Stochastic Methods Lab, Jacobs University Bremen, Fall 2020
Prof. Dr. Soeren Petrat

Session 2
Sep. 7, 2020

Steps for setting up git (see also "Intro to git for academics" on website):

- download and install git (git-scm.com)
- open command line (git CMD) to configure git (see "Intro")
- get bitbucket account (bitbucket.org), register with Jacobs email address
- on bitbucket: • fork repository spetrat/sml_2020, mark as private!
  • under "User and group access", add s.petrat@jacobs... and s.agrawal@jacobs... with "write" access.
- on your computer: clone your repository (via command line)
  your own branch that you just created

0.2 Scientific Python

- optimized for vectorized operations (runs near machine speed, although interpreted language), using NumPy arrays
- SciPy package comprises functionality of all aspects of scientific computing

ToDo:

Install Anaconda package (version 3.8 or higher)

↳ SciPy library already included

↳ spyder editor (development environment)

↳ jupyter notebooks (editor + comments + run code fragments separately)

Code examples will be discussed in class, and added to git folder

1. Basics of Financial Math

1.1 Time Value of Money

r = annual interest rate

FV = future value, PV = present value

after n years: $\boxed{FV = PV(1+r)^n}$ (the "value" of money changes in time)

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

If interest is compounded m times a year: $FV = PV\left(1+\frac{r}{m}\right)^{n \cdot m}$

teminology:
- BEY (bond-equivalent yield)
  annualized yield, compounded semi-annually (twice a year)
- MEY (mortgage-equivalent yield)
  annualized yield, compounded monthly

An effective annual interest rate r_{eff} allows us to compare financial instruments with different compounding standards:

$$(1+r_{eff})^n = \left(1+\frac{r}{m}\right)^{n \cdot m} \Rightarrow 1+r_{eff} = \left(1+\frac{r}{m}\right)^m$$

$$\Rightarrow \boxed{r_{eff} := \left(1+\frac{r}{m}\right)^m - 1}$$

Example: $r = 10\%$ BEY (compounded twice a year)

$$r_{eff} = \left(1 + \frac{0.1}{2}\right)^2 - 1 = 0.1025 = 10.25\%$$

Is r_{eff} always bigger than r? $(m \geq 1, r > 0)$

↳ $r_{eff} = \left(1 + \frac{r}{m}\right)^m - 1 = 1 + m\frac{r}{m} + \underbrace{\text{rest}}_{>0 \text{ if } r>0} - 1 \geq r \Rightarrow$ Yes

recall: Bernoulli's inequality $(1+x)^m \geq 1 + mx$ holds for all $x > -1$

$\Rightarrow r_{eff} \geq r$ as long as $\frac{r}{m} > -1$

One often useful idealization is that of continuous compounding, i.e., taking the limit $m \to \infty$:

$$\Rightarrow FV = PV \lim_{m\to\infty} \left(1 + \frac{r}{m}\right)^{n \cdot m}$$

$$= PV \underbrace{\left[\lim_{m\to\infty} \left(1 + \frac{r}{m}\right)^m\right]^n}_{= e^r}$$

$$= PV \; e^{rn} = PV \exp(rn)$$

↑ exponential fct. (recall from calculus)

1.2 General Cash Flows

n years, r the yearly interest rate

Suppose at the end of the year, there are cash flows $C_1, C_2, C_3, \dots, C_n$

(C_i at the end of year i).

$$\Rightarrow \quad PV = \frac{C_1}{1+r} + \frac{C_2}{(1+r)^2} + \dots + \frac{C_n}{(1+r)^n} = \sum_{i=1}^{n} \frac{C_i}{(1+r)^i}$$

Note: future value after n years:

$$FV_n = C_1(1+r)^{n-1} + C_2(1+r)^{n-2} + \dots + C_{n-1}(1+r) + C_n = \sum_{i=1}^{n} C_i(1+r)^{n-i}$$

To compute present values in python, set $x = \frac{1}{1+r}$, and then we need to evaluate polynomials: $\sum_{i=1}^{n} C_i x^i$

In python:

- best: vectorized operations

  define vector $i = \text{arange}(1, n+1)$, suppose vector C given

  we could compute vector $\underbrace{X{*}{*}{*}i}_{\text{to the power}} = \begin{pmatrix} X{*}{*}{*}1 \\ X{*}{*}{*}2 \\ \vdots \\ X{*}{*}{*}n \end{pmatrix}$

  ("number to the power of a vector" does not make sense mathematically, but in python it is very useful.)

  use dot product (scalar product) to evaluate sum: $PV = \text{dot}(C, X{*}{*}{*}i)$

- explicit loop (discouraged, very slow)

- Horner's scheme:

$$PV = \left(\ldots\left(\left(C_n x + C_{n-1}\right)x + \ldots + C_3\right)x + C_2\right)x + C_1\Big)x$$

(fewer operations than explicit loop)

- polyval (fct. from SciPy), uses optimized Horner's scheme
  ↳ look up documentation

Annuity: $C_i = C$ for all i

types:

- ordinary annuity (usually assumed): pays C at end of year

$$FV = \sum_{i=0}^{n-1} C(1+r)^i = C \underbrace{\sum_{i=0}^{n-1} (1+r)^i}$$

this is a geometric series

geometric series: $x\sum_{i=0}^{N} x^i = \sum_{i=1}^{N+1} x^i = \sum_{i=0}^{N} x^i + x^{N+1} - 1$

$$\Rightarrow (x-1)\sum_{i=0}^{N} x^i = x^{N+1} - 1$$

$$\Rightarrow \sum_{i=0}^{N} x^i = \frac{x^{N+1} - 1}{x - 1}$$

$$\Rightarrow FV = C\,\frac{(1+r)^n - 1}{1+r-1} = C\,\frac{(1+r)^n - 1}{r}$$

- annuity due : pays at beginning of year

$$\Rightarrow FV = C\sum_{i=1}^{n}(1+r)^i = C(1+r)\sum_{i=0}^{n-1}(1+r)^i$$

$$= C(1+r)\,\frac{(1+r)^n - 1}{r}$$