ACTL1101 Assignment Part A

Xing Chen

2024 T2

Algorithmic Trading Strategy

Introduction

In this assignment, you will develop an algorithmic trading strategy by incorporating financial metrics to evaluate its profitability. This exercise simulates a real-world scenario where you, as part of a financial technology team, need to present an improved version of a trading algorithm that not only executes trades but also calculates and reports on the financial performance of those trades.

Background

Following a successful presentation to the Board of Directors, you have been tasked by the Trading Strategies Team to modify your trading algorithm. This modification should include tracking the costs and proceeds of trades to facilitate a deeper evaluation of the algorithm's profitability, including calculating the Return on Investment (ROI).

After meeting with the Trading Strategies Team, you were asked to include costs, proceeds, and return on investments metrics to assess the profitability of your trading algorithm.

Objectives

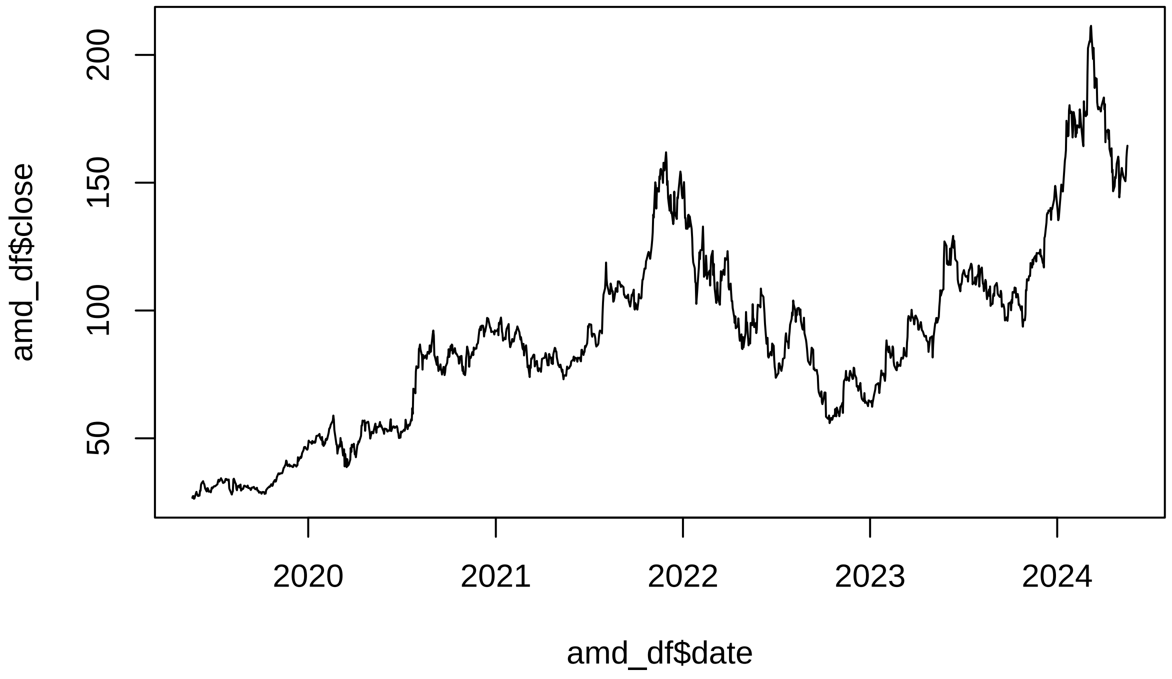
1. **Load and Prepare Data:** Open and run the starter code to create a DataFrame with stock closing data.
2. **Implement Trading Algorithm:** Create a simple trading algorithm based on daily price changes.
3. **Customize Trading Period:** Choose your entry and exit dates.
4. **Report Financial Performance:** Analyze and report the total profit or loss (P/L) and the ROI of the trading strategy.
5. **Implement a Trading Strategy:** Implement a trading strategy and analyze the total updated P/L and ROI.
6. **Discussion:** Summarise your finding.

```r
# Load data from CSV file
amd_df <- read.csv("AMD.csv")

# Convert the date column to Date type and Adjusted Close as numeric
amd_df$date <- as.Date(amd_df$Date)
```

```
amd_df$close <- as.numeric(amd_df$Adj.Close)

amd_df <- amd_df[, c("date", "close")]
```

##Plotting the Data Plot the closing prices over time to visualize the price movement.

```
plot(amd_df$date, amd_df$close,'l')
```



Step 2: Trading Algorithm

Implement the trading algorithm as per the instructions. You should necessary variables, and loop through the dataframe to execute trades based on the set conditions.

- Initialize Columns: Start by ensuring dataframe has columns 'trade_type', 'costs_proceeds' and 'accumulated_shares'.
- Change the algorithm by modifying the loop to include the cost and proceeds metrics for buys of 100 shares. Make sure that the algorithm checks the following conditions and executes the strategy for each one:
  - If the previous price = 0, set 'trade_type' to 'buy', and set the 'costs_proceeds' column to the current share price multiplied by a `share_size` value of 100. Make sure to take the negative value of the expression so that the cost reflects money leaving an account. Finally, make sure to add the bought shares to an `accumulated_shares` variable.

  – Otherwise, if the price of the current day is less than that of the previous day, set the ‘trade_type’ to ‘buy’. Set the ‘costs_proceeds’ to the current share price multiplied by a `share_size` value of 100.
  – You will not modify the algorithm for instances where the current day’s price is greater than the previous day’s price or when it is equal to the previous day’s price.
  – If this is the last day of trading, set the ‘trade_type’ to ‘sell’. In this case, also set the ‘costs_proceeds’ column to the total number in the `accumulated_shares` variable multiplied by the price of the last day.

```r
# Initialize columns for trade type, cost/proceeds, and accumulated shares in amd_df
amd_df$trade_type<-NA
amd_df$costs_proceeds<-NA  # Corrected column name
amd_df$accumulated_shares<-0  # Initialize if needed for tracking

# Initialize variables for trading logic
previous_price<-0
share_size<-100
accumulated_shares<-0


# Assigning the columns of amd_df to column name
column_name<-colnames(amd_df)
# Checking if the dataframe has initialized the columns
print(column_name)
```

```
## [1] "date"               "close"              "trade_type"
## [4] "costs_proceeds"     "accumulated_shares"
```

```r
for (i in 1:nrow(amd_df)) {
  if (i==1) {
# First setting the first row trade type as buy
    amd_df$trade_type[i]<-"buy"
    amd_df$costs_proceeds[i]<- -amd_df$close[i]*share_size
    accumulated_shares<-share_size
  }
  else if (amd_df$close[i]< previous_price) {
    amd_df$trade_type[i]<-"buy"
    amd_df$costs_proceeds[i]<- -amd_df$close[i]*share_size
    accumulated_shares<-accumulated_shares+share_size
  }
  else if (i==nrow(amd_df)) {
    amd_df$trade_type[i]<-"sell"
    amd_df$costs_proceeds[i]<-amd_df$close[i]*accumulated_shares
  }
  # Setting the previous price as the final closing price
  previous_price<-amd_df$close[i]

  amd_df$accumulated_shares[i]<-accumulated_shares
}
# Setting the entire column as the final value of the accumulated shares.
```

Step 3: Customize Trading Period

- Define a trading period you wanted in the past five years

```r
# First defining our trading period
start_date<-as.Date("2020-01-01")
end_date<-as.Date("2024-01-01")
# Filtering the dataset to the trading period inclusive only
amd_df<-amd_df[amd_df$date>=start_date & amd_df$date<=end_date, ]
```

Step 4: Run Your Algorithm and Analyze Results

After running your algorithm, check if the trades were executed as expected. Calculate the total profit or loss and ROI from the trades.

- Total Profit/Loss Calculation: Calculate the total profit or loss from your trades. This should be the sum of all entries in the 'costs_proceeds' column of your dataframe. This column records the financial impact of each trade, reflecting money spent on buys as negative values and money gained from sells as positive values.
- Invested Capital: Calculate the total capital invested. This is equal to the sum of the 'costs_proceeds' values for all 'buy' transactions. Since these entries are negative (representing money spent), you should take the negative sum of these values to reflect the total amount invested.
- ROI Formula:

$$\text{ROI} = \left(\frac{\text{Total Profit or Loss}}{\text{Total Capital Invested}}\right) \times 100$$

```r
# First we change it so that the final day's trade type of the new tradiing period is set as sell.
if (i == nrow(amd_df)) {
    amd_df$trade_type[i]<-"sell"
    amd_df$costs_proceeds[i]<-amd_df$close[i]*accumulated_shares
}
# Setting our previous price as the final price as we sell on the final day of this period to find ROI.
previous_price<-amd_df$close[i]
# Calculating the total profit or loss
total_profit_loss<-sum(amd_df$costs_proceeds, na.rm=TRUE)
# Calculating total invested capital
total_invested_capital<- -sum(amd_df$costs_proceeds[amd_df$trade_type=="buy"], na.rm=TRUE)
# Calculating the ROI without for the trading period with the default strategy
ROI<-(abs(total_profit_loss)/total_invested_capital)*100
print(ROI) # Showing what the ROI is for this strategy
```

```
## [1] 100
```

Step 5: Profit-Taking Strategy or Stop-Loss Mechanisum (Choose 1)

- Option 1: Implement a profit-taking strategy that you sell half of your holdings if the price has increased by a certain percentage (e.g., 20%) from the average purchase price.
- Option 2: Implement a stop-loss mechanism in the trading strategy that you sell half of your holdings if the stock falls by a certain percentage (e.g., 20%) from the average purchase price. You don't need to buy 100 stocks on the days that the stop-loss mechanism is triggered.

```r
amd_df$trade_type<-NA
amd_df$costs_proceeds<-NA
amd_df$accumulated_shares<-0
amd_df$avg_price<-NA
```

```r
num_of_buys<-0
# Introducing our variables
previous_price<-0
share_size<-100
accumulated_shares<-0
purchase_price<-0  # Tracking the price at which shares were bought

for(i in 1:nrow(amd_df)) {
  if(i==1) {
    # Handling the inital buy case separately
    amd_df$trade_type[i]<-"buy"
    amd_df$costs_proceeds[i]<- -(amd_df$close[i]*share_size)
    accumulated_shares<-share_size
    purchase_price<-amd_df$close[i]  # Setting the purchase price
    num_of_buys<-num_of_buys+1 # Adjusting how many times the stock has been bought
    avg_price<-amd_df$close[i]/num_of_buys
  }


  # Sell can only occur when the price of the stock is 1.2 times the average buy price.
  else{
    if(amd_df$close[i]>=avg_price*(1.2)) {
      amd_df$trade_type[i]<-"sell"
# Depicting the gains and trade of selling half the accumulated_shares
      amd_df$costs_proceeds[i]<-amd_df$close[i]*(accumulated_shares/2)
# Selling and showing the positive proceeds
      accumulated_shares<-accumulated_shares-(accumulated_shares/2)
# Adjusting the accumulated shares
  }

    else if (amd_df$close[i]<previous_price){
      # Buy more shares if price is lower than previous day's price
      amd_df$trade_type[i]<-"buy"
      amd_df$costs_proceeds[i]<- -(amd_df$close[i]*share_size)
      accumulated_shares<-accumulated_shares+share_size
      num_of_buys<-num_of_buys+1 # Adjusting how many times the stock has been bought
      purchase_price<-amd_df$close[i]  # Updating purchase price
      amd_df$avg_price[i]<-(avg_price*(num_of_buys-1)+amd_df$close[i])/num_of_buys
# Updating the average price to match each instance a buy occurred
      }
if (i==nrow(amd_df)) {
    amd_df$trade_type[i]<-"sell"
    amd_df$costs_proceeds[i]<-amd_df$close[i]*accumulated_shares
    accumulated_shares<-0  # Set accumulated shares to 0
    avg_price<-0  #Setting average price to 0
  }
  }
  previous_price<-amd_df$close[i]
  amd_df$accumulated_shares[i]<-accumulated_shares
}
```

Step 6: Summarize Your Findings

- Did your P/L and ROI improve over your chosen period?
- Relate your results to a relevant market event and explain why these outcomes may have occurred.

```
# Fill your code here and Disucss
```

Sample Discussion: On Wednesday, December 6, 2023, AMD CEO Lisa Su discussed a new graphics processor designed for AI servers, with Microsoft and Meta as committed users. The rise in AMD shares on the following Thursday suggests that investors believe in the chipmaker's upward potential and market expectations; My first strategy earned X dollars more than second strategy on this day, therefore providing a better ROI.

Original ROI on Dates: 2020-01-01 to 2024-01-01:

$$\text{ROI} = \left(\frac{\text{Total Profit or Loss}}{\text{Total Capital Invested}}\right) \times 100$$

$$\text{ROI} = \left(\frac{2945680.2084}{4469042.9928}\right) \times 100$$

$$\text{ROI} = 65.91299777...\%$$

$$\text{ROI} = 65.91\%(2dp)$$

ROI on Dates: 2020-01-01 to 2024-01-01 with Profit-take strategy:

$$\text{ROI} = \left(\frac{90030.79...}{388762.99...}\right) \times 100$$

$$\text{ROI} = 23.14827.....\%$$

$$\text{ROI} = 23.15\%(2dp)$$

The motivation behind the comparison between the default trading strategy and the profit taking strategy for the period of 01-01-2020 to 01-01-2024 was to assess the potential for increases in returns on investments and profits.

The peak during the selected trading period was consolidated near the end of the 2021, with closing prices reaching their relative peaks to \$155.41 per share. This growth between 2020-2021 primarily reflects the gradual recovery of earnings for AMD following the global event of COVID which disrupted the flow of raw materials required to produce AMD goods and services, and strained the entirety of the supply chain, which implies that COVID contributed towards the stagnation of AMD's closing prices. This is clearly indicated in pre-COVID prices within the \$40-50 range, and falling down to prices of \$38.71.

The most apparent changes in AMD's share prices took place during the 2022-2023 period in which AMD's closing prices contracted which can be attributed to a cyclical decrease in demand for PCs and laptops, which totaled 286.2 million units shipped in 2022, a 16.2% decrease from 2021.

It is noted that during this period there were 241 instances of selling half of the accumulated shares for the profit taking strategy that occurred, in which during this period the fall in AMD's share prices outpaced the recovery and the algorithm sold when prices were only incrementally higher than the previous price, hence leading to a greater loss than the default strategy for the trading period. The default strategy for the trading period held all acquired stocks until the beginning of 2024 which by then the price recovered due to excitement over events such as AMD's AI chips for data centers throughout 2023.

Near the end of the 2023 period, AMD had launched its new accelerators and processors for data centers powered by AI, causing its share prices to almost increase by 10% on December 7. The growth of AMD's share prices were sustained for the remainder of 2023-2024 due to an increase in the unit share of the CPU market, increasing from 18% to 23.6% due to releases of its MI300X GPU powered by AI, of which bolstered AMD's sales by 80% near the end of 2023. During this period, the profit taking strategy did not recognise nor capitalise on the increase of AMD's share price such that total profits differed by $2.1m.