Time Value of Money

BUSI 721: Data-Driven Finance I

Kerry Back, Rice University

Overview

- PV = present value = today's value
- FV = future value = value at some particular time in the future
- r = interest rate, discount rate, expected rate of return, required rate of return, cost of capital, . . .
- n = number of periods (could be years or months or . . .)

$$\mathrm{PV} = \frac{\mathrm{FV}}{(1+r)^n} \quad \Leftrightarrow \quad \mathrm{FV} = \mathrm{PV} \times (1+r)^n$$

- used for loan calculations, retirement planning, evaluation of corporate investment projects, . . .

Example

Suppose we invest $1,000 at an annual interest rate of 5% for 2 years.

End of Year 1:

At the end of the first year, we'll have earn interest on our initial investment equal to $1,000 multiplied by 0.05. So, the balance at the end of the year would be our initial investment plus the interest earned.

Balance = Initial Investment + Interest

Balance = $1,000 * 1+$1,000 * 0.05

Balance = $1,000 * 1.05

End of Year 2:

During the second year, we'll earn interest on the balance from the end of the first year, which is $1,000 * 1.05.

Interest for the second year: ($1,000 * 1.05) * 0.05

Balance = $(1,000 * 1.05) * 1+$(1,000 * 1.05) * 0.05

Balance = $1,000 * 1.05 * 1.05

Balance = $1,000 * 1.05^2

Continuing this pattern, for any given year n, the future value of the investment will be:

FV = $1,000 * 1.05^n

Loan Balance Example

Imagine you take out a loan of $10,000 atanannualinterestrateof 5$2,500.

End of Year 1:

- Interest for the year: $10,000 * 0.05 =$500
- Total amount owed (before payment): $10,000+$500 = \$10,500
- After making the annual payment of $2,500, theremainingbalanceis :$10,500 - $2,500 =$8,000

End of Year 2:

- Interest for the year: $8,000 * 0.05$ =400
- Total amount owed (before payment): $8,000$+400 = $8,400
- After making the annual payment of $2,500, there remaining balance is$:8,400 - $2,500$ =5,900

And so on...

Loan Balances as FVs

- B = balance
- P = payment
- r = interest rate
- each period B $\mapsto$ B(1+r) - P

$$
\begin{aligned}
B_1 &= B_0(1+r) - P \\
B_2 &= B_1(1+r) - P \\
B_2 &= \left[B_0(1+r) - P\right](1+r) - P \\
B_2 &= B_0(1+r)^2 - P(1+r) - P
\end{aligned}
$$

Likewise,

$$B_3 = B_0(1+r)^3 - P(1+r)^2 - P(1+r) - P, \ldots$$

Loan Balances and PVs

- Balance = FV of initial balance - combined FVs of payments
- Divide by (1+r)^n to convert to PVs.

$$\frac{B_2}{(1+r)^2} = B_0 - \frac{P}{1+r} - \frac{P}{(1+r)^2}$$
$$\frac{B_3}{(1+r)^3} = B_0 - \frac{P}{1+r} - \frac{P}{(1+r)^2} - \frac{P}{(1+r)^3}, \ldots$$

- PV of future balance is initial balance minus combined PVs of payments

Annuity Factor and Loan Terms

- After the final payment, the future balance must be zero.
- So, PV of future balance = initial balance - combined PVs of payments implies
- **initial balance = combined PVs of all payments**

$$B_0 = \frac{P}{1+r} + \frac{P}{(1+r)^2} + \cdots + \frac{P}{(1+r)^n}$$
$$B_0 = P\left[\frac{1}{1+r} + \frac{1}{(1+r)^2} + \cdots + \frac{1}{(1+r)^n}\right]$$

- Expression in braces is called the **Annuity Factor**

Calculating Annuity Factors

- In Excel, use pv(r, n, 1)
- In python, use

```
import numpy_financial as npf
npf.pv(r, n, 1)
```

- Or, in python use

```
import numpy as np
pv_factors = (1+r)**np.arange(-1, -n-1,
-1)
np.sum(pv_factors)
```

Formula for Annuity Factor

There is also a somewhat simpler formula for the sum of PV factors.

$$\text{Annuity Factor} = \frac{1}{r}\left[1 - \frac{1}{(1+r)^n}\right]$$

pv, pmt, and rate

- What will the payment on a loan be?
- How much can you borrow?
- What must the rate be in order for your desired payment to work?

In [1]:

```python
import numpy_financial as npf

amount_borrowed = 40000
num_years = 5
rate = 0.06
```

In [2]:

```
required_payment = npf.pmt(
    rate=rate,
    pv=amount_borrowed,
    nper=num_years,
    fv=0
)

print(f"your payment will be ${-required_payment:,.2f}")
```

```
your payment will be $9,495.86
```

In [3]:

```python
payment = 10000
num_years = 5
rate = 0.06

loan_amount = npf.pv(rate=rate, nper=num_years, pmt=-payment, fv=0)
print(f"you can borrow ${loan_amount:,.2f}")
```

```
you can borrow $42,123.64
```

In [4]:
```
amount_borrowed = 40000
payment = 10000
num_years = 5

rate = npf.rate(pv=amount_borrowed, nper=num_years, pmt=-payment, fv=0)
print(f"you need a rate of {rate:,.2%} or less")
```

```
you need a rate of 7.93% or less
```

In [5]:

```
# Loan with a Balloon

amount_borrowed = 40000
payment = 10000
num_years = 5
rate = 0.1
```

In [6]:

```python
balloon = - npf.fv(
    pv=amount_borrowed,
    nper=num_years,
    pmt=-payment,
    rate=rate
)

print(f"you will have a balloon payment of ${balloon:,.2f}")
```

```
you will have a balloon payment of $3,369.40
```

Calculating with numpy

- pv = pmt * annuity_factor to get loan amount
- pmt = pv / annuity_factor to get payment
- use scipy.optimize.solve or similar to get rate

In [7]:

```python
import numpy as np

amount_borrowed = 40000
num_years = 5
rate = 0.06

pv_factors = (1+rate)**np.arange(-1, -num_years-1, -1)
annuity_factor = np.sum(pv_factors)
payment = amount_borrowed / annuity_factor
print(f"you can borrow ${loan_amount:,.2f}")
```

```
you can borrow $42,123.64
```

Monthly Payments

- Banks quote annual rates.
- They divide by 12 to get the monthly rate.
- The number of periods (nper) in the formulas should be the number of months (=12*num_years).

In [8]:

```
amount_borrowed = 40000
num_years = 5
annual_rate = 0.06
```

In [9]:

```
monthly_rate = annual_rate / 12
num_months = 12 * num_years
required_payment = npf.pmt(
    rate=monthly_rate,
    pv=amount_borrowed,
    nper=num_months,
    fv=0
)

print(f"your payment will be ${-required_payment:,.2f} each month")
```

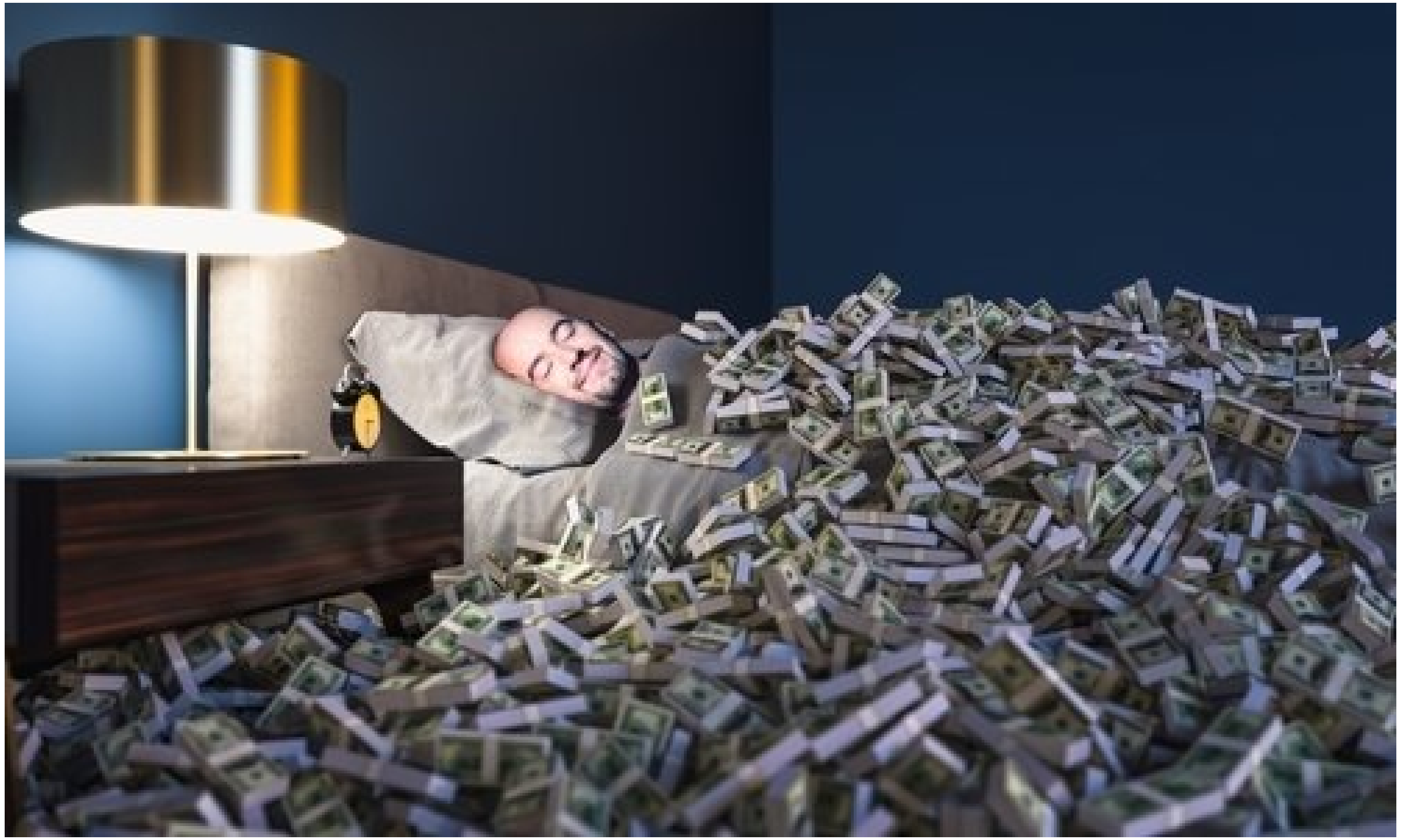
```
your payment will be $773.31 each month
```

Retirement Planning (future value problems)

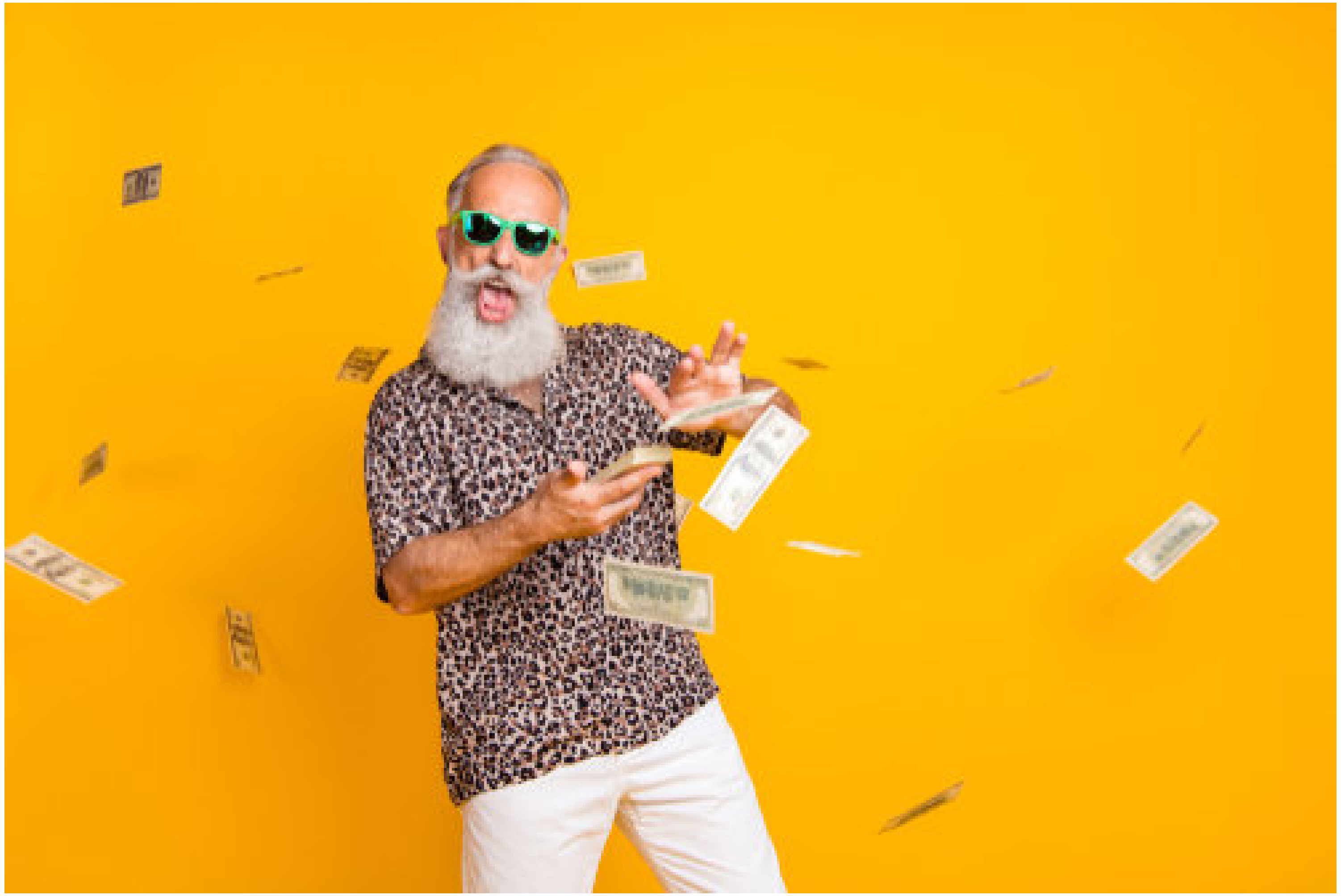
- Imagine you want to have x dollars in n years and expect to make an annual return of r. How much do you need to save each year?
- Imagine you want to spend x dollars per year for m years beginning in year n and expect to make an annual return of r. How much must you save in years 1, ..., n?

- The balance at any future date is the sum of the future values of all of the cash flows.
    - Future value of all savings for the first question.
    - Future value of all savings and spending for the second question, treating spending as negative.
- For the first question, find a savings amount such that the sum of future values is x.
    - Sum of future values = x if and only if sum of present values = PV of x
- For the second question, find a savings amount such that the sum of future values (including negative spending) is zero
    - Sum of future values = 0 if and only if sum of present values = 0
    - Solve: PV of savings = PV of spending

Our Goal



Or maybe this



Question 1

In [10]:

```
desired_balance = 2000000
num_years = 30
rate = 0.06

npf.pmt(
    pv=0,
    fv=desired_balance,
    rate=rate,
    nper=num_years
)
```

Out[10]:

```
-25297.822980094406
```

In [11]:

```
# alternatively, matching PVs:

npf.pmt(
    pv=desired_balance/(1+rate)**num_years,
    rate=rate,
    nper=num_years,
    fv=0
)
```

Out[11]:

```
-25297.822980094406
```

Question 2

In [12]:
```
spending = 100000
num_spending_years = 25
num_saving_years = 30
rate = 0.06

pv_spending_at_retirement = npf.pv(
    rate=rate,
    nper=num_spending_years,
    pmt=-spending
)

print(f"We need to have ${pv_spending_at_retirement:,.0f} at retirement.")
```

```
We need to have $1,278,336 at retirement.
```

In [13]:

```
saving = -npf.pmt(
    pv=0,
    rate=rate,
    fv=pv_spending_at_retirement,
    nper=num_saving_years
)

print(f"We need to save ${saving:,.0f} each year.")
```

```
We need to save $16,170 each year.
```