title: ACTL1101 Assignment Part A author: Michael Liu date: 2024 T2 output: html_document: df_print: paged pdf_document: default —

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.

Instructions

Step 1: Data Loading

Start by running the provided code cells in the "Data Loading" section to generate a DataFrame containing AMD stock closing data. This will serve as the basis for your trading decisions. First, create a data frame named `amd_df` with the given closing prices and corresponding dates.

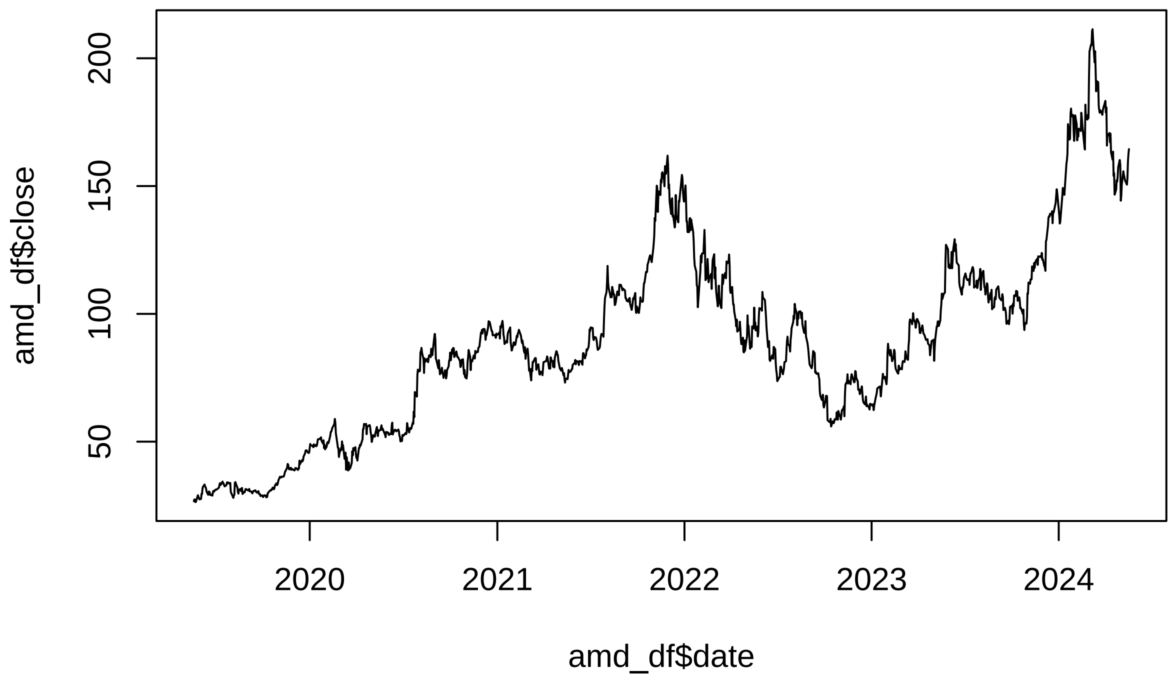
```r
# Load data from CSV file
amd_df <- read.csv("AMD.csv")

# 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.

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



Step 2: Trading Algorithm

Implement the trading algorithm as per the instructions. You should initialize 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
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
```

```
##     intersect, setdiff, setequal, union
```

```
# Initialize columns for trade type, cost/proceeds, and accumulated shares in amd_df
amd_df$trade_type <- NA
amd_df$costs_proceeds <- 0  # 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

for (i in 1:nrow(amd_df)) {
  if (i == 1) {
    amd_df$accumulated_shares[i] <- 0
  } else {
    amd_df$accumulated_shares[i] <- amd_df$accumulated_shares[i-1]
  }
  if (i == 1 || (amd_df$close[i] < amd_df$close[i-1])) {
    amd_df$trade_type[i] <- 'buy'
    amd_df$costs_proceeds[i] <- -amd_df$close[i]*share_size
    amd_df$accumulated_shares[i] <- amd_df$accumulated_shares[i] + share_size
  } else if (i == nrow(amd_df)) {
    amd_df$trade_type[i] <- 'sell'
    amd_df$costs_proceeds[i] <- amd_df$accumulated_shares[i]*(amd_df$close[i])
  }
}

if ("cost_proceeds" %in% colnames(amd_df)) {
  amd_df <- amd_df %>% select(-cost_proceeds)
}
```

Step 3: Customize Trading Period

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

```
#2023-01-03 - 2024-05-17
#defined by rows 914 to 1259


start_date <- as.Date("2023-01-03")
end_date <- as.Date("2024-05-17")
```

#Explain reasoning for choosing this time period Amidst the backdrop of the post-covid recovery and subsequent correction of overly expansiory macroeconomic policy, the start of 2023 marked a recovery of the stock market fueled by the progress in AI. Central to this are companies such as NVIDIA and AMD who lead R&D in producing the necessary hardware for AI processes. As such, with predictable wider economic conditions and company specific progress, I thought it would be an interesting period to conduct the trading strategy.

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:

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

```
# This is excel strategy trading


#Calculate for the customized trading period
#Re-intialising the variables
amd_df$trade_type <- NA
amd_df$costs_proceeds <- 0
amd_df$accumulated_shares <- 0

previous_price <- 0
share_size <- 100
accumulated_shares <- 0


  for (i in 914:nrow(amd_df)) {
    if (i == 914) {
      amd_df$accumulated_shares[i] <- 0
    } else {
      amd_df$accumulated_shares[i] <- amd_df$accumulated_shares[i-1]
    }
    if (i == 914 || (amd_df$close[i] < amd_df$close[i-1])) {
      amd_df$trade_type[i] <- 'buy'
      amd_df$costs_proceeds[i] <- -amd_df$close[i]*share_size
      amd_df$accumulated_shares[i] <- amd_df$accumulated_shares[i] + share_size
    }

    if (i == nrow(amd_df)) {
      amd_df$trade_type[i] <- 'sell'
      amd_df$costs_proceeds[i] <- amd_df$accumulated_shares[i]*(amd_df$close[i])
    }
  }

  profit <- sum(amd_df$costs_proceeds[914:i])
  if (profit >= 0) {
    print(sprintf("Profit of $%.0f", profit))
  } else if (profit < 0) {
    print(sprintf("Loss of %.0f", profit))
  }
```

```
## [1] "Profit of $714288"
```

```
  Invested_Capital <- -sum(amd_df$costs_proceeds[914:i-1])
  ROI <- profit/Invested_Capital*100

  print(sprintf("The ROI was %.2f%%", ROI))
```

```
## [1] "The ROI was 35.72%"
```

My code constructs a for loop from the dates of 2023-01-03 to 2024-05-17. I Initialize variables and then set the logic to buy whenever the price is lower than the day before. Then I sum the cost proceeds excluding the last day to figure my invested capital and sum the whole thing to find profit.

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
#logic is sell 50% if price is 1.2x the average purchase price and selling takes priority over buying
#essentially if day price is 1.2 x sum(1:i)/accumulated shares then sell 0.5xaccumulated shares
amd_df$trade_type <- NA
amd_df$costs_proceeds <- 0
amd_df$accumulated_shares <- 0

previous_price <- 0
share_size <- 100
accumulated_shares <- 0
sum_cost <- 0
avg_cost <- 0
Invested_Capital <- 0
sold_tracker <- 0
start_date <- as.Date("2023-01-03")
end_date <- as.Date("2024-05-17")

if ("cost_proceeds" %in% colnames(amd_df)) {
  amd_df <- amd_df %>% select(-cost_proceeds)
}

  for (i in 914:nrow(amd_df)) {
    if (i == 914) {
      amd_df$accumulated_shares[i] <- 0
    } else {
      amd_df$accumulated_shares[i] <- amd_df$accumulated_shares[i-1]
    }
    if (i == 914 || (amd_df$close[i] < amd_df$close[i-1])) {
      amd_df$trade_type[i] <- 'buy'
      amd_df$costs_proceeds[i] <- -amd_df$close[i]*share_size
      amd_df$accumulated_shares[i] <- amd_df$accumulated_shares[i] + share_size
      sum_cost <- sum_cost - amd_df$costs_proceeds[i]
    }

    if (i == nrow(amd_df)) {
      amd_df$trade_type[i] <- 'sell'
      amd_df$costs_proceeds[i] <- amd_df$accumulated_shares[i]*(amd_df$close[i])
      amd_df$accumulated_shares[i] <- 0
    }

    avg_cost <- sum_cost/amd_df$accumulated_shares[i]
    if (amd_df$close[i] > 1.2*avg_cost ) {
      amd_df$trade_type[i] <- 'sell'
      sum_cost <- sum_cost - amd_df$accumulated_shares[i-1]/2*avg_cost
      amd_df$costs_proceeds[i] <- amd_df$accumulated_shares[i-1]/2 * amd_df$close[i]
      amd_df$accumulated_shares[i] <- amd_df$accumulated_shares[i-1]/2

    }
  }

for (i in 914:nrow(amd_df)) {
  if (amd_df$costs_proceeds[i] > 0 && i != nrow(amd_df)) {
    sold_tracker <- sold_tracker + amd_df$costs_proceeds[i]
  }
}

profit <- sum(amd_df$costs_proceeds[914:i])
  if (profit >= 0) {
    print(sprintf("Profit of $%.0f", profit))
  } else if (profit < 0) {
    print(sprintf("Loss of %.0f", profit))
  }
```

```
## [1] "Profit of $380783"
```

```
  Invested_Capital <- -sum(amd_df$costs_proceeds[914:i-1]) + sold_tracker
  ROI <- profit/Invested_Capital*100

  print(sprintf("The ROI was %.2f%%", ROI))
```

```
## [1] "The ROI was 19.30%"
```

My code constructs a for loop from the dates of 2023-01-03 to 2024-05-17. I Initialize variables and then set the logic to buy whenever the price is lower than the day before. However, in this strategy I keep track of avg_costs and sold_tracker to figure out invested capital later since we are selling whenever it 1.2x the average cost. Then I use similar strategies to determine ROI and profit.

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.

```
#P/L and ROI for both cases have been calculated in steps 4 and 5.
#Discussion is in the text box below
```

Discussion: The P/L and ROI with the new trading strategy worsened relative to the original strategy. Given the same period specified in part 3, the original trading strategy recorded P/L of $714288 and ROI of 35.72%, whilst the new trading strategy recorded a P/L $380783 and ROI of 19.30%. Though, the original trading strategy recorded a better profit and ROI, this may have been skewed due to certain market events. Namely, the AI boom which has placed increased demand for AI-related technology. Specific products such as M1300X combined the with the general rally of the semiconductor industry during this period catalysed this growth. As such, since the second strategy takes greater risk management and profit taking protocol which would likely lead to higher ROI over the long term when factoring in downswings. In this period due to exponential growth has recorded less profit than simply buying more and selling at the end which is in essence what the first strategy does.