FE621 - Homework #5

Author: Sid Bhatia

Date: May 10th, 2024

Pledge: I pledge my honor that I have abided by the Stevens Honor System.

Professor: Sveinn Olafsson

TA: Dong Woo Kim

Problem 1 (Portfolio Wealth Growth)

1.1 Portfolio Wealth Growth Theory

This section delves into the theoretical mathematical foundation governing the growth of portfolio wealth over time. The analysis is crucial for understanding how investments evolve under the influence of various market factors, including returns and volatility.

1.1.1 Mathematical Formulation

The wealth process $\{V_t\}_{t\geq 0}$ is modeled as a geometric Brownian motion (GBM), which is frequently used to represent stock prices and, by extension, portfolio values under stochastic environments. The stochastic differential equation (SDE) governing this process is given by:

$$\frac{dV_t}{V_t} = \mu dt + \sigma dW_t \tag{1}$$

Portfolio Wealth Growth Simulation vs. Expectation Here:

- V_t represents the portfolio value at time t.
- μ is the expected return of the portfolio, expressed as a percentage of the portfolio value.
- σ is the volatility of the portfolio, which measures the standard deviation of the portfolio's returns.
- dW_t is the increment of a standard Brownian motion, which captures the random fluctuations in the market.

Interpretation

Equation (1) can be interpreted as follows:

- The term μdt captures the expected growth of the portfolio due to returns over an infinitesimally small time interval dt.

- The term σdW_t introduces randomness into the growth process, reflecting the uncertainty and risk inherent in the financial markets.

Solution to the Differential Equation

The solution to the stochastic differential equation (SDE) given in equation (1) can be expressed explicitly by integrating both sides over the interval from 0 to t:

$$\ln \frac{V_t}{V_0} = \left(\mu - \frac{\sigma^2}{2}\right) t + \sigma W_t \tag{2}$$

where:

- V_0 is the initial value of the portfolio at time $t = 0$.
- W_t represents the standard Brownian motion at time t.

From equation (2), we can exponentiate both sides to obtain the explicit form of V_t:

$$V_t = V_0 \exp\left((\mu - \frac{\sigma^2}{2})t + \sigma W_t\right) \tag{3}$$

Mathematical Synthesis

Equation (3) clearly shows how the portfolio value V_t evolves over time. It indicates that the portfolio value is log-normally distributed with its mean and variance increasing over time. This formulation is fundamental in finance for modeling asset prices and helps in understanding the dynamic nature of investment growth under uncertainty.

1.1.2 Expectation of Portfolio Wealth

The following section explores and delves into the expectation (first raw moment/arithmetic average) of the portfolio wealth process.

Expectation Calculation

Given the wealth process V_t which follows a geometric Brownian motion (GBM) as described by:

$$\frac{dV_t}{V_t} = \mu dt + \sigma dW_t \tag{1}$$

The solution to this stochastic differential equation (SDE) indicates:

$$V_t = V_0 \exp\left((\mu - \frac{\sigma^2}{2})t + \sigma W_t\right) \tag{3}$$

To find the expectation $\mathbb{E}[V_t]$, we note that W_t is a standard Brownian motion (BM), which implies σW_t is normally distributed with mean 0 and variance $\sigma^2 t$. Thus, $\sigma W_t \sim N(0, \sigma^2 t)$, and $e^{\sigma W_t}$ follows a log-normal distribution.

We can use the moment-generating function (MGF) of a normally distributed random variable to compute the expectation of a log-normal variable. For a random variable $X \sim \mathcal{N}(\mu_X, \sigma_X^2)$, the MGF of X at s is $M_X(s) = e^{\mu_X s + \frac{1}{2}\sigma_X^2 s^2}$. Setting $s = 1$, we find:

$$\mathbb{E}[e^X] = e^{\mu_X + \frac{1}{2}\sigma_X^2} \tag{4}$$

Applying this to our case, where $\mu_X = 0$ and $\sigma_X^2 = \sigma^2 t$, we get:

$$\mathbb{E}[e^{\sigma W_t}] = e^{0 + \frac{1}{2}\sigma^2 t} = e^{\frac{1}{2}\sigma^2 t} \tag{5}$$

Now, substituting this into the solution for V_t:

$$\begin{aligned}\mathbb{E}[V_t] &= \mathbb{E}\left[V_0 \exp\left((\mu - \frac{\sigma^2}{2})t + \sigma W_t\right)\right] \\ &= V_0 \exp\left((\mu - \frac{\sigma^2}{2})t\right)\mathbb{E}[e^{\sigma W_t}]\end{aligned} \tag{6}$$

Substituting the expectation of $e^{\sigma W_t}$:

$$\begin{aligned}\mathbb{E}[V_t] &= V_0 \exp\left((\mu - \frac{\sigma^2}{2})t\right)\exp\left(\frac{1}{2}\sigma^2 t\right) \\ &= V_0 \exp(\mu t)\end{aligned} \tag{7}$$

Thus, the expected wealth at time (t) is indeed given by:

$$\mathbb{E}[V_t] = V_0 \exp(\mu t) \tag{7}$$

This demonstrates that the expectation grows exponentially at a rate determined by the drift μ, independent of the volatility σ.

1.2 Portfolio Wealth Growth Implementation

The following section implements applications of factors governing the growth of portfolio wealth over time in Python.

1.2.1 True vs. Expected Path Simulation

This Python code snippet simulates 50 paths of a portfolio's wealth process $\{V_t\}_{t\in[0,T]}$ modeled as a geometric Brownian motion (GBM) alongside the expected (arithmetic average) path $\{\mathbb{E}[V_t]\}_{t\in[0,T]}$. We use the parameters $\mu = 0.08$, $\sigma = 0.2$, $T = 30$ years, and an initial portfolio value $V_0 = 100$.

Code Breakdown

Step 1: Import Libraries:

- `numpy` for numerical operations.
- `matplotlib.pyplot` for plotting the results.

```
import numpy as np
import matplotlib.pyplot as plt
```

Step 2: Set Parameters:

- `mu` : the expected return rate of the portfolio.
- `sigma` : the volatility or standard deviation of returns.
- `T` : the total time horizon for the simulation (30 years).
- `dt` : the time increment for each step in the simulation.
- `V0` : the initial portfolio value.
- `N` : the number of time steps calculated as the total time divided by the increment.
- `num_paths` : the number of simulation paths.

```
mu = 0.08       # drift coefficient
sigma = 0.2     # volatility coefficient
T = 30          # time horizon
dt = 0.01       # time increment
V0 = 100        # initial wealth
N = int(T/dt)   # number of time steps
num_paths = 50  # number of paths to simulate
```

Step 3: Simulate Paths:

- Generate multiple paths of the GBM using random normal distributions to simulate daily returns.
- Calculate the portfolio value over time for each path based on the GBM formula.

```
np.random.seed(42)  # for reproducibility
paths = np.zeros((num_paths, N))
for i in range(num_paths):
    dB = np.sqrt(dt) * np.random.normal(size=N-1)
    W = np.cumsum(dB)
    W = np.insert(W, 0, 0)  # insert the initial condition W_0 = 0
    paths[i] = V0 * np.exp((mu - 0.5 * sigma**2) * t + sigma * W)
```

Step 4: Calculate Expected Path:

- Compute the expected path using the deterministic part of the GBM formula.

```
expected_path = V0 * np.exp(mu * t)
```

Step 5: Plot the Results:

- Plot all simulated paths and the expected path to visualize the potential variance around the expected growth.

```
plt.figure(figsize=(12, 8))
for path in paths:
    plt.plot(t, path, 'r', linewidth=0.5, alpha=0.5)  # red lines for
```

```
simulated paths
plt.plot(t, expected_path, 'b', linewidth=2.5, label='Expected Path
$\mathbb{E}[V_t]$')  # blue line for the expected path
plt.title('Simulation of Portfolio Wealth Growth and Expected Path')
plt.xlabel('Time (years)')
plt.ylabel('Portfolio Value')
plt.legend()
plt.grid(True)
plt.show()
```

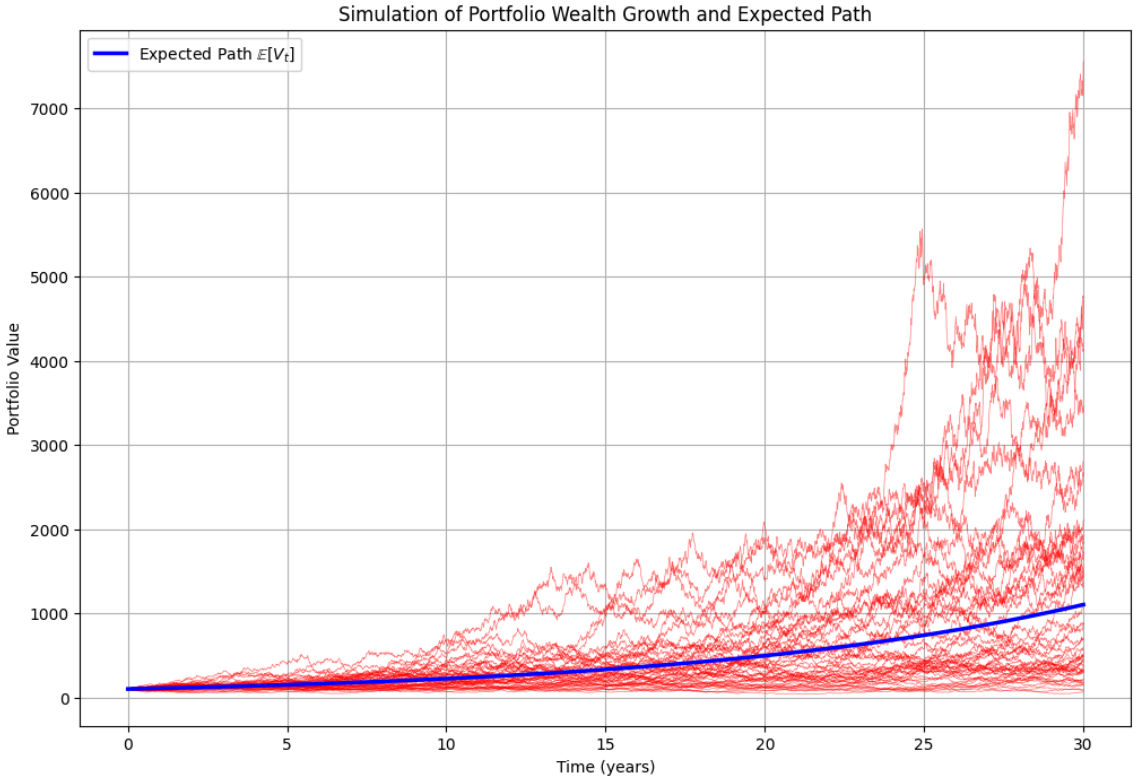

Figure 1 - Portfolio Wealth Growth Simulation vs. Expectation

1.2.2 Underperformance Confidence Interval Simulation

In this section, we estimate the probability that the terminal wealth V_T underperforms the expected terminal wealth $\mathbb{E}[V_T]$ by varying degrees specified by α. We will compute the 95% confidence intervals for these probabilities.

Code Breakdown

Step 1: Library Importation

First, we need to import the necessary Python libraries for calculations and data handling.

```
import numpy as np
import scipy.stats as stats
```

Step 2: Parameter Definition

We will define the parameters for the simulation, including the number of simulations n, and setup the range for α.

```
# Parameters
mu = 0.08       # drift coefficient
sigma = 0.2     # volatility coefficient
T = 30          # time horizon
V0 = 100        # initial wealth
n = 10000       # number of simulations
alphas = np.arange(1, 0, -0.1)  # range of alpha from 1 to 0.1
```

Step 3: Terminal Wealth Value Simulation

Simulate the terminal wealth values V_T using the geometric Brownian motion model.

```
np.random.seed(42)  # for reproducibility
terminal_values = V0 * np.exp((mu - 0.5 * sigma**2) * T + sigma *
np.sqrt(T) * np.random.normal(size=n))
```

Step 4: Underperformance Probability Computation

For each α_i, calculate the probability that V_T is less than or equal to $\alpha \times \mathbb{E}[V_T]$. The expectation $\mathbb{E}[V_T]$ is computed based on its analytical expression.

```
expected_VT = V0 * np.exp(mu * T)  # calculate expected V_T
probabilities = [np.mean(terminal_values <= alpha * expected_VT) for alpha
in alphas]
```

Step 5: Confidence Interval Computation

Calculate the 95% confidence intervals for the probabilities of underperformance using the normal approximation.

```
confidence_intervals = [stats.norm.interval(0.95, loc=p, scale=np.sqrt((p*
(1-p))/n)) for p in probabilities]
```

Step 6: Result Output

Finally, output the results in a structured format.

```
print("Alpha\tProbability\t95% Confidence Interval")
for alpha, p, ci in zip(alphas, probabilities, confidence_intervals):
    print(f"{alpha:.1f}\t{p:.4f}\t{ci}")
```