DATA COLLECTION

Data Sources

Our dataset comes from publicly available U.S. financial and economic sources accessed through **R's built-in datasets**:

- **FRED (Federal Reserve Economic Data)** – Unemployment rate, personal savings rate
- **Bureau of Labor Statistics (BLS) –** Labor-related variables
- **S&P 500 –** Daily OHLCV (open, high, low, close, volume) stock-market data

Our teammate Wonjae collected and merged all three into one unified time-series dataset.

Population: Monthly measurements of the U.S. economy and stock market from 1985–2025 (40 years).

Sample

S&P 500 daily OHLCV was converted into **monthly** data.

Final dataset: **~480 monthly observations**

Training set: 391 rows × 7 columns

Test set: 98 rows × 7 columns

Temporal split: We used the most recent observations for testing instead of a random split, which is standard for time-series forecasting.

Features

All features are continuous and numeric:

1. **Open**
2. **High**
3. **Low**
4. **Close**
5. **Volume**
6. **Unemployment rate**
7. **Personal savings rate**

We used expanding-window normalization, which avoids look-ahead bias by only using past information for scaling.

Total final features: 7

Potential Limitations & Biases

- **Mixed-frequency bias:** S&P data is daily, but macroeconomic indicators are monthly. Aggregation may lose information.
- **Reporting lag:** Economic indicators (especially unemployment) are published with delays, which can affect prediction accuracy.
- **Non-stationarity:** Markets change over decades, which makes long-term modeling difficult.

Why We Did Not Collect Our Own Data

Collecting financial and economic data manually introduces:

- human error,
- short time horizons,
- inconsistent measurements,
- and poor reliability.

Instead, we use reputable institutional sources (FRED, BLS, and S&P data) that provide **high-quality, standardized, 40-year time-series data** suitable for machine-learning and reinforcement-learning analysis.

We began with **daily S&P 500 OHLCV data** and converted it into **monthly data** to match the frequency of the economic indicators. From this, we computed **monthly percent-change returns**.

Because financial time series have strong long-term trends and noise, we applied **fractional differencing**. Unlike regular differencing, fractional differencing removes drift **without destroying long-memory patterns**, which helps keep important information for prediction. This step makes the series more stationary and easier for the reinforcement-learning model to learn from.

Data Preparation Flow

1. Daily S&P 500 OHLCV Data

↓

2. Convert to Monthly Data

(aggregate daily prices → monthly open, high, low, close, volume)

↓

3. Compute Monthly Percent-Change Returns

↓

4. Fractional Differencing

- removes long-term drift
- keeps long-memory patterns
- stabilizes the time series

↓

5. Expanding-Window Normalization

(only uses past information → avoids look-ahead bias)

↓

6. Final Feature Matrix (7 continuous features)

Used for training the reinforcement learning model

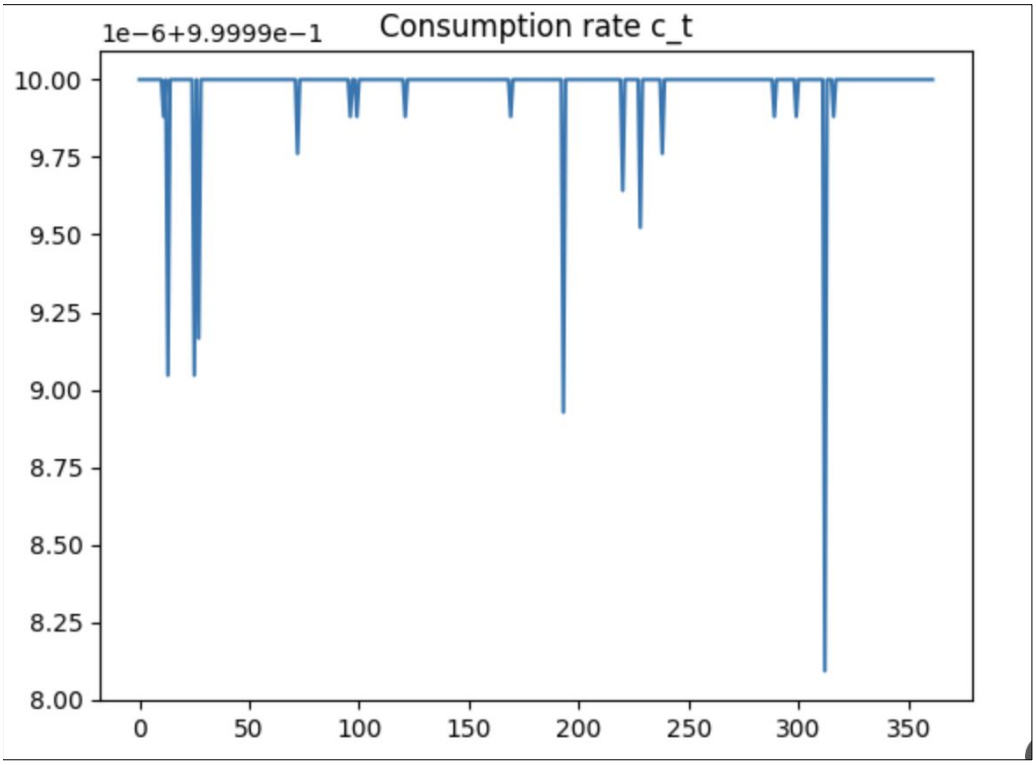
Visualizations

The **first wealth plot** showed the model failing — the line drops straight down because the model was untrained and didn't know how to act.

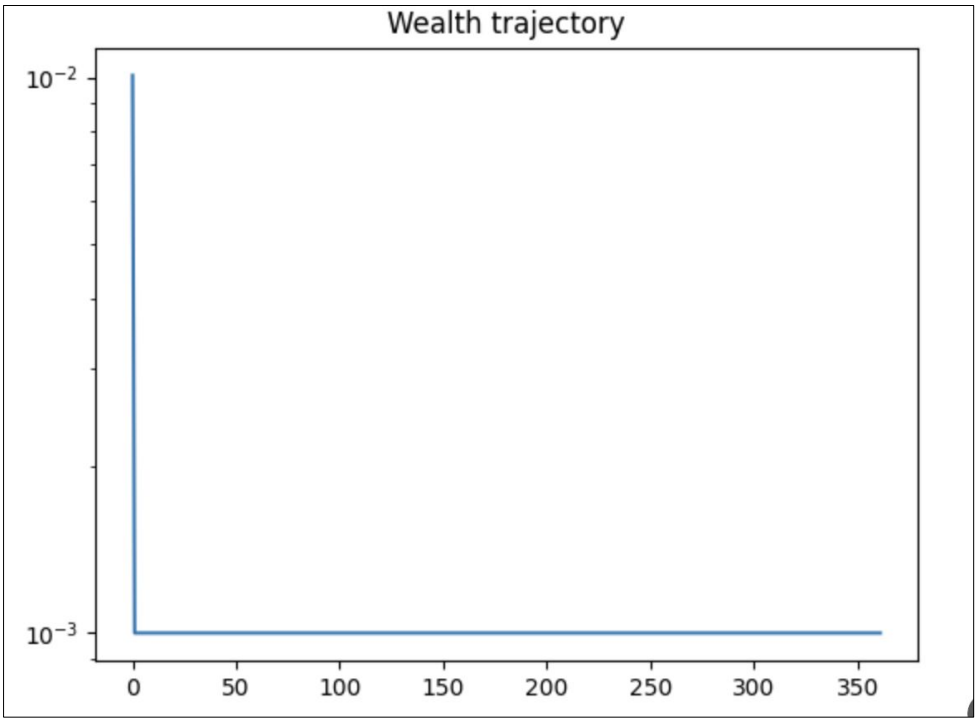
The **first consumption plot** looked messy with random dips because the model was still experimenting.

The **second wealth plot** (after training) shows a much smoother, upward path, meaning the model finally learned a good strategy.

The **later consumption plot** is more stable and makes more sense once the model started learning correctly.

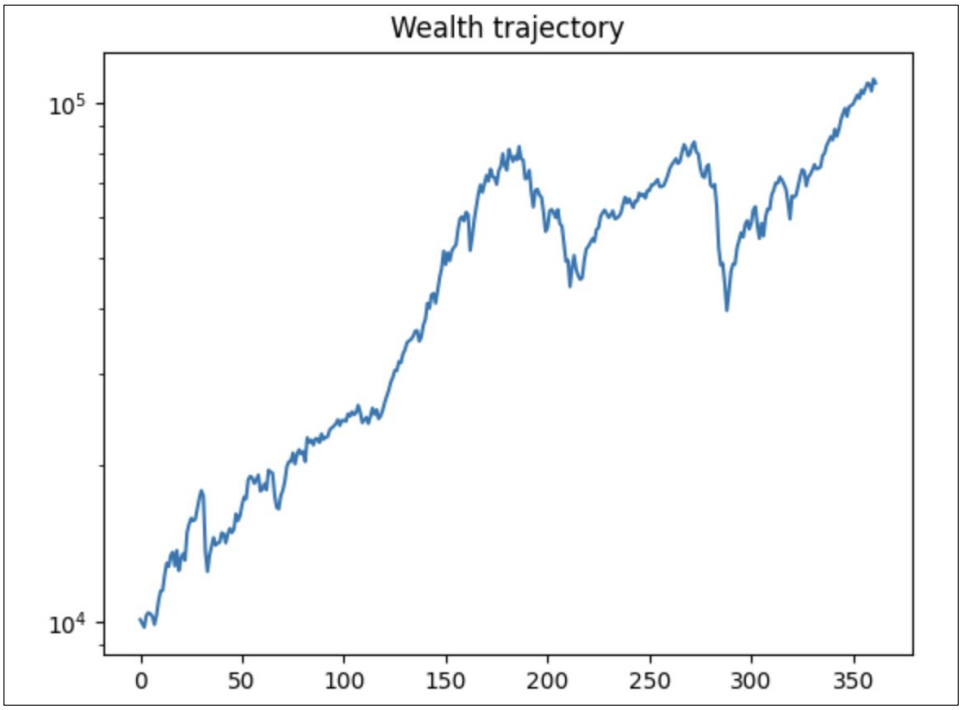

Consumption rate c_t
1e−6+9.9999e−1
10.00
9.75
9.50
9.25
9.00
8.75
8.50
8.25
8.00
0
50
100
150
200
250
300
350

1

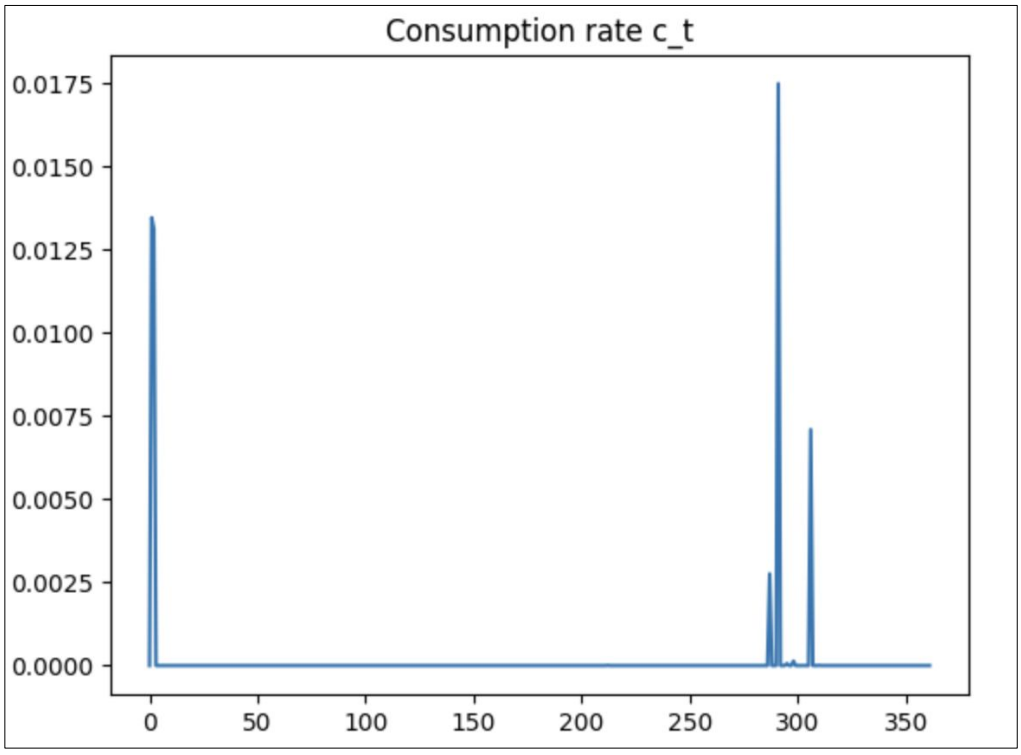

Wealth trajectory
10^{-2}
10^{-3}
0
50
100
150
200
250
300
350

2

3


Wealth trajectory
10^5
10^4
0
50
100
150
200
250
300
350

4


Consumption rate c_t
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
0
50
100
150
200
250
300
350

Summary Statistics

Mean return: **0.84%**

Std of returns: **0.0436**

Min return: **–0.2176**

Max return: **0.1317**

Unemployment range: **5,481–15,352**

Savings rate range: **2.2%–12%**

**Returns are noisy, while unemployment and savings move slowly.*

What does it mean ?

Mean return: 0.84%

The market *on average* goes up slightly each month.
→ This tells the model that most months are positive, but prediction requires detecting the *exceptions*.

Min return: –0.2176

The worst months are *very* negative.
→ These large crashes are important for the model to learn — unemployment or savings spikes may help signal them.

Max return: 0.1317

The best months are strongly positive.
→ The model should learn which economic conditions often precede strong market gains.

Unemployment range: 5,481–15,352

Unemployment changes slowly across time.
→ Because unemployment is stable month-to-month, its value may help detect long-term economic cycles (expansion vs recession), which influence returns.

Savings rate range: 2.2%–12%

Savings also changes slowly.
→ Higher savings may signal economic stress (people hold cash), while lower savings may signal confidence — useful clues for predicting returns.

"Returns are noisy, while unemployment and savings move slowly."

→ Stock returns jump wildly from month to month.
→ Economic indicators move gradually over long periods.

Why this matters:

The AI must learn how *slow-moving* economic conditions connect to *fast-moving* market returns — meaning the relationship is subtle and nonlinear.

Statistical Overview of the Dataset

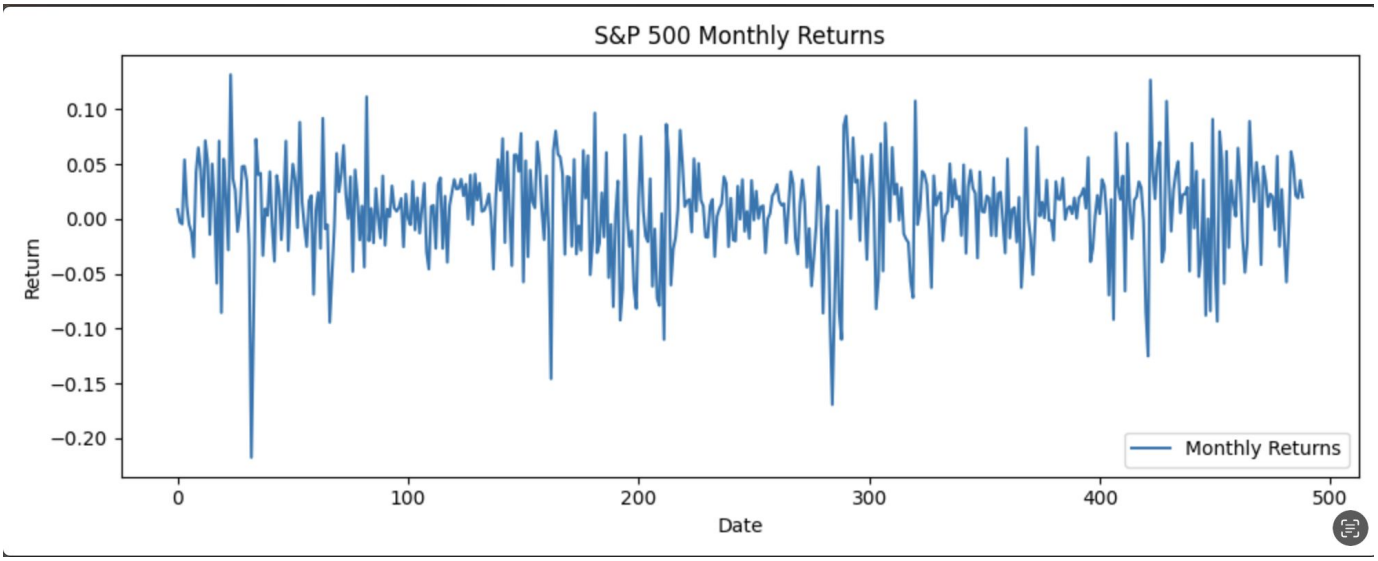


Figure: S&P 500 Monthly Returns

Returns fluctuate sharply from month to month, with both large positive and negative spikes. This confirms that the stock market is highly volatile, which makes prediction challenging.

At the beginning, the model performed really badly because it wasn't trained yet. It didn't understand how unemployment, savings, or market returns were connected, so its decisions looked random. That's why the early graphs show wealth crashing and the lines looking chaotic.

After training and especially after fixing the loss function the model finally started learning the patterns we wanted it to see. It began to understand how slow-moving indicators like unemployment and personal savings can help signal whether the stock market might have a good or bad month.

Once it learned these relationships, the model's behavior became much more stable: the wealth curve started rising instead of collapsing, and the consumption decisions smoothed out.

This shows the model beginning to answer our main question:
Can we teach an AI to use things like unemployment and savings to predict the market and make better investment decisions?
After training, the graphs show that the answer is **yes it can learn to invest more effectively over time.**

What We Learned

Path Dependent Learning Dynamics of ICM PPO with Epstein Zin Utility in Time Series Reinforcement Learning

GROUP NAME:CODING CO