Workshop 7: Stochastic processes & wealth dynamics

FIE463: Numerical Methods in Macroeconomics and Finance using Python

Richard Foltyn
NHH Norwegian School of Economics

February 27, 2025

See GitHub repository for notebooks and data:

https://github.com/richardfoltyn/FIE463-V25

Exercise 1: Wealth dynamics with AR(1) returns

Recall the household wealth dynamics we studied in the previous lecture, where assets $a_{i,t}$ evolved according to

$$a_{i,t+1} = R s a_{i,t} + y_{i,t+1}$$

and we assumed a fixed savings rate s, a fixed gross return R and some stochastic income process $y_{i,t}$.

In this exercise, we alter this setting to *stochastic returns* which follow an AR(1) so that the model of wealth dynamics is now given by

$$\begin{aligned}
a_{i,t+1} &= R_{i,t+1} s a_{i,t} + y_{i,t+1} \\
\log R_{i,t+1} &= \mu_r + \rho_r \log R_{i,t} + \epsilon_{i,t+1}, \qquad \epsilon_{i,t+1} \stackrel{\text{iid}}{\sim} \mathcal{N}\left(0, \sigma_r^2\right) \\
\log y_{i,t+1} &\stackrel{\text{iid}}{\sim} \mathcal{N}\left(\mu_y, \sigma_y^2\right)
\end{aligned}$$

where $R_{i,t+1}$ follows an AR(1) in logs and $y_{i,t+1}$ is log-normally distributed.

Parameters

To remain comparable with the scenarios discussed in the lecture, we set the following parameters:

Parameter	Description	Value
s	Savings rate	0.75
σ_y	Volatility of log labor income	0.1
μ_y	Mean of log labor income	$-\frac{1}{2}\sigma_y^2$
ρ_r	Autocorrelation of log returns	0.6
σ_r	Volatility of log return shocks	0.2
μ_r	Intercept of log returns	$(1-\rho_r)\log(1.1) - \frac{1}{2}\frac{\sigma_r^2}{1+\rho_r}$

The parameter μ_y is chosen so that average income in levels is one, $\mathbb{E}[y_{i,t}] = 1$, while μ_r is chosen so that average gross returns are 1.1 as in the lecture, i.e., $\mathbb{E}[R_{i,t}] = 1.1$.

The following code defines the parameters class for this problem:

[1]:
```python
# Parameters
import numpy as np
from dataclasses import dataclass

@dataclass
class Parameters:
    """
    Parameters for model with stochastic returns.
    """
    s: float = 0.75                                   # Exogenous savings rate
    sigma_y: float = 0.1                              # Standard deviation of log income
    mu_y: float = -sigma_y**2.0/2.0                   # Mean of log income
    rho_r: float = 0.6                                # Persistence of log gross returns
    sigma_r: float = 0.2                              # Standard deviation of log gross returns
    mu_r: float = (1-rho_r) * np.log(1.1) - sigma_r**2/2/(1+rho_r)     # Mean of log gross
↪returns
```

[2]:
```python
# Create an instance of the Parameters class
par = Parameters()
```

The following code uses functions in the module `workshop_ex1` to compute selected moments of the income and return processes to verify that the calibration of parameters yields the desired values:

[3]:
```python
# Enable automatic reloading of external modules
%load_ext autoreload
%autoreload 2
```

[4]:
```python
import numpy as np
from workshop07_ex1 import compute_lognorm_mean_var, compute_return_ar1_mean

# Mean and variance of income
y_mean, _ = compute_lognorm_mean_var(par.mu_y, par.sigma_y)

# Conditional variance of gross returns
_, R_var = compute_lognorm_mean_var(par.mu_r, par.sigma_r)

# Unconditional mean of gross returns
log_R_mean = compute_return_ar1_mean(par)

print(f'Mean income:                    {y_mean:.3f}')
print(f'Mean gross return:              {log_R_mean:.3f}')
print(f'Cond. std. dev. of gross return: {np.sqrt(R_var):.3f}')
```

```
Mean income:                    1.000
Mean gross return:              1.100
Cond. std. dev. of gross return: 0.211
```

We are interested in simulating the wealth dynamics and compute the Gini coefficient using the same approach as we did in the lecture.

1. Write a function `simulate_wealth()` to simulate the wealth trajectories of a panel of households who face AR(1) returns. The function signature should look as follows:

```python
def simulate_wealth(par: Parameters, a0, T, N, rng=None):
    """
    Simulate the evolution of wealth when returns are stochastic.

    Parameters
    ----------
    par : Parameters
    a0 : float
        Initial wealth.
    T : int
```

```
        Number of time periods to simulate.
    N : int
        Number of individuals to simulate.
    rng : numpy.random.Generator, optional
        A random number generator instance.

    Returns
    -------
    a_sim : numpy.ndarray
        A (T+1, N) array of simulated wealth paths.
    """
```

Set the initial value of $\log R_{i,t}$ to the unconditional mean $\frac{\mu_r}{1-\rho_r}$ for all households.

Hint: Use the wealth simulation routine from the lecture as a template and make the necessary changes

2. Using an initial wealth of $a_0 = 1$ for all households, simulate $N = 20$ households for $T = 100$ periods. Plot the wealth trajectories for these households in a single graph and also include the average simulated wealth.
3. Simulate a larger panel of $N = 1,000,000$ households for $T = 200$ periods. Compute the cross-sectional mean and variance of wealth for each period t and plot these in a figure with two subplots (one for the mean, one for the variance).

   How do these plots compare to the scenarios (with IID and AR(1) income) discussed in the lecture?
4. From the previous plots you suspect that the Gini coefficient changes somewhat across periods. Compute a cross-sectional Gini coefficients each of the last 100 periods of your simulation and plot these Ginis as a time series. Add a horizontal line indicating the average Gini coefficient.

   Hint: Use the `gini()` function from the `workshop07_ex1` module for this purpose.

Exercise 2: AR(1) vs Random Walk

Recall the AR(1) process we studied in the lecture, defined as

$$x_{i,t+1} = \rho x_{i,t} + \epsilon_{i,t+1}\,, \qquad \epsilon_{i,t+1} \stackrel{\text{iid}}{\sim} \mathcal{N}\left(0, \sigma^2\right)$$

In the lecture, we restricted our attention to the stationary case with $\rho \in (-1, 1)$. With $\rho = 1$, the above process is called a Random walk which no longer is no longer stationary as its variance is linearly increasing in time.

To demonstrate this, perform the following tasks:

1. Write a function `simulate_panel()` which simulates a panel of individuals i where each individual-specific realization $x_{i,t}$ follows the stochastic process defined above. The function signature should look as follows:

```
def simulate_panel(rho, sigma, T, N, x0=0, rng=None):
    """
    Simulates a panel of stochastic processes.

    Parameters
    ----------
    rho : float
        The autoregressive parameter.
    sigma : float
        The standard deviation of the noise term.
    T : int
```

```
        The number of time periods to simulate.
    N : int
        The number of individuals to simulate.
    x0 : float, optional
        The initial value of the process.
    rng : Generator, optional
        Random number generator to use.

    Returns
    -------
    numpy.ndarray
        A (T+1, N) array with the simulated values.
    """
```

2. Let $\sigma = 0.1$. Simulate the trajectories of a cross section of $N = 100,000$ individuals for $T = 300$ periods for two different scenarios:

    1. AR(1) with $\rho = 0.9$;
    2. Random walk with $\rho = 1$

    Make sure to use the same seed for both simulations.

3. Create a figure with two subplots:

    1. The first subplot should contain two lines showing the average value of the simulated AR(1) and Random walk for each period t, i.e., average across N individuals for each t.
    2. The second subplot should contain two lines showing the variance of the simulated AR(1) and Random walk for each period t.

4. Repeat the previous exercise, but use $\rho = 0.99$ for the AR(1) instead. How does behavior of the cross-sectional mean and variance of the AR(1) change?