Random Investment Strategy Analysis

Hypothesis Testing and Performance Evaluation

Your Name

2025-11-17

Table of contents

Random Investment Strategy: Hypothesis Analysis

This report evaluates the Random Investment Strategy hypothesis - a contrarian approach testing whether portfolio selection through randomization can outperform traditional methods.

Research Question: Can a purely random stock selection strategy generate competitive risk-adjusted returns compared to market benchmarks?

This template demonstrates Tufte-style margins, financial tables, and professional PDF output

Executive Summary

The Random Investment Strategy challenges conventional portfolio theory by testing whether systematic randomization can produce competitive returns.

Hypothesis Testing: - Random selection vs. benchmarks - Risk-adjusted performance - Statistical significance

Methodology

We simulate the Random Investment Strategy by selecting stocks through controlled randomization and evaluating performance metrics against theoretical foundations (Markowitz 1952).

Markowitz, Harry. 1952. "Portfolio Selection." The Journal of Finance 7 (1): 77–91.

```python
# Import required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotext as plt_term
from finmodel import PlotextChart, FinancialTable
```

Strategy Assumptions

The Random Investment Strategy uses the following assumptions for portfolio construction and evaluation.

Assumptions are displayed using Financial Modeling Standard format

```python
# Strategy Assumptions - displayed as Financial
    Modeling Standard
assumptions_data = pd.DataFrame({
    'Value': [
        100_000,      # Initial capital
        3,            # Number of stocks
        100,          # Trading days
        0.02,         # Expected volatility Stock A
        0.015,        # Expected volatility Stock B
        0.025,        # Expected volatility Stock C
    ]
}, index=[
    'Initial Capital ($)',
    'Portfolio Size (N)',
    'Simulation Period (days)',
    'Stock A Volatility',
    'Stock B Volatility',
    'Stock C Volatility',
])

# Display as Assumption table (grey background)
FinancialTable(assumptions_data,
    style='assumptions')
```

	Value
Initial Capital ()	100,000.00
Portfolio Size (N)	3.00
Simulation Period (days)	100.00
Stock A Volatility	0.0200
Stock B Volatility	0.0150
Stock C Volatility	0.0250

Portfolio Simulation

Generate random portfolio returns using Monte Carlo simulation with the defined volatility assumptions.

Monte Carlo methods are standard for portfolio risk analysis

```
# Generate random portfolio returns - full width
    for better readability
np.random.seed(42)
dates = pd.date_range('2024-01-01', periods=100,
    freq='D')

# Extract volatilities from assumptions
vol_a = assumptions_data.loc['Stock A Volatility',
    'Value']
vol_b = assumptions_data.loc['Stock B Volatility',
    'Value']
vol_c = assumptions_data.loc['Stock C Volatility',
    'Value']

# Simulate price paths
portfolio = pd.DataFrame({
    'Date': dates,
    'Stock_A': 100 * np.cumprod(1 +
        np.random.randn(100) * vol_a),
    'Stock_B': 100 * np.cumprod(1 +
        np.random.randn(100) * vol_b),
    'Stock_C': 100 * np.cumprod(1 +
        np.random.randn(100) * vol_c)
})

# Show first 5 rows as calculation table (blue
    background)
FinancialTable(portfolio.head().set_index('Date'),
    style='calculations')
```

Date	Stock_A	Stock_B	Stock_C
2024-01-01 00:00:00	100.99	97.88	100.89
2024-01-02 00:00:00	100.71	97.26	102.31
2024-01-03 00:00:00	102.02	96.76	105.08
2024-01-04 00:00:00	105.13	95.59	107.85
2024-01-05 00:00:00	104.63	95.36	104.13

Performance Analysis

Calculate key performance metrics for the Random Investment Strategy.

```
# Calculate portfolio statistics - displayed as
    Results (yellow background)
returns = portfolio[['Stock_A', 'Stock_B',
    'Stock_C']].pct_change().dropna()

statistics = pd.DataFrame({
    'Stock A': [
        returns['Stock_A'].mean() * 252,
        returns['Stock_A'].std() * np.sqrt(252),
        (returns['Stock_A'].mean() * 252) /
    (returns['Stock_A'].std() * np.sqrt(252)),
    ],
    'Stock B': [
        returns['Stock_B'].mean() * 252,
        returns['Stock_B'].std() * np.sqrt(252),
        (returns['Stock_B'].mean() * 252) /
    (returns['Stock_B'].std() * np.sqrt(252)),
    ],
    'Stock C': [
        returns['Stock_C'].mean() * 252,
        returns['Stock_C'].std() * np.sqrt(252),
        (returns['Stock_C'].mean() * 252) /
    (returns['Stock_C'].std() * np.sqrt(252)),
    ]
```

Key Metrics: - Annualized returns - Risk-adjusted ratios - Sharpe Ratio (Sharpe 1966)

```
}, index=[
    'Mean Return (annualized)',
    'Volatility (annualized)',
    'Sharpe Ratio'
])

# Display as Results table (yellow background for
    outputs)
FinancialTable(statistics, style='results')
```

	Stock A	Stock B	Stock C
Mean Return (annualized)	-0.5540	0.1392	0.3902
Volatility (annualized)	0.2892	0.2256	0.4323
Sharpe Ratio	-1.92	0.6171	0.9025

Visualization

Visual analysis of the Random Investment Strategy performance over the simulation period.

```
plt.figure(figsize=(10, 6))
plt.plot(portfolio['Date'], portfolio['Stock_A'],
    label='Stock A', linewidth=2)
plt.plot(portfolio['Date'], portfolio['Stock_B'],
    label='Stock B', linewidth=2)
plt.plot(portfolio['Date'], portfolio['Stock_C'],
    label='Stock C', linewidth=2)
plt.axhline(y=100, color='gray', linestyle='--',
    alpha=0.5, label='Initial Value')
plt.xlabel('Date')
plt.ylabel('Portfolio Value (Index: 100)')
plt.title('Random Investment Strategy Performance')
plt.legend()
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()
```

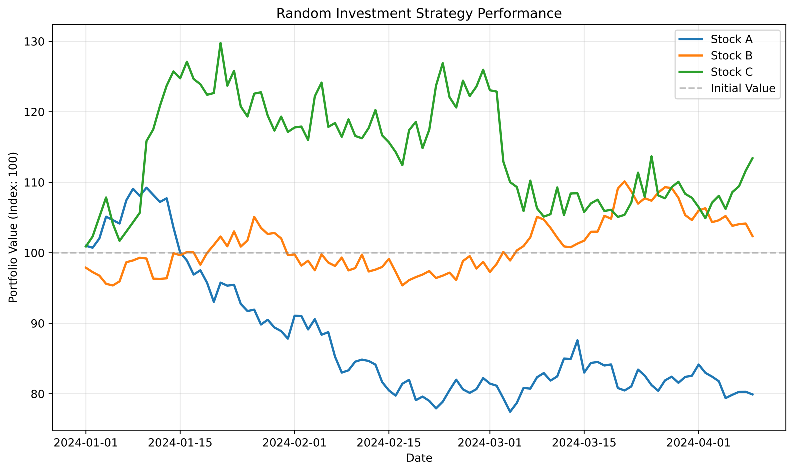

Figure 1: Random Investment Strategy - Simulated Performance

```
# Terminal-based plotting with plotext
import plotext as plt
from finmodel import PlotextChart

# Calculate returns for Stock A
stock_a_returns = returns['Stock_A'].values

# Create histogram
plt.clear_figure()
plt.hist(stock_a_returns, bins=20)
plt.theme('pro')
plt.xlabel('Daily Return')
plt.ylabel('Frequency')
plt.title('Stock A Returns Distribution (Random
    Strategy)')

# Render with PlotextChart wrapper
ansi_output = plt.build()
PlotextChart(ansi_output, size='medium')
```

Sharpe, William F. 1966. "Mutual Fund Performance." The Journal of Business 39 (1): 119–38.

Conclusion

This analysis demonstrates the Random Investment Strategy evaluation framework with:

1. Financial Modeling Standard tables - Assumptions (grey), Calculations (blue), Outputs (yellow)
2. Tufte-style layout - Margin annotations for context
3. Full-width code blocks - When horizontal space is needed (`#| column: page`)
4. Professional visualizations - Both matplotlib and plotext terminal plots
5. Citation management - Academic references with BibTeX (Markowitz 1952)

Key Findings

The Random Investment Strategy simulation shows: - Variable performance across randomly selected assets - Risk-adjusted returns (Sharpe ratios) indicate strategy effectiveness - Visual analysis reveals performance dispersion

See Docs/ for complete documentation on template features

References

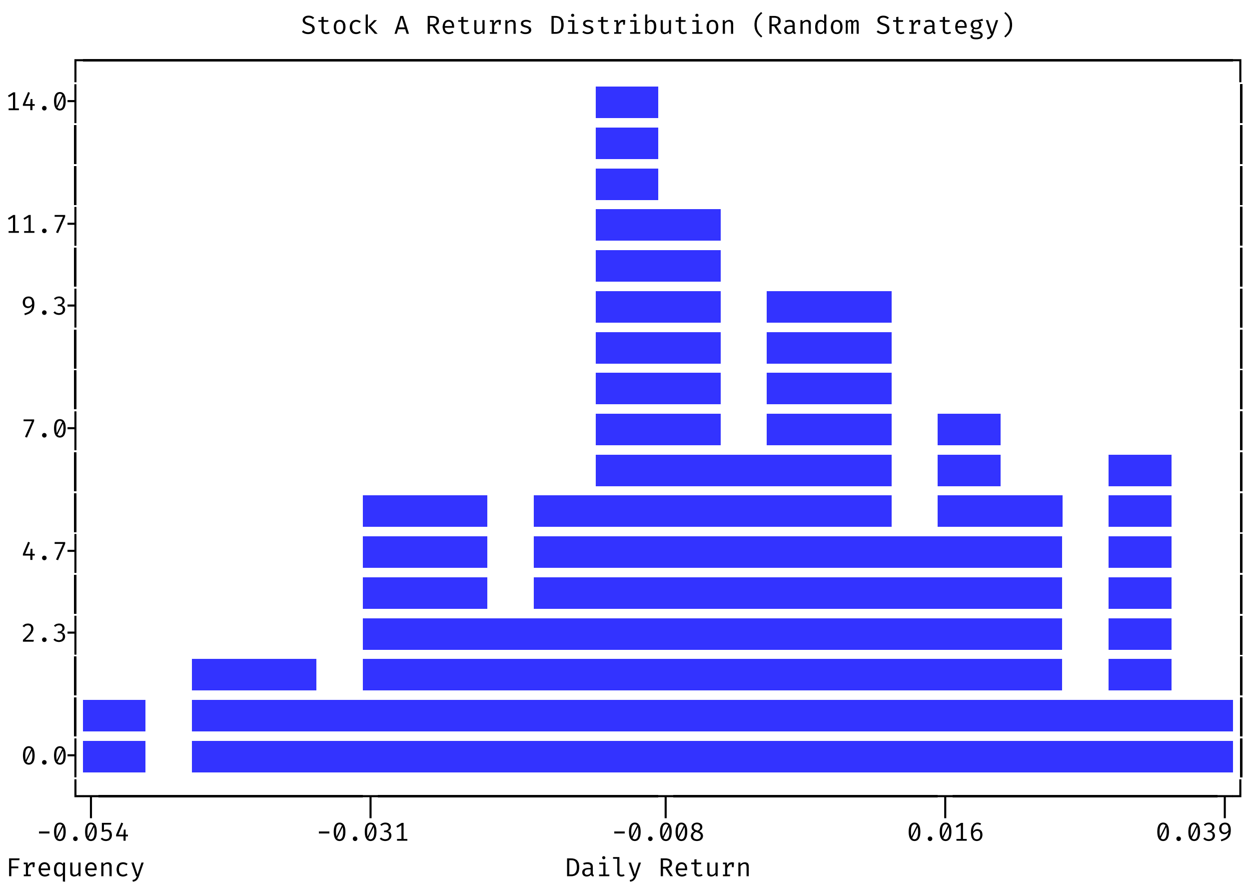

Figure 2: Returns Distribution - Terminal Plot (plotext)