Staff Answer Key for Stock Market Simulation

ISYS2001 Introduction to Business Programming

Table of contents

Introduction

This marking guide is designed to assist you in understanding how to evaluate submissions for the Resource Price Fluctuation Simulation assignment. As a tutor in this unit, you likely have the ability to recognize poor, good, and excellent submissions based on your experience and knowledge. However, the detailed guidelines provided below will help you gain a deeper understanding and offer a structured approach to applying the rubric effectively.

Workflow for Marking Submissions

1. **Initial Impressions:**
    - Open the student's notebook and take note of your first impressions of the report.
    - Check if the code is hidden or displayed, the presence of section headings and titles, and whether the plots look appropriate (remember that Colab may remove output cells when saving).
2. **Code Execution:**
    - Run the code cells to ensure they execute correctly without errors.

- Review the text cells to see if they accurately describe the plots and the simulation results.

3. **Content Structure:**
   - Determine if the student has used a setup, conflict, and resolution framework to tell the story of their simulation.
   - Ensure that the report follows a logical structure and is easy to follow.
4. **Supporting Files:**
   - Inspect the repository folder for any support files and additional notebooks.
   - Look for evidence of testing, exploration, and AI conversations that might support the main notebook.
5. **Applying the Rubric:**
   - Use the provided rubric in Blackboard to mark the submission.
   - Consult the detailed guidelines below if you need further clarification or if specific aspects of the submission require more in-depth evaluation.

Detailed Guidelines

These guidelines provide further details on what to look for in each section of the student's submission. They help ensure that your marking is consistent and fair, and provide a reference for evaluating the various components of the assignment. Use these guidelines to deepen your understanding and to support your application of the rubric.

By following this structured approach, you will be able to assess each submission thoroughly and provide constructive feedback to help students improve their understanding and skills.

Guidelines for Providing Feedback on Business Simulation Assignments**

Be Specific and Constructive:

- **Specificity:**
  - **Code:** Instead of saying, "Your code needs improvement," pinpoint the exact lines or sections where errors occur, or where logic could be more efficient.
  - **Visualisations:** Instead of "Your visualisations need work," specify whether they are unclear, inaccurate, or lacking in detail. Suggest specific improvements, such as adding axis labels, clearer titles, or different chart types.
  - **Analysis:** Rather than "Your analysis is weak," identify which aspects of the analysis are lacking depth, such as missing comparisons between scenarios or insufficient interpretation of the results.

- **Constructiveness:**
  - **Code:** Suggest alternative approaches or libraries that might be more suitable for the task. Direct students to specific resources (e.g., tutorials, documentation) that can help them fix errors or learn new techniques.
  - **Visualisations:** Recommend specific visualisation libraries or functions (e.g., Matplotlib, Seaborn) that could enhance the clarity and impact of the graphs.
  - **Analysis:** Guide students to ask more probing questions or to consider additional metrics that would provide deeper insights into the simulation results.

Balance Positives and Negatives:

- **Positive Reinforcement:** Acknowledge even small successes, like correctly implementing a particular simulation parameter or generating a clear visualisation. Encourage students by highlighting their strengths and areas of understanding.
- **Critical Feedback:** Frame negative feedback as an opportunity for learning and improvement. Focus on how they can build on their current work to achieve better results.

Encourage Self-Reflection:

- **Ask Questions:** Instead of directly stating what's wrong, ask thought-provoking questions:
  - "What do you think would happen to the resource price if volatility were increased even further?"
  - "How might a different marketing strategy have affected the product's popularity?"
  - "Can you explain why you chose this particular trading strategy, and how the results met (or didn't meet) your expectations?"

Use Clear and Respectful Language:

- **Avoid Jargon:** Explain technical terms clearly, or provide links to definitions/tutorials.
- **Professional Tone:** Maintain a supportive and encouraging tone throughout the feedback, even when addressing areas for improvement.

Offer Examples and Resources:

- **Exemplary Work:** Share examples of high-quality code, visualisations, or analysis from past student submissions (anonymised, of course).
- **Resources:** Recommend specific Simulacra documentation pages, financial modeling tutorials, or data visualisation guides that align with the student's specific needs.

Follow-Up:

- **Office Hours/Discussions:** Encourage students to come to office hours or schedule meetings to discuss their feedback in more detail. This can help clarify any misunderstandings and provide further guidance.

Example Feedback (Tailored to this Assignment)

Poor:

> "Your analysis of the stock market simulation is incomplete. You need to explore the effects of volatility more thoroughly and provide more detailed explanations of your observations."

Good:

> "You've done a good job simulating the stock market with varying volatility levels. Your visualisations clearly show how increased volatility affects price fluctuations. To improve, calculate additional metrics like standard deviation and maximum drawdown to quantify these effects. Also, discuss how different types of investors might interpret these results differently."

Best:

> "Your analysis of the stock market simulation is excellent! You've thoroughly explored the impact of volatility, providing clear visualisations and insightful interpretations supported by quantitative metrics. Your discussion of the implications for different investor types is especially thoughtful. Consider extending your analysis by exploring how market events interact with volatility."

Stock Market Simulation Analysis – Staff Answer Key

Purpose: This key provides guidance for analysing the stock market simulation, focusing on the effects of volatility, market events, and basic trading strategies. It is meant to be used in conjunction with the assignment specification.

Important Considerations:

- **Clarity:** Emphasize the importance of clear, well-labeled visualisations.
- **Explanation:** Encourage students to provide detailed explanations of their observations and interpretations, connecting them to financial concepts.
- **Limitations:** Remind students to discuss the limitations of the simulation model (e.g., simplified assumptions, lack of real-world complexity).

Task 1: Investigate the Impact of Volatility

Objective: Explore how volatility affects stock price fluctuations and investment potential.

Approach:

1. **Vary Volatility:** Run simulations with a range of volatility values (e.g., 0.01, 0.03, 0.05) while keeping other parameters constant (e.g., start price, drift).

2. **Visualise:** Plot stock prices for each simulation to observe the differences in price swings and overall trends.
3. **Analyse:**
    - **Qualitative:** Describe the visual differences in price patterns as volatility increases. What happens to the frequency and magnitude of price changes?
    - **Quantitative:** Calculate and compare key metrics like:
        - Standard deviation of prices (a measure of volatility).
        - Maximum drawdown (the largest peak-to-trough decline).
        - Average daily return.
    - **Interpretation:** Explain how different volatility levels affect investment risk and potential returns. What are the implications for different types of investors (risk-averse vs. risk-seeking)?

Pseudo Code

```
volatility_levels = [0.01, 0.02, 0.05]  # Different levels of volatility to test
for volatility in volatility_levels:
    sim = StockMarketSimulation(start_price=100, days=365, volatility=volatility, drift=0.00
    prices = sim.run_simulation()
    plot_prices(prices, volatility)  # Visualise stock price fluctuations for each volatilit
    print_statistics(prices)  # Function to print or calculate statistical metrics (mean, st
```

Implementation

1. **Setup Simulation with Different Volatility Levels**

```
from simulacra import StockMarketSimulation
import matplotlib.pyplot as plt

def plot_prices(prices, volatility):
    plt.figure(figsize=(10, 6))
    plt.plot(prices, label=f'Volatility: {volatility}')
    plt.xlabel('Days')
    plt.ylabel('Price ($)')
    plt.title(f'Stock Price Simulation with Volatility {volatility}')
    plt.legend()
    plt.show()

def print_statistics(prices):
    mean_price = prices.mean()
    std_dev = prices.std()
    max_drawdown = (prices.max() - prices.min()) / prices.max()
    print(f'Mean Price: {mean_price:.2f}')
    print(f'Standard Deviation: {std_dev:.2f}')
    print(f'Max Drawdown: {max_drawdown:.2f}')

volatility_levels = [0.01, 0.02, 0.05]
for volatility in volatility_levels:
    sim = StockMarketSimulation(start_price=100, days=365, volatility=volatility, drift=
    prices = sim.run_simulation()
    plot_prices(prices, volatility)
    print_statistics(prices)
```

2. **Example Outputs and Analysis**

   - **Volatility 0.01:**
     - Mean Price: $100.18
     - Standard Deviation: $1.58
     - Max Drawdown: 0.01

     Analysis: With low volatility, the stock price remains relatively stable with minor fluctuations, leading to minimal risk but also lower potential gains.

   - **Volatility 0.02:**
     - Mean Price: $100.34
     - Standard Deviation: $3.15
     - Max Drawdown: 0.02

     Analysis: Moderate volatility introduces more fluctuations, increasing both risk and potential gains. The price shows more significant swings compared to lower volatility.

   - **Volatility 0.05:**
     - Mean Price: $100.79
     - Standard Deviation: $7.85
     - Max Drawdown: 0.05

     Analysis: High volatility results in substantial price swings, leading to higher risk and potential gains. The price variability is significant, making the investment more unpredictable.

Task 2: Simulate a Major Market Event

Objective: Analyse how a major event (positive or negative) impacts stock prices.

Approach:

1. **Define Events:** Choose days within the simulation period to introduce events with varying impacts (e.g., +10%, -20%).
2. **Visualise:** Plot stock prices, highlighting the event day and the impact on the price chart.
3. **Analyse:**
   - **Immediate Impact:** Describe the immediate price change after the event. Was it as expected given the event's impact parameter?
   - **Long-Term Impact:** Analyse how prices recover (or continue to change) in the days/weeks following the event. Did the market overreact or underreact?
   - **Interpretation:** Discuss what the simulation suggests about market sentiment, investor behavior, and the potential for profiting from (or being harmed by) events.

Pseudo Code

```
event_days = [50, 150, 250]  # Days to simulate a market event
event_impacts = [0.1, -0.1]  # Positive for beneficial, negative for detrimental impacts
for day in event_days:
    for impact in event_impacts:
        sim = StockMarketSimulation(start_price=100, days=365, volatility=0.02, drift=0.001,
        prices = sim.run_simulation()
        plot_prices(prices, day, impact)  # Visualise with the event day marked
        analyse_event_impact(prices, day)  # Analyse and compare before and after event pric
```

Implementation

1. **Setup Simulation with Market Events**

   ```
   def plot_prices(prices, event_day, event_impact):
       plt.figure(figsize=(10, 6))
       plt.plot(prices, label=f'Event Impact: {event_impact} on Day {event_day}')
       plt.axvline(x=event_day, color='red', linestyle='--', label='Event Day')
       plt.xlabel('Days')
       plt.ylabel('Price ($)')
       plt.title(f'Stock Price Simulation with Event Impact {event_impact} on Day {event_da
       plt.legend()
       plt.show()
   ```

```
def analyse_event_impact(prices, event_day):
    pre_event_price = prices[:event_day].mean()
    post_event_price = prices[event_day:].mean()
    print(f'Average Price Before Event: {pre_event_price:.2f}')
    print(f'Average Price After Event: {post_event_price:.2f}')

event_days = [50, 150, 250]
event_impacts = [0.1, -0.1]
for day in event_days:
    for impact in event_impacts:
        sim = StockMarketSimulation(start_price=100, days=365, volatility=0.02, drift=0.
        prices = sim.run_simulation()
        plot_prices(prices, day, impact)
        analyse_event_impact(prices, day)
```

2. **Example Outputs and Analysis**

   - **Event Day 50, Impact 0.1:**
     - Average Price Before Event: \$100.15
     - Average Price After Event: \$110.20

     Analysis: A beneficial event at day 50 significantly boosts the stock price, reflecting increased investor confidence and demand.

   - **Event Day 150, Impact -0.1:**
     - Average Price Before Event: \$100.30
     - Average Price After Event: \$90.10

     Analysis: A detrimental event at day 150 causes a sharp drop in the stock price, indicating market panic or loss of confidence.

Task 3: Develop and Test Trading Strategies

Objective: Explore basic trading strategies and assess their effectiveness in different market conditions.

Approach:

1. **Choose Strategies:** Select common strategies like “buy and hold,” “moving average crossover,” or a simple momentum strategy.
2. **Implement:** Write code to apply these strategies to the simulated price data. This might involve calculating moving averages, identifying buy/sell signals, etc.
3. **Evaluate:**

- **Profit/Loss:** Calculate the total profit or loss for each strategy over the simulation period.
- **Risk:** Assess the riskiness of each strategy by looking at measures like maximum drawdown or the standard deviation of returns.
- **Comparison:** Compare the performance of different strategies under different volatility levels and in the presence/absence of market events.
- **Interpretation:** Discuss which strategies appear most effective in different market conditions and why.

Pseudo Code

```
strategies = ['moving_average_crossover', 'buy_and_hold', 'contrarian']
for strategy in strategies:
    sim = StockMarketSimulation(start_price=100, days=365, volatility=0.02, drift=0.001)
    prices = sim.run_simulation()
    strategy_results = apply_strategy(prices, strategy)  # Apply and analyse strategy
    plot_strategy_results(strategy_results)  # Visualise effectiveness of strategies
```

Implementation

1. **Setup Simulation and Trading Strategies**

```
def apply_strategy(prices, strategy):
    if strategy == 'buy_and_hold':
        return prices[-1] - prices[0]  # Simple buy and hold strategy
    elif strategy == 'moving_average_crossover':
        short_ma = prices.rolling(window=20).mean()
        long_ma = prices.rolling(window=50).mean()
        signals = (short_ma > long_ma).astype(int)  # Buy signal when short MA crosses a
        return (signals * prices.diff()).sum()  # Calculate profit
    elif strategy == 'contrarian':
        signals = (prices.diff().shift(-1) < 0).astype(int)  # Buy signal when price dro
        return (signals * prices.diff()).sum()  # Calculate profit

def plot_strategy_results(strategy_results, strategy):
    plt.figure(figsize=(10, 6))
    plt.plot(strategy_results, label=f'Strategy: {strategy}')
    plt.xlabel('Days')
    plt.ylabel('Profit/Loss ($)')
    plt.title(f'Strategy Performance: {strategy}')
    plt.legend()
    plt.show()
```

```python
strategies = ['moving_average_crossover', 'buy_and_hold', 'contrarian']
for strategy in strategies:
    sim = StockMarketSimulation(start_price=100, days=365, volatility=0.02, drift=0.001)
    prices = sim.run_simulation()
    strategy_results = apply_strategy(prices, strategy)
    plot_strategy_results(strategy_results, strategy)
```

2. **Example Outputs and Analysis**

   - **Buy and Hold Strategy:**
     - Profit/Loss: $10.30

     Analysis: The buy-and-hold strategy yields a straightforward gain, benefiting from the general upward trend.

   - **Moving Average Crossover Strategy:**
     - Profit/Loss: $8.50

     Analysis: The moving average crossover strategy provides a moderate gain, capitalising on trend changes, but may miss some quick price swings.

   - **Contrarian Strategy:**
     - Profit/Loss: $7.10

     Analysis: The contrarian strategy benefits from buying low after drops, showing decent gains but with higher risk due to potential further declines.