i_realp,2)} % a.p.')
```

```
A taxa real no período é 3.78 % a.p.
```

> Capítulo 2

Capítulo 2. Produtos de Renda Fixa

Em geral, os títulos de renda fixa podem apresentar três tipos de rentabilidade: prefixada, pós-fixada e híbrida. Na rentabilidade prefixada, o investidor tem pleno conhecimento de quanto irá receber no dia do resgate. Na rentabilidade pós-fixada, o investidor tem conhecimento de qual indexador será usado para definir o valor que ele irá ganhar, porém, com apenas essa informação, não é possível saber de antemão o quanto irá receber no dia do resgate. Os indicadores mais comuns são o certificado de depósito interbancário (CDI), a taxa Selic e o Índice de Preços para o Consumidor Amplo(IPCA). Por último, a rentabilidade híbrida possui uma parte prefixada e outra pós-fixada.

Poupança

A modalidade de investimento mais conhecida e mais utilizada pelos investidores no Brasil é a caderneta de poupança. Isso não significa que seja a mais rentável, muito pelo contrário, mas ela tem certas comodidades como: facilidade de migrar o dinheiro da conta corrente para a poupança, a não cobrança de imposto de renda sobre os rendimentos da caderneta e a liquidez imediata.

Atualmente, o rendimento da caderneta de poupança segue a seguinte regra: se a Selic estiver acima de 8,5%, a poupança rende 0,5% a.m. mais a taxa referencial (TR); caso a Selic esteja menor ou igual a 8,5%, a poupança rende 70% da meta Selic ao ano mais a taxa referencial (TR). Esquematicamente, temos:

$$Selic\ > 8{,}5\%:\ TR + 0{,}5\%\ a.m.$$

$$Selic\ \leq 8{,}5\%: TR + 70\%\ da\ meta\ selic\ ao\ ano$$

Exemplo de código em Python para calcular montante e rentabilidade acumulada

```python
def poup(P, list_tr, list_selic):
    import numpy as np
    list_rent = []
    for tr, selic in zip(list_tr, list_selic):
        if selic > 8.5:
            i = (1 +tr/100)*(1 + 0.5/100) - 1 # a.m.
        else:
            i = (1 + tr/100)*(1 + 0.7*selic/100)**(1/12) - 1 # a.m.
        list_rent.append(1 + i)
    M = P*np.prod(list_rent)
    rent_acum_per = 100*(M/P - 1)
    return M, rent_acum_per
```

Python

Exemplo de aplicação

Um indivíduo deixa uma quantia de R$ 10.000,00 investida por 4 meses na poupança. As taxas referenciais no período foram:

- Mês 1 : 0,022%
- Mês 2 : 0,0194%
- Mês 3 : 0,0083%
- Mês 4 : 0,00 %

E as metas Selic do período foram:

- Mês 1 : 9,5% a.a.
- Mês 2 : 8,5% a.a.
- Mês 3 : 8,5% a.a.
- Mês 4 : 8,0% a.a.

Determine o montante e a rentabilidade efetiva calculada no período.

```python
P = 10_000 # reais
list_tr = [0.022, 0.0194, 0.0083, 0.00]
list_selic = [9.5, 8.5, 8.5, 8.0]
M, i_efp = poup(P, list_tr, list_selic)
print(f'O montante é dado por {M} reais')
print(f'A rentabilidade é dada por {i_efp} % a.p.')
```

[56] Python

```
O montante é dado por 10198.525183095144 reais
A rentabilidade é dada por 1.98525183095144 % a.p.
```

CDB

Quando um investidor empresta dinheiro ao banco, o banco empresta o dinheiro recebido para outras pessoas ou instituições por uma taxa maior do que aquela que ele paga ao investidor. Uma das formas através da qual o banco capta dinheiro dos investidores é a que vimos na seção anterior, através da caderneta de poupança. Uma outra forma é através dos Certificados de Depósito Bancário (CDBs), que são os títulos de créditos emitidos por bancos. São comuns CDBs prefixados e pós-fixados. O investidor que investe em CDB conta com a garantia do Fundo Garantidor de Crédito (FGC) e usualmente com liquidez diária. Diferentemente da caderneta de poupança, os investidores que investem nessa modalidade estão sujeitos à cobrança de IR e IOF.

Exemplos de aplicação em Python

Um investidor aplica $ 200.000,00 em um CDB prefixado, à taxa de 24,80% a.a. (base 360 dias), por um período de 32 dias corridos, no qual estão contidos 21 dias úteis. O imposto de renda é retido na fonte à alíquota de 22,5% sobre o rendimento bruto.

a) O montante bruto

```python
P = 200_000 # reais
i_ano = 24.80 #% a.a. (base 360 dias)
nc = 32/360 # anos
M = P*(1 + i_ano/100)**nc
print(f' O montante Bruto é {M} reais')
```

```
O montante Bruto é 203977.5650936861 reais
```

b) O rendimento bruto

```python
rendimento_bruto = M - P
print(f' O rendimento bruto é R$ {rendimento_bruto}')
```

```
O rendimento bruto é R$ 3977.5650936861057
```

c) O imposto de renda retido na fonte

```python
imposto = (22.5/100)*rendimento_bruto
print(f' O imposto é R${imposto}')
```

```
O imposto é R$894.9521460793738
```

d) O Montante líquido

```python
ML = M - imposto
print(f' O montante liquido é R${ML}')
```

```
O montante liquido é R$203082.61294760674
```

e) A taxa efetiva liquida do período

```python
taxa_efetiva_liquida = 100*(ML/P - 1)
print(f' A taxa efetiva líquida ao período é {taxa_efetiva_liquida} % a.p.')
```

```
A taxa efetiva líquida ao período é 1.5413064738033722 % a.p.
```

f) A taxa over ano equivalente

```python
taxa_efetiva_bruta = 100*(M/P - 1)
n1 = 1 # período
n2 = 21/252 # anos
taxa_over_equivalente = taxas_equivalentes(taxa_efetiva_bruta, n1, n2)
print(f' A taxa over ano equivalente é {taxa_over_equivalente} % a.a.o.')
```

[

$$i2 = \left(1 + \frac{i1}{100}\right)^{\left(\frac{n1}{n2}\right)} - 1 = \left(1 + \frac{1.989}{100}\right)^{\left(\frac{1}{0.083}\right)} - 1 \quad = 0.267$$

]

```
A taxa over ano equivalente é 26.656910738235396 % a.a.o.
```

g) A taxa over ano líquida equivalente

```python
N1 = 1 # período
N2 = 21/252 # anos
taxa_over_liquida_equivalente = taxas_equivalentes(taxa_efetiva_liquida, N1, N2)
print(f' A taxa over ano equivalente é {taxa_over_liquida_equivalente} % a.a.o.')
```

[

$$i2 = \left(1 + \frac{i1}{100}\right)^{\left(\frac{n1}{n2}\right)} - 1 = \left(1 + \frac{1.541}{100}\right)^{\left(\frac{1}{0.083}\right)} - 1 \quad = 0.201$$

]

```
A taxa over ano equivalente é 20.147008818436607 % a.a.o.
```

LCA

Uma outra forma bem conhecida do banco captar dinheiro é através das Letras de Crédito Agrícolas (LCAs). Em outras palavras, o investidor empresta dinheiro ao banco e este por sua vez empresta para empresas que atuam no agronegócio. Nessa modalidade, o investidor é isento da cobrança de IR e IOF, conta com a garantia do FGC, mas perde na questão da liquidez, visto que, em geral, as LCAs só podem ser resgatadas no vencimento.

Exemplos de aplicação em Python

Com o objetivo de financiar o agronegócio, uma instituição financeira emite LCA. Um investidor (pessoa física) aplica R $ 580.000,00 nesse título que tem como características:

- Isenção de impostos
- Rentabilidade: 7,49% a.a.
- Prazo: 15 meses Determine, do ponto de vista do investidor:

a) O montante a receber no final do período e a rentabilidade efetiva

```python
taxa = 7.49/100 # a.a.
P = 580_000 # reais
M = P*(1 + taxa)**(15/12)
print(f'O montante a receber é de R${M}')
rentabilidade_efetiva = 100*(M/P - 1)
print(f' A rentabilidade efetiva é {rentabilidade_efetiva} % a.p.')
```

```
O montante a receber é de R$634801.6869098026
 A rentabilidade efetiva é 9.448566708586647 % a.p.
```

b) Comparação com uma aplicação de CDB que tenha a mesma rentabilidade (imposto 17.5%)

```python
rendimento_bruto = M - P
imposto = (17.5/100)*rendimento_bruto
ML_cdb = M - imposto
print(f'O montante liquido do cdb é de R${ML_cdb}')
rentabilidade_cdb = 100*(ML_cdb/P - 1)
print(f' A rentabilidade efetiva do cdb é {rentabilidade_cdb} % a.p.')
```

```
O montante liquido do cdb é de R$625211.3917005871
 A rentabilidade efetiva do cdb é 7.795067534583988 % a.p.
```

c) Se a rentabilidade fosse pós-fixada, atrelada a 90% do CDI, qual seria o montante após três meses? CDI Acumulado

Mês 1 : 0.5866 % a.m.

Mês 2 : 0.4815 % a.m.

Mês 3 : 0.5377 % a.m.

```python
M = P*(1 + 0.9*0.5866/100)*(1 + 0.9*0.4815/100)*(1 + 0.9*0.5377/100)
print(f'O montante após três meses é de R$ {M}')
```

```
O montante após três meses é de R$ 588422.5910470034
```

LCI

Muito similares ao LCA são as Letras de Crédito Imobiliário (LCIs). Nesta modalidade, o investidor empresta dinheiro ao banco e este, por sua vez, faz financiamentos imobiliários. Neste caso, o investidor também é isento de IR e IOF, conta com a garantia do FGC e tem problemas com a liquidez, visto que as LCIs só podem ser resgatadas no vencimento.

Exemplos de aplicação em Python

Um empresário compra uma LCI em sua emissão pelo valor de R$ 50.000,00. Pelo prazo de um ano, o papel apresenta remuneração pós-fixada indexada a 90% do CDI. O CDI do período é de 10%. Determine:

a) Taxa efetiva do período

```python
cdi_p = 10/100
taxa_efetiva_p = (90/100)*cdi_p
print(f' A taxa efetiva no período é {round(100*taxa_efetiva_p,2)} % a.p.')
```

```
A taxa efetiva no período é 9.0 % a.p.
```

b) Montante no vencimento

```python
P = 50_000 # reais
M = P*(1 + taxa_efetiva_p)
print(f'O montante é R${M}')
```

```
O montante é R$54500.00000000001
```

Considerando que a alternativa do investidor era aplicar em um CDB, a 103% do CDI de prazo igual, pergunta-se: qual a melhor alternativa de investimento? Nota: Considere 17.5% de Imposto de renda no período

```python
taxa_efetiva_p_cdb = (103/100)*cdi_p
M_cdb = P*(1 + taxa_efetiva_p_cdb)
print(f'O montante bruto é R${M_cdb}')
```

```
O montante bruto é R$55150.0
```

```python
rendimento_bruto = M_cdb - P
imposto = (17.5/100)*rendimento_bruto
ML_cdb = M_cdb - imposto
print(f'O montante líquido é {ML_cdb}')
```

Python

```
O montante líquido é 54248.75
```

CRA

De forma semelhante ao que é feito na LCA, também é possível o investidor emprestar dinheiro para empresas que atuam no agronegócio através dos Certificados de Recebíveis Agrícolas emitidos por securitizadoras. Nessa modalidade de investimento, o investidor também é isento da cobrança de IR e IOF, tem problemas com liquidez e não conta com a proteção do FGC.

CRI

De forma semelhante ao que é feito no LCI, também é possível o investidor emprestar dinheiro para construtoras realizarem novos projetos imobiliários através dos Certificados de Recebíveis Imobiliários que são emitidos por securitizadoras. Nessa modalidade de investimento, o investidor também é isento da cobrança de IR e IOF, tem problemas com liquidez e não conta com a proteção do FGC.

Exemplos de aplicação em Python

Um investidor comprou cinco CRIs emitidas por uma securitizadora. O valor da emissão foi de R $ 270.000.000,00, composta por 900 CRIs, pelo prazo de dez anos, com valores não atualizados. O título é prefixado e rende 9,5% ao ano, com pagamento semestral de juros e o principal na data de vencimento. Pede-se (base 360 dias):

a) Quanto foi o valor investido em CRIs

```python
qtd = 5 # CRIs
valor_unit = 270_000_000/900
P = valor_unit*qtd
print(f'O valor investido em CRIs foi de R$ {P}')
```

Python

```
O valor investido em CRIs foi de R$ 1500000.0
```

b) Juros do primeiro semestre

```python
taxa = 9.5 #% a.a.
M1 = P*(1 + taxa/100)**(180/360)
J1 = M1 - P
print(f' Juros do primeiro semestre: R$ {J1}')
```

Python

```
 Juros do primeiro semestre: R$ 69633.71523422608
```

c) Taxa efetiva desse período

```python
taxa_efetiva =100*(M1/P - 1)
print(f'A taxa efetiva ao período é {taxa_efetiva}% a.p.')
```

Python

```
A taxa efetiva ao período é 4.642247682281742% a.p.
```

d) O montante líquido ao final do período

```python
montante_liquido = P + J1 # = M1
print(f'O montante líquido ao final do primeiro semestre é de R$ {montante_liquido}')
```

Python

```
O montante líquido ao final do primeiro semestre é de R$ 1569633.715234226
```

> Capítulo 3

Capítulo 3. Formação de Preços dos Títulos Públicos

Assim como o investidor pode emprestar dinheiro para bancos e securitizadoras através de CDBs, LCIs, LCAs, CRIs e CRAs, também é possível emprestar dinheiro para o governo através dos títulos públicos e receber uma remuneração por este empréstimo. Dessa forma, é possível para o governo utilizar esse dinheiro para gastos públicos em educação, saúde etc.

Os títulos negociados são basicamente de três tipos: Tesouro Prefixado, Tesouro Selic e Tesouro IPCA.

Títulos Prefixados

Tesouro Prefixado (LTN) e Tesouro Prefixado com Juros Semestrais (NTN-F) são os dois títulos prefixados ofertados pelo Tesouro Direto. Neste capítulo, vamos tratar apenas do LTN.

Tesouro Prefixado (LTN)

O Tesouro Prefixado, também conhecido como Letra do Tesouro Nacional (LTN), é um título prefixado e isso significa que o investidor sabe exatamente quanto irá receber se permanecer com o título até a data de vencimento. O investidor compra um título por um determinado preço e na data de vencimento recebe o valor investido somado à rentabilidade contratada; este valor final é conhecido como Valor de Face ou Nominal e é fixado em R$ 1.000,00. Este é o valor de face para um título, mas o investidor pode comprar, por exemplo, até 1% de um título. Dito isso, o preço do título na data da compra tem que ser um valor menor do que R$ 1.000,00. Como é formado o valor presente do título? Como vimos no capítulo 1, para o regime de capitalização composta devemos ter:

$$VF = VP(1 + \frac{i}{100})^{(\frac{du}{252})},$$

VP é o valor presente, VF é o valor de face (no caso, para um título, esse valor é R$ 1.000), i é a taxa contratada e du é o número de dias úteis entre a data de compra do título e o vencimento do título. Dessa forma, temos que o valor presente do título é dado por:

$$VP = \frac{VF}{(1+\frac{i}{100})^{(\frac{du}{252})}}.$$

É possível fazer a recompra do título antes da data de vencimento, porém, devido a condições de mercado, a sua rentabilidade pode ser maior ou menor do que a contratada. Veja que existe uma relação inversa entre a taxa i e o valor presente VP do título, isso significa que se a taxa aumenta em relação à taxa contratada, o preço diminui.

[Para uma visão mais detalhada de cálculo de valor presente e rentabilidade desse título, clique aqui.]()

Exemplos de aplicação em Python

```python
import handcalcs.render as render
from handcalcs import handcalc
```

Tesouro Prefixado (LTN)

$$VF = VP\left(1+\frac{i}{100}\right)^{\frac{du}{252}}$$

```python
@handcalc(jupyter_display=True)
def vp(taxa, du):
    vf = 1_000 # reais
    vp = vf/((1 + taxa/100)**(du/252))
    return vp
```

Exemplo 1:

Exemplo 1

Um investidor comprou uma LTN, com 250 dias úteis entre a data de compra e a data de vencimento) com uma taxa de 13 % a.a. Qual é o preço desse título na data da compra?

```python
taxa = 13# % a.a.
du = 250 # dias úteis
vp = vp(taxa, du)
print(f' O preço unitário para um título pré com taxa bruta de {taxa}% a.a. e {du} dias úteis é R$ {vp}')
```

[

$$\mathrm{vf} = 1000 \ \text{(reais)}$$

$$\mathrm{vp} = \frac{\mathrm{vf}}{(1+\frac{\mathrm{taxa}}{100})^{(\frac{\mathrm{du}}{252})}} = \frac{1000}{(1+\frac{13}{100})^{(\frac{250}{252})}} = 885.815$$

]

```
O preço unitário para um título pré com taxa bruta de 13% a.a. e 250 dias úteis é R$ 885.8145591115162
```

Para o cálculo da taxa, podemos usar:

$$i = 100\left[\left(\frac{VF}{VP}\right)^{252/du} - 1\right]$$

```python
@handcalc(jupyter_display=True)
def taxa(vp, du):
    vf = 1_000 # reais
    taxa = 100*((vf/vp)**(252/du) - 1)
    return taxa
```

Exemplo 2:

Exemplo 2

Um investidor comprou uma LTN, com 430 dias úteis entre a data de compra e a data de vencimento) com um preço unitário de R$ 800? Qual é a taxa desse título na data da compra?

```python
du = 430 # dias úteis
vp = 800 # reais
i = taxa(vp, du)

print(f'A taxa desse título na compra é de {i} % a.a.')
```

[

$$\mathrm{vf} = 1000 \ \ (\mathrm{reais})$$

$$\mathrm{taxa} = 100 \cdot \left(\left(\frac{\mathrm{vf}}{\mathrm{vp}}\right)^{\left(\frac{252}{\mathrm{du}}\right)} - 1\right) = 100 \cdot \left(\left(\frac{1000}{800}\right)^{\left(\frac{252}{430}\right)} - 1\right) = 13.971$$

]

```
A taxa desse título na compra é de 13.97084679573295 % a.a.
```

Tesouro Prefixado com Juros Semestrais (NTN-F)

O Tesouro Prefixado com juros semestrais, também conhecido como Notas do Tesouro Nacional Série F (NTN-F), é um título prefixado como a LTN, de tal forma que o investidor sabe exatamente quanto vai receber caso fique com o título até o vencimento. O diferencial em relação à LTN é que a NTN-F realiza pagamentos de juros semestrais, conhecidos como cupons. Os cupons são calculados sobre o valor de face (R$ 1.000,00) com uma taxa de 10% a.a. sobre o valor de face, pagos semestralmente, o que equivale a aproximadamente 4,88% a.s.

Esquematicamente, temos:

$$V_p = \frac{cp}{(1+\frac{i}{100})^{(\frac{du_1}{252})}} + \frac{cp}{(1+\frac{i}{100})^{(\frac{du_2}{252})}} + \ldots + \frac{cp + VF}{(1+\frac{i}{100})^{(\frac{du_N}{252})}}$$

$$cp = VF \times [(1 + tx_{cp}/100)^{1/2} - 1]$$

$$tx_{cp} = 10\%\ a.a.$$

$$VF \ = \ R\$\ 1.000{,}00,$$

No caso, cp é o cupom. [Um cálculo detalhado de valor presente e rentabilidade desse título pode ser encontrado aqui.]

Exemplo de código em Python para cálculo do valor presente

```python
def vp(taxa, list_du):
    vf = 1_000 # reais
    tx_c = 10 # % a.a.
    list_num = len(list_du)*[vf*((1 + tx_c/100)**(1/2) - 1)]
    list_num[-1] += vf
    list_vp = []
    for du, num in zip(list_du, list_num):
        temp_vp = num/(1 + taxa/100)**(du/252)
        list_vp.append(temp_vp)
    vp = np.sum(list_vp)
    return vp
```

Exemplo 1:

Qual é o valor presente de uma NTN-F com taxa de 11% a.a. e pagamentos de cupom com intervalos de 120, 240, 360 e 480 dias úteis em relação à hoje?

```python
tx = 11
list_du = [120, 240, 360, 480]
v_p = vp(tx, list_du)
print(f' O valor presente é  de R$ {v_p}')
```

```
O valor presente é  de R$ 992.4216114430419
```

Exemplo 2:

Qual é o valor presente de uma NTN-F com taxa de 15% a.a. e pagamentos de cupom com intervalos de 122, 250, 374, 501, 625, 750, 874 e 1000 dias úteis em relação à hoje?

```python
#Exemplo2 NTN-F Apostila Tesouro Direto
tx = 15.00
list_du = [122, 250, 374, 501, 625, 750, 874, 1000]
v_p = vp(tx, list_du)
print(f' O valor presente é de R$ {v_p}')
```

```
O valor presente é de R$ 863.8354466031728
```

Também é possível vender o título antes do vencimento, mas, nesse caso, a rentabilidade vai variar de acordo com as condições de mercado assim como vimos com a LTN.

Tesouro Selic (LFT)

O Tesouro Selic, também conhecido como Letra Financeira do Tesouro (LFT), é um título no qual a sua rentabilidade está atrelada a um indexador, que é a taxa Selic. A Selic é determinada pelo Comitê de Política Monetária e, portanto, pode variar ao longo do tempo em que o investidor permanece com o título; sendo assim, o investidor não tem visibilidade de qual será a sua rentabilidade no vencimento do título, ou seja, esse é um título pós-fixado. De todos os títulos ofertados pelo Tesouro Direto, é o que possui o menor risco e ele não possui pagamentos de cupom.

O valor presente de uma LFT depende do seu Valor Nominal Atualizado (VNA). O VNA foi definido como tendo o valor de R$1.000,00 no data-base (1/7/2000) e vem sendo corrigido diariamente, conforme a evolução da taxa Selic. Para o cálculo do valor presente, podemos utilizar, esquematicamente:

$$VNA_t = VNA_{t-1} \times (1 + Selic_{t-1})^{(1/252)}$$

$$VP_t = VNA_t \times Cot$$

$$Cot = \frac{100}{(1+tx)^{(du/252)}}$$

Um cálculo detalhado de cada uma dessas quantidades pode ser encontrado aqui.

Exemplos de aplicação em Phyton

```python
import numpy as np

# Sobre as taxas https://www.tesourodireto.com.br/conheca/regras.htm
class titulo_ntnb(object):
    def __init__(self):
        self.vf = 1_000 # valor de face, o valor que será recebido no vencimento por cada título comprado
        self.tx_c = 10

    def get_pu(self, vna, selic_p, du, tx):
        vna_p = vna*(1 + selic_p/100)**(1/252)
        cot = 1/(1 + tx/100)**(du/252)
        pu = vna_p*cot
        return pu

    def venda_antecipada(self, vnapc, vnapv, duc, duv, txc, txv):
        cotc = 1/(1 + txc/100)**(duc/252)
        puc = vnapc*cotc
        print('pu_compra', puc)
        cotv = 1/(1 + txv/100)**(duv/252)
        puv = vnapv*cotv
        print('pu_venda', puv)
        rb = puv/puc - 1
        rba = (1 + rb)**(252/(duc - duv)) - 1

        return rb*100, rba*100
```
Python

Tesouro IPCA+ (NTN-B Principal)

O Tesouro IPCA+, também conhecido como Notas do Tesouro Nacional Série B Principal (NTN-B Principal), é um título pós-fixado com a sua rentabilidade atrelada ao indexador IPCA (Índice Nacional de Preços ao Consumidor Amplo) mais os juros que são definidos no momento da aplicação. Para essa modalidade, não há pagamentos de cupons intermediários, ou seja, ocorre apenas um pagamento, que é o valor investido acrescido da rentabilidade na data de vencimento.

$$VNA_p = VNA \times (1 + IPCA_p)^{pr}$$

$$V_p = VNA_p \times Cot$$

$$Cot = \frac{100}{(1 + tx)^{(du/252)}}$$

[Uma amostra detalhada dessas quantidades pode ser encontrada aqui.]

Exemplos de aplicação em Python

```python
import numpy as np

# Sobre as taxas https://www.tesourodireto.com.br/conheca/regras.htm
class titulo_ntnb(object):
    def __init__(self):
        self.vf = 1_000 # valor de face, o valor que será recebido no vencimento por cada título comprado
        self.tx_c = 10

    def get_pu(self, vna, ipca_p, du, dc1, dc2, tx):
        vna_p = vna*(1 + ipca_p/100)**(dc1/dc2)
        cot = 1/(1 + tx/100)**(du/252)
        pu = vna_p*cot
        return pu

    def venda_antecipada(self, vnapc, vnapv, duc, duv, txc, txv):
        cotc = 1/(1 + txc/100)**(duc/252)
        puc = vnapc*cotc
        cotv = 1/(1 + txv/100)**(duv/252)
        puv = vnapv*cotv
        rb = puv/puc - 1
        rba = (1 + rb)**(252/(duc - duv)) - 1

        return rb*100, rba*100
```

Tesouro IPCA+ com Juros Semestrais (NTN-B)

O Tesouro IPCA+ com Juros Semestrais, também conhecido como Notas do Tesouro Nacional Série B (NTN-B), é um título pós-fixado com a sua rentabilidade atrelada ao indexador IPCA (Índice Nacional de Preços ao Consumidor Amplo) mais os juros que são definidos no momento da aplicação. A diferença do NTN-B para o NTN-B Principal é que a primeira paga cupons semestrais, de forma muito parecida como vimos no capítulo 4 com a NTN-F.

Uma explicação detalhada sobre esse título pode ser encontrada aqui.

Exemplos de aplicação em Python

```python
import numpy as np

# Sobre as taxas https://www.tesourodireto.com.br/conheca/regras.htm
class titulo_ntnb(object):
    def __init__(self):
        self.vf = 1
        self.tx_c = 6

    def get_pv(self, list_du, tx, vna):
        list_num = len(list_du)*[(1 + self.tx_c/100)**(1/2) - 1]
        list_num[-1] += 1
        list_pv = []
        for du, num in zip(list_du, list_num):
            temp_pv = num/(1 + tx/100)**(du/252)
            list_pv.append(temp_pv)
        pv = vna*np.sum(list_pv)
        return pv, list_pv

    def venda_antecipada(self, list_du, list_vna, ipca, tx_compra, vnac, vnacv, tx_venda, du_venda, tipo='r'):
        list_num = len(list_du)*[(1 + self.tx_c/100)**(1/2) - 1]
        list_num[-1] += 1
        list_pv = []
        list_cup = []
        pv_compra,_ = titulo_ntnb().get_pv(list_du, tx_compra, vnac)
        for du, num, vna in zip(list_du, list_num, list_vna):
            if du_venda < du:
                temp_pv = num/(1 + tx_venda/100)**((du - du_venda)/252)
                list_pv.append(temp_pv)
            else:
                list_cup.append(vna*num*((1+ipca/100)**((du_venda-du)/252))*(1 + tx_venda/100)**((du_venda - du)/252) if tipo == 'r' else num)
        pv_venda = vnacv*np.sum(list_pv)
        valor_bruto_resgate = np.sum(list_cup) + pv_venda
        rb = valor_bruto_resgate/pv_compra - 1
        rba = (1 + rb)**(252/du_venda) - 1

        return pv_venda, rb*100, rba*100
```

Python

> Capítulo 4

Capítulo 4. Fontes de Dados para Análise em Renda Fixa

Abaixo, são apresentados alguns exemplos de locais em que podemos coletar dados interessantes para análise de renda fixa.

Tesouro Direto

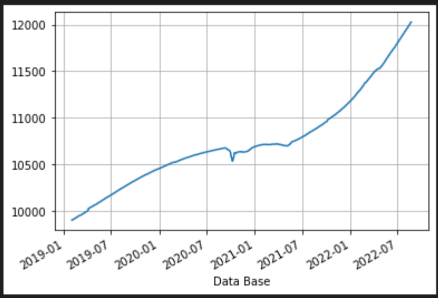
O Tesouro Direto oferece a possibilidade de consulta dos dados através da base de dados, em que é possível obter informações históricas dos títulos negociados, taxas de compra e venda, PU compra e venda, quantidades de vendas de títulos entre muitas outras informações relevantes.

Exemplos

			Taxa Compra Manha	Taxa Venda Manha	PU Compra Manha	PU Venda Manha	PU Base Manha
Tipo Titulo	Data Vencimento	Data Base					
Tesouro Prefixado	2016-01-01	2014-06-16	11.31	11.36	847.55	846.96	846.24
Tesouro IPCA+ com Juros Semestrais	2035-05-15	2014-06-20	6.09	6.19	2440.00	2412.38	2410.01
Tesouro IPCA+	2024-08-15	2014-06-20	6.12	6.20	1340.79	1330.61	1329.30
Tesouro Prefixado	2018-01-01	2014-06-20	11.79	11.85	675.80	674.53	673.93
	2017-01-01	2014-06-20	11.59	11.65	757.90	756.87	756.21
Tesouro IGPM+ com Juros Semestrais	2017-07-01	2014-08-04	5.42	5.46	3045.03	3041.92	3041.33
Tesouro IPCA+ com Juros Semestrais	2045-05-15	2014-08-04	6.17	6.27	2436.21	2404.13	2403.46
	2017-05-15	2014-08-04	5.37	5.41	2525.21	2522.75	2522.13
Tesouro IGPM+ com Juros Semestrais	2021-04-01	2014-08-04	5.92	5.98	3059.60	3050.13	3049.49
	2031-01-01	2014-08-04	6.10	6.18	4767.07	4733.91	4732.87
...	...	...	...	...	...	...	...
Tesouro IPCA+ com Juros Semestrais	2015-05-15	2013-03-21	2.93	2.97	2454.37	2452.45	2451.75
Tesouro IGPM+ com Juros Semestrais	2017-07-01	2013-03-21	3.42	3.46	3116.35	3111.75	3111.03
Tesouro IPCA+ com Juros Semestrais	2020-08-15	2013-03-21	3.70	3.76	2608.48	2599.21	2598.40
	2045-05-15	2013-03-21	4.18	4.28	3031.75	2984.22	2983.23
Tesouro IPCA+	2035-05-15	2013-03-21	4.18	4.28	917.96	898.73	898.43
Tesouro IGPM+ com Juros Semestrais	2031-01-01	2013-03-21	4.00	4.08	5612.07	5567.48	5566.05
Tesouro IPCA+ com Juros Semestrais	2024-08-15	2013-03-21	3.91	3.99	2705.46	2687.49	2686.63
Tesouro Prefixado	2016-01-01	2013-07-25	10.13	10.19	790.49	789.44	789.13
	2015-01-01	2013-07-25	9.45	9.49	877.71	877.25	876.93
Tesouro Prefixado com Juros Semestrais	2017-01-01	2013-07-25	10.20	10.26	1000.98	999.36	998.97

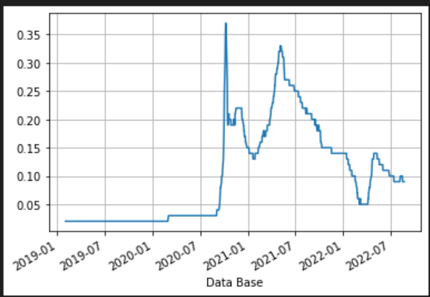
```python
selic2025['PU Base Manha'].plot(grid=True)
```

```
<AxesSubplot:xlabel='Data Base'>
```




```python
selic2025['Taxa Compra Manha'].plot(grid=True)
```

```
<AxesSubplot:xlabel='Data Base'>
```



BACEN

Outra fonte de dados interessante é a [api do bacen](), na qual é possível acessar informações históricas de índices tais como ipca, cdi, selic etc.

Exemplos de consulta em Python

```python
def con_bc(cod):
    url = 'http://api.bcb.gov.br/dados/serie/bcdata.sgs.{}/dados?formato=json'.format(cod)
    df = pd.read_json(url)
    df['data'] = pd.to_datetime(df['data'], dayfirst=True)
    df.set_index('data', inplace=True)
    return df
```

```python
ipca = con_bc(433) # índice nacional de preços ao consumidor-amplo IBGE
ipca_12 = con_bc(13522) # índice nacional de preços ao consumidor - amplo (IPCA) - em 12 meses IBGE
selic = con_bc(1178) # Taxa de juros - Selic anualizada base 252
selic_meta = con_bc(432) # Taxa de juros = Meta Selic definida pelo Copom
cdi = con_bc(4389) # Taxa de juros - CDI anualizada base 252
cdi_dia = con_bc(12) # Taxa de juros - CDI % a.d.
cdi_acum_mes = con_bc(4391) # Taxa de juros - CDI acumulada no mês % a.m.
cdi_acum_mes_anu = con_bc(4392) # Taxa de juros - CDI acumulada no mês anualizada base 252
```

SIDRA

O Sistema IBGE de Recuperação Automática (SIDRA) também é uma fonte de dados disponibilizada pelo IBGE, pode ser acessada pelo site [Sistema IBGE de Recuperação Automática - SIDRA]() ou ainda através de uma API [Home Page da API Sidra (ibge.gov.br).]()

```python
import sidrapy
```

✓ 1.1s Python

```python
data = sidrapy.get_table(table_code="1737", territorial_level="1", ibge_territorial_code="all", period="last 200")
data2 = data[['V', 'D2N', 'D3N']]
data2 = data2[data2['D3N']==data['D3N'][1]]
data2
```

✓ 3.9s Python

	V	D2N	D3N
1	2560.8200000000000	fevereiro 2006	IPCA - Número-índice (base: dezembro de 1993 =...
7	2571.8300000000000	março 2006	IPCA - Número-índice (base: dezembro de 1993 =...
13	2577.2300000000000	abril 2006	IPCA - Número-índice (base: dezembro de 1993 =...
19	2579.8100000000000	maio 2006	IPCA - Número-índice (base: dezembro de 1993 =...
25	2574.3900000000000	junho 2006	IPCA - Número-índice (base: dezembro de 1993 =...
...	...	...	...
1171	6412.8800000000000	maio 2022	IPCA - Número-índice (base: dezembro de 1993 =...
1177	6455.8500000000000	junho 2022	IPCA - Número-índice (base: dezembro de 1993 =...
1183	6411.9500000000000	julho 2022	IPCA - Número-índice (base: dezembro de 1993 =...
1189	6388.8700000000000	agosto 2022	IPCA - Número-índice (base: dezembro de 1993 =...
1195	6370.3400000000000	setembro 2022	IPCA - Número-índice (base: dezembro de 1993 =...

200 rows × 3 columns

```python
def get_ipca():
    new_dates = []
    data = sidrapy.get_table(table_code="1737", territorial_level="1", ibge_territorial_code="all", period="last 500")
    data2 = data[['V', 'D2N', 'D3N']]
    ipca = data2[data2['D3N']==data['D3N'][1]]
    list_names = ['janeiro ', 'fevereiro ', 'março ', 'abril ', 'maio ', 'junho ', 'julho ', 'agosto ', 'setembro ', 'outubro ',\
        'novembro ', 'dezembro ']
    list_dates = ['01-01-', '01-02-', '01-03-', '01-04-', '01-05-', '01-06-', '01-07-', '01-08-', '01-09-', '01-10-', '01-11-', '01-12-']
    for i in range(len(ipca['D2N'].values)):
        string_temp = ipca['D2N'].values[i]
        for j in range(len(list_names)):
            if list_names[j] in string_temp:
                string_temp = string_temp.replace(list_names[j], list_dates[j])
                new_dates.append(string_temp)
    ipca['D2N'] = pd.to_datetime(new_dates, format='%d-%m-%Y')
    ipca = ipca[['D2N', 'V']]
    ipca = ipca.rename(columns={'D2N':'Date', 'V':'Numero Indice'})
    return ipca
```

```python
df_ipca = df_ipca.reset_index(drop=True)
df_ipca
```

✓ 0.6s Python

	Date	Numero Indice
0	1981-02-01	0.0000000172555
1	1981-03-01	0.0000000181134
2	1981-04-01	0.0000000192839
3	1981-05-01	0.0000000203560
4	1981-06-01	0.0000000214792
...	...	...
495	2022-05-01	6412.8800000000000
496	2022-06-01	6455.8500000000000
497	2022-07-01	6411.9500000000000
498	2022-08-01	6388.8700000000000
499	2022-09-01	6370.3400000000000

500 rows × 2 columns