RICE UNIVERSITY

Mini Project 2

STAT 682 - Fall 2024

Authors:

Dolese, Ryker

Fang, Judy

Hao, Sirui

Hecht, Martin

Kumar, Krish

Thetford, Jackson

Professors:

Dr. Michael D Jackson

Max Lee

Data Overview

Data Sources

The data used in this analysis comes from multiple sources to provide a detailed picture of FX G10 currency markets, relevant economic indicators, and macroeconomic fundamentals. The primary data sources include:

- **Bloomberg (BBG)**: Provides FX G10 currency exchange rates and related indicators. Key data on currency pairs such as EURUSD, GBPUSD, and USDJPY were extracted for calculating trend and value metrics.
- **International Monetary Fund (IMF) World Economic Outlook (WEO)**: Contains essential economic indicators, including Purchasing Power Parity (PPP), serving as a baseline for value metrics. The WEO data captures long-term economic trends affecting currency valuation.
- **Federal Reserve Economic Data (FRED)**: Supplies interest rate data across G10 countries, crucial for assessing the cost of carry for each currency. FRED provides consistent, reliable data for interest rate comparisons.
- **Other National Economic Data Sources** (e.g., OECD): Supplemental economic data was used to fill gaps in current account balance (CAB) or GDP when necessary.

CSV Files Used (all available at this Google Drive link):

- **accountbalanceg10.csv**: Contains current account balance data for G10 countries.
- **current_account_balance.csv**: Provides detailed current account balance information, used to assess economic stability and trade flows within G10 currencies.
- **GDP_1.csv**: Includes GDP data across G10 countries, serving as an additional fundamental indicator.
- **MP2.Data.From.BBG.xlsx**: This Excel file includes currency pair data for FX G10, necessary for constructing the trend-value indicator (TVI).
- **switzerlandCAB.csv**: Contains specific current account balance data for Switzerland, ensuring thorough data coverage for all G10 countries.
- **WEOOct2024all.xlsx**: A comprehensive dataset from the IMF's World Economic Outlook, including PPP and other critical economic indicators.

Data Frequency

The datasets used in this study feature various frequencies:

- **FX G10 Currency Rates**: Daily frequency, providing high-resolution data for calculating the trend indicators (momentum, persistence, and price positioning) for each currency pair.
- **Interest Rates from FRED**: Monthly frequency, capturing broader changes in interest rates across G10 countries. Monthly data is sufficiently granular to capture trends relevant to currency valuation.
- **WEO Data from IMF**: Annual frequency, reflecting slower-moving economic fundamentals like PPP and GDP, aligning with long-term currency valuation assessments.
- **Current Account Balances (CAB) and GDP Data**: Mixed frequencies depending on the source; CAB data generally follows a quarterly or annual update cycle, while GDP updates can range from quarterly to annual. This variability was managed by aligning dates for comparability in analysis.

Data Coverage

The data covers major G10 currencies and includes:

- **Currencies in Focus**: Key currency pairs from the G10, including EURUSD, GBPUSD, USDJPY, USDCHF, AUDUSD, NZDUSD, USDCAD, USDNOK, and USDSEK. This selection provides a balanced view of currencies with diverse economic backgrounds and interest rate environments.
- **Economic Indicators**: A comprehensive set of economic indicators, including GDP, interest rates, CAB, and PPP for each G10 country. These indicators allow for a multidimensional approach to currency valuation, supporting the construction of the trend-value indicator (TVI) by capturing both fundamental and quantitative factors.
- **Time Span**: The dataset spans several decades, with WEO data dating back to the 1980s for annual PPP values, and FX rates covering multiple business cycles.

Value Trading Strategy Case Study

As we discussed in class on October 10, Value trading is a known quantity in both discretionary and systematic styles alike; Warren Buffet is perhaps best known for his approach to buying long-term investments. We are going to work through an example of this in the systematic format and build a case-study of a "quant-trading" strategy in the context of the factor/risk premia work we were doing.

Step 1

Download Data

Go to the IMF website and navigate to the WEO data. Download all data so you can see the full file by all countries; it should look something like the below. Spend a little time and describe the data to us. Thinking back to the class on data methods, what is the most effective way to communicate the intent, use, as well as flaws in the series. One of the most valued skills in a job market where there is so much technical fluency is how to actually present information in a meaningful, concise way. I would think about how to give the fullest description without leaning on 100 histograms and pages of text. Put your executive hats on!

Answer:

We wrote python code for the data loading, cleaning and analysis. For the code refer to A.1

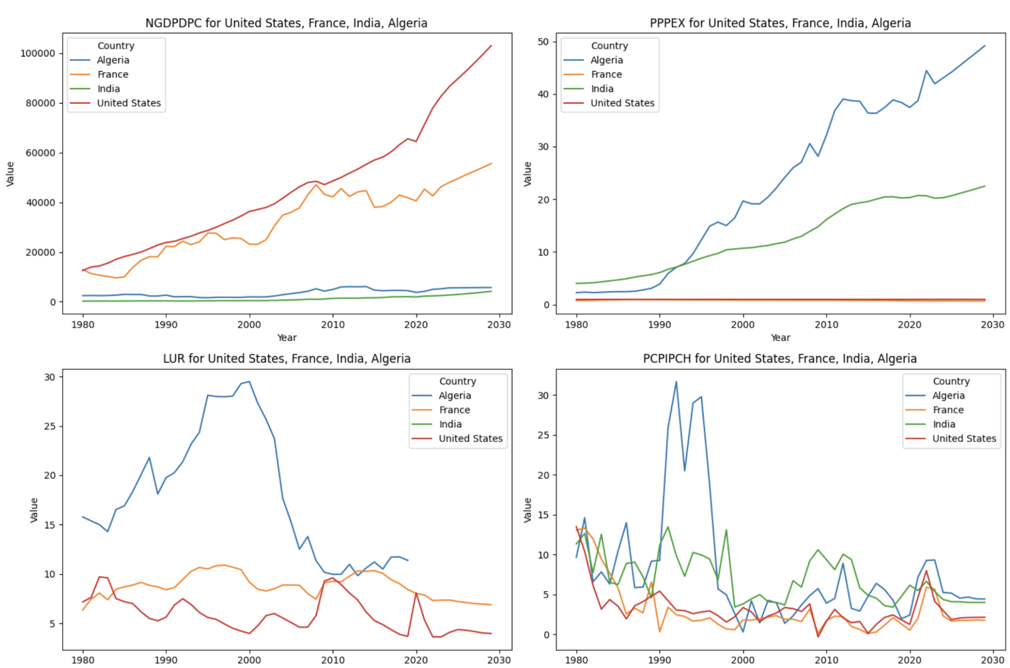

Figure 1: Economic Conditions for Four Chosen Countries

Figure 1 gives us an idea of the type of data we are working with. We have many countries and several economic indicators from 1980-Present. We also have projections for years beyond 2024. While we've only chosen 4 countries, we can see how economic conditions vary widely between 'developed' countries and 'developing' countries. NGDPDPC is GDP per capita, as expressed in USD and we see how the United States tends to significantly outperform the likes of India and Algeria, which makes sense. Though the U.S. and France we similar in terms of GDP per capita until 2010 or so, the U.S. seems to have separated themselves in that regard.

We can also notice that the underdeveloped countries tend to have greater implied PPP (PPPEX), greater unemployment (LUR), and greater yearly inflation (PCPIPCH). This suggests more robust economic conditions for countries like the U.S. and France.

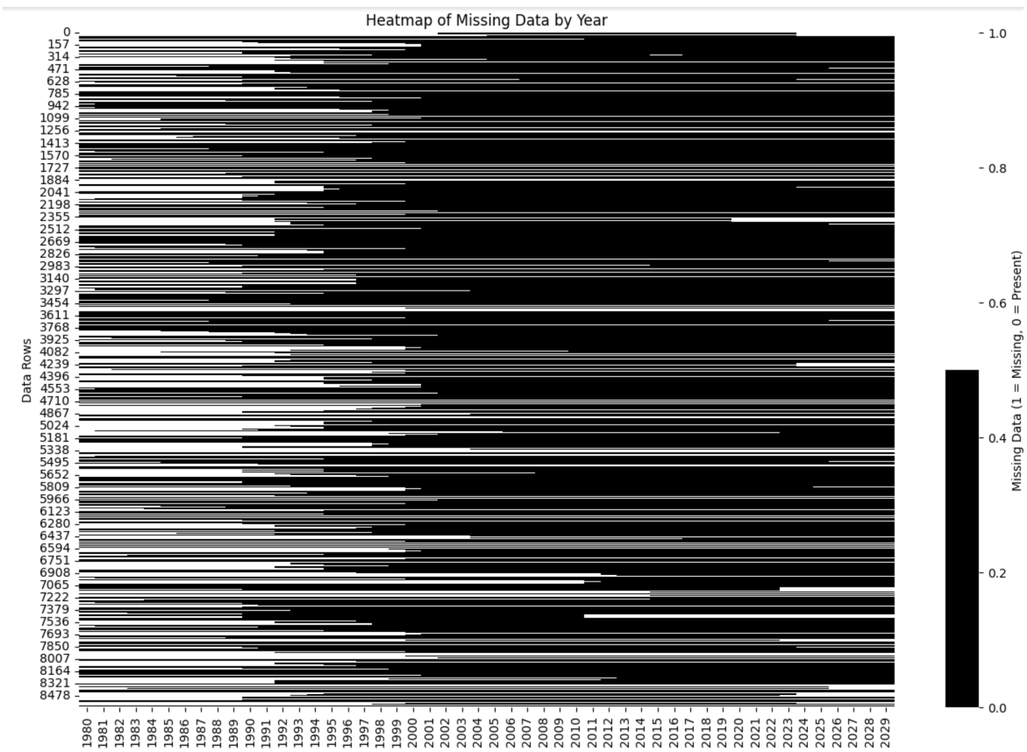

Figure 2: Missing Values

When analyzing the data, it's important to consider what information we have and don't have. For this reason, we have decided to plot a heatmap of missing values in the dataframe in Figure 2, ordered by year. Here, an obvious pattern emerges: there is a lot of missing data prior to year 2000, but not many missing values thereafter. This suggests our analysis might be better off focusing on more recent years then previous ones. Moreover, missing values tend to cluster together vertically as well, meaning when some countries have missing values for one economic indicator, they are likely to be missing many more as well.

Step 2

The subset we are interested in is the Implied PPP. Condense the filter and create a time series that is usable for a proof of concept.

We run the Python code for the analysis in this step. See A.2 for the full details of the code we used.

1. Focus on Implied PPP

At its core, what is Purchasing Power Parity? In short form, what would a trading signal on this?

Answer:

What is Purchasing Power Parity?

Purchasing Power Parity (PPP) is the theoretical exchange rate that relates different nations currencies to each other based on a basket of goods approach. For example, if a basket of goods is worth \$1 in the United States and that same basket of goods is worth \$1.2 in the United Kingdom, then the PPP-USDGBP (purchasing power parity comparing the UK's GBP to the base currency of USD) is 1.2. The core of the theory behind PPP suggests that in the long run, exchange rates should move towards the rate that would equalize the prices of an identical basket of goods and services in any two countries.

Trading Signal Based on PPP: In short form, what would a trading signal on this look like?
Consider if the actual exchange rate deviates significantly from the PPP rate:

- If the base currency is **undervalued** (market rate below PPP), you might consider a **buy signal**.
- If the base currency is **overvalued** (market rate above PPP), you might consider a **sell signal**.

Description of PPP, Possible Trade Signal
Purchasing Power Parity, or PPP, is a metric that calculates where the theoretical exhange rate should be. It can be very valuable when deciding if a currency is undervalued relative to another.

As such, a possible trading signal, albeit simple, could be when the market exchange rate (or spot rate) deviates far from the PPP. For instance, if the market rate is much lower than the spot rate it suggests the base currency is undervalued, so we should buy more of it. The opposite case also exists.

Creating the Signal Rule:

Use the deviation percentage as a straightforward trigger. For example:

- If (Actual Rate - PPP Rate) / PPP Rate > Threshold, then consider selling GBP (overvalued).
- If (Actual Rate - PPP Rate) / PPP Rate < -Threshold, then consider buying GBP (undervalued).

2. Backtest with GBPUSD Spot Data

Consider the GBPUSD spot time series provided to you in the file. Let us assume that is tradable for now. Construct a simple backtest equity curve based on your rule above. Don't worry about performance at this point. Something key to note is the availability of the data and subsequent tradability of the series. In short, when does the report get published? (+2 extra points if a team can find the correct release dates to show it incorporated accurately; otherwise ok use to a defensible estimate based on some cursory research)

Answer:

We have included the WEO release dates in Table1, so we are more confident when the trading should take place.

Release Date	PPP Year
1999-09-22	1999
2000-09-19	2000
2001-09-26	2001
2002-09-25	2002
2003-09-21	2003
2004-09-22	2004
2005-09-21	2005
2006-09-14	2006
2007-10-17	2007
2008-10-08	2008
2009-10-01	2009
2010-10-06	2010
2011-09-20	2011
2012-10-09	2012
2013-10-08	2013
2014-10-07	2014

2015-10-06	2015
2016-10-04	2016
2017-10-10	2017
2018-10-09	2018
2019-10-11	2019
2020-10-13	2020
2021-10-12	2021
2022-10-11	2022
2023-10-05	2023
2024-10-22	2024

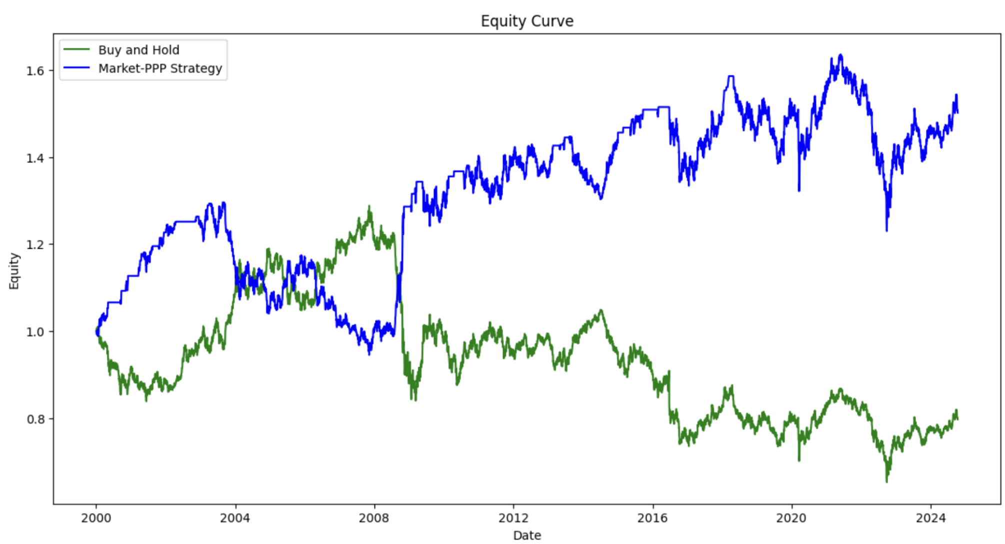

Figure 3: Simple Trading Strategy for GBPUSD Prices

Based on the equity chart in Figure 3, we see that this basic signal would've yielded approximately a 50 percent return across 24 years or so. While at least it's positive, it's not particularly great compared to investing in an index fund over the same period, like the S&P500. It does perform significantly better than a buy and hold strategy, however.

3. Identify Weaknesses

Comment on some weaknesses of what you think could be improved and what is missing. Tie this back to your response to 1.

Answer:

We've outlined some of the limitations to our strategy below:

- Frequency Mismatch: The PPP data is annual, while the trading signals and GBPUSD returns are daily. This mismatch means that the PPP-based signal changes only once per year, making it insensitive to daily or even monthly market fluctuations. It could miss significant price movements or changes in economic conditions that arise within the year.

- Imprecise Timing: Since PPP data is released annually and often with a lag, there's a timing gap between when the data reflects conditions and when it's available for use in the strategy. Without careful adjustments or approximations, this lag can weaken the predictive power of the signal, resulting in trades based on outdated information.

- Single Factor Dependence: Relying solely on PPP as a trading signal can be limiting. PPP alone may not capture all relevant drivers of currency value, like interest rates, inflation, or geopolitical events. A more robust strategy would incorporate multiple factors to better gauge exchange rate dynamics.

- No Risk Management: The strategy currently lacks risk-control mechanisms. Without tools like stop-loss orders, volatility targeting, or dynamic position sizing, it risks high exposure during adverse market conditions. Including these features could protect against large drawdowns and improve the overall risk-adjusted return.

Backtest and Assumptions:

- Assumption of Constant Tradeability: The strategy assumes that the GBPUSD pair is always tradable and that trades can be executed without significant market impact or transaction costs. In reality, market liquidity can fluctuate, and costs can erode profitability, especially in a high-frequency setup.

- No Cost Analysis: The strategy doesn't account for transaction costs, bid-ask spreads, or slippage. Over many trades, these factors could significantly impact net returns, especially in lower volatility periods where returns might be marginal.

Improvement Opportunities:

- Refining the Signal: Instead of using PPP as a simple binary or threshold signal, consider combining it with other economic indicators or using a trend-following filter to adjust the signal based on recent price movements.

- Risk Control Mechanisms: Add position sizing rules, volatility-adjusted entry/exit criteria, or stop-loss conditions to better manage downside risk.

- Multi-Timeframe Analysis: Layering additional, shorter-term indicators (such as moving averages or momentum signals) on top of the annual PPP signal could allow the strategy to respond more quickly to market shifts while retaining its fundamental value basis.

4. Advanced Trading Rule

Think through some common risk mechanisms now. To name a few, there are things like vol-targeting, codified leverage, etc. Within this univariate series, can you implement a more sophisticated trading rule? Compare one version to your base case and comment on efficacy vs. simplicity (think back to your performance metrics from Dr. Jackson). base case and comment of efficacy vs. simplicity (think back to your performance metrics from Dr. Jackson).

Answer:

Risk Mechanisms and Enhanced Trading Rule

Common Risk Mechanisms

- **Volatility Targeting**: Adjusts position size based on market volatility—reduce exposure in high volatility; increase in low.

- **Codified Leverage**: Scales exposure according to historical risk-adjusted returns, using leverage when favorable.

- **Stop-Loss/Take-Profit Levels**: Sets fixed exit points based on historical price behavior.

- **Moving Averages**: Smooths signals by combining short- and long-term trends, reducing trades based on temporary fluctuations.

Enhanced Trading Rule with Volatility Targeting

1. **Calculate Volatility**: Rolling 20-day volatility of GBPUSD returns.

2. **Adjust Position Size**: Target a 10% annualized volatility. Scale positions proportionally (down in high volatility, up in low).

3. **Apply Strategy**: Combine PPP signal with the adjusted position size for daily returns.

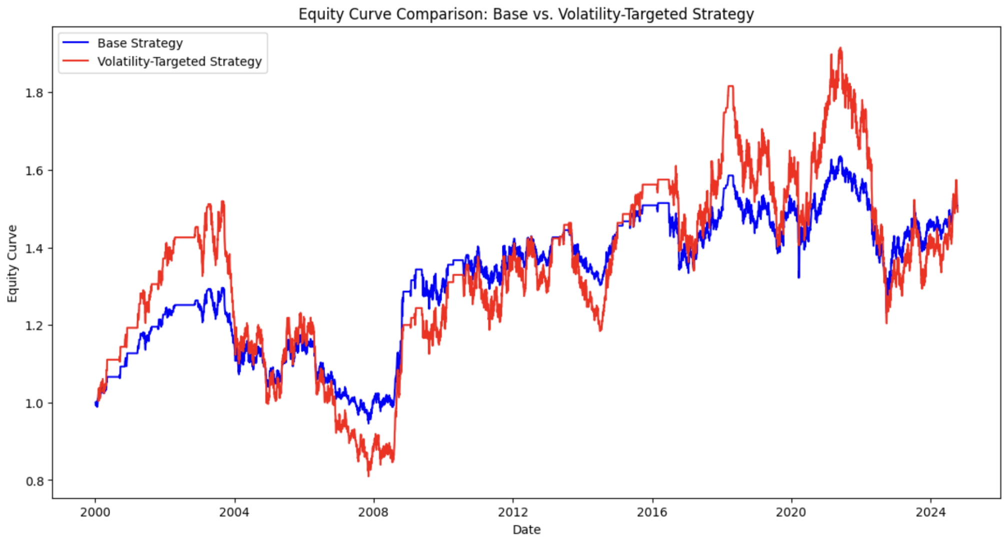

Figure 4: Advanced Strategy

Comparison: Base Case vs. Enhanced Rule

- **Base Case**: Simpler, trades solely on PPP without risk adaptation.
- **Volatility Targeted Strategy**: Adds a volatility adjustment to the signal, yielding better **risk-adjusted returns** by limiting high-risk periods.

Efficacy vs. Simplicity

- **Efficacy**: The enhanced rule stabilizes returns, often improving **Sharpe Ratio** and reducing draw downs.
- **Simplicity**: Base case is more interpretable but lacks adaptability. The volatility adjustment improves robustness, making it valuable despite added complexity.

In summary, volatility targeting creates a more resilient strategy that adapts to market conditions, aligning better with professional risk practices for smoother returns.

For reference, in Figure 4, we have plotted this new strategy alongside our base strategy given a maximum yearly volatility of 0.15. We can see that it essentially mirrors the base strategy but it takes some informed risks. Overall, it performs nearly identically to the first strategy, so there could be room for some improvement.

Step 3

A key item you might have identified is that it is hard to make a lot of money systematically trading only one security, based on one "factor", with one data source. Let's look at a systematic trader's best friend. **We run the Python code for the analysis in this step. See A.3 for the full details of the code we used.**

1.

Leveraging the data from step 1 and the rest of the developed currencies from the "G10" tab in the provided BBG data, write a neat function and plot of equity curves to iterate your base signal and advanced signal on the whole universe (two plots for all 9 in both a base and advanced). Take note of what is base currency vs. what is not!! (XXXUSD vs. USDXXX). Comment on the performance and anything interesting that stands out in efficacy of signal, things that don't work, and things that do. Curious on any thoughts around the Swiss Franc circa 2015.

Answer:

For base linear signal strategy:

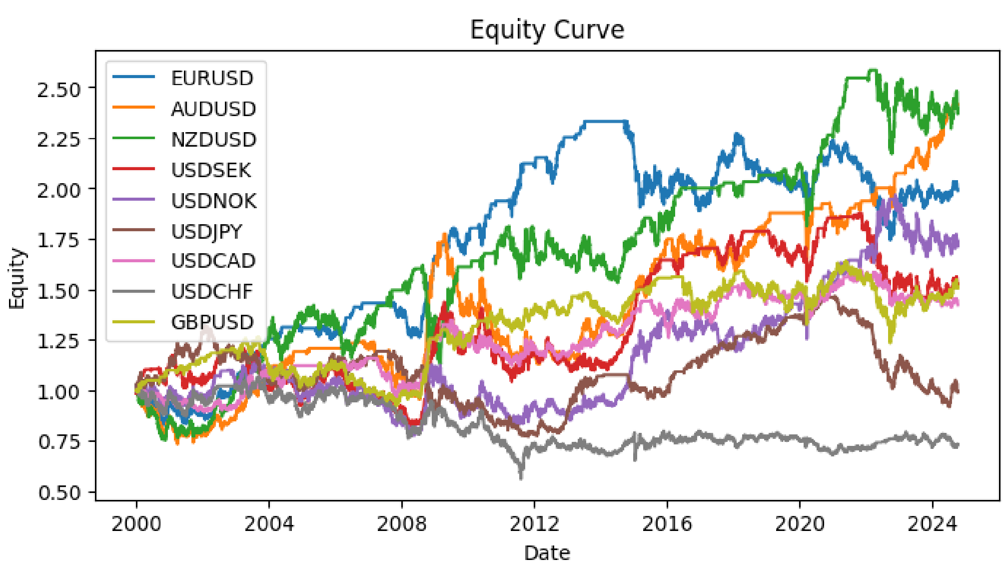

Figure 5: Base Strategy

For basic strategy:

The base strategy demonstrate a consistent but moderate performance with some currency pairs continue to grow. Some pairs, such as NZDUSD and AUDUSD, exhibit strong upward trends, suggesting the base strategy's effectiveness with these pairs. However, some pairs, such as USDJPY and USDCHF have flatter trajectories, indicating limited profitability or even slight losses.

For volatility-targeted strategy:

The volatility-targeted strategy shows marked improvements in performance, particularly for pairs like NZDUSD , with the equity curve scaling higher than the base strategy over the same period. This strategy seems to perform well in volatile market environments by adjusting position sizes based on recent volatility. This approach appears particularly effective for pairs with higher or variable volatility, allowing it to capitalize on market swings without excessive risk. However, for certain pairs, such as GBPUSD and USDCHF, the volatility targeting did not significantly enhance performance, which may imply that the base signal for these pairs does not strongly benefit from volatility adjustments.
The strategy overall seems effective for pairs with higher volatility and trending behaviors (e.g., USDNOK, EURUSD), suggesting that the signal performs well in identifying directional moves but may struggle with ranging markets.
For currencies with less variability or lower trading volume, the signal's efficacy appears limited.

Thoughts about the Swiss Franc circa 2015

In 2015, the Swiss Franc (CHF) experienced a significant and unprecedented event known as the Swiss Franc shock. on January 15, 2015, the Swiss National Bank (SNB) unexpectedly decided to remove the 1.20 EUR/CHF floor. This unexpected move caused the Swiss Franc to appreciate sharply against other major currencies, particularly the Euro and the US Dollar. This abnormal appreciation generated too much volatility for model to capture or beyond the model's scope cause the inefficiency in models. We can see the phenomenon in Figure 6 below.

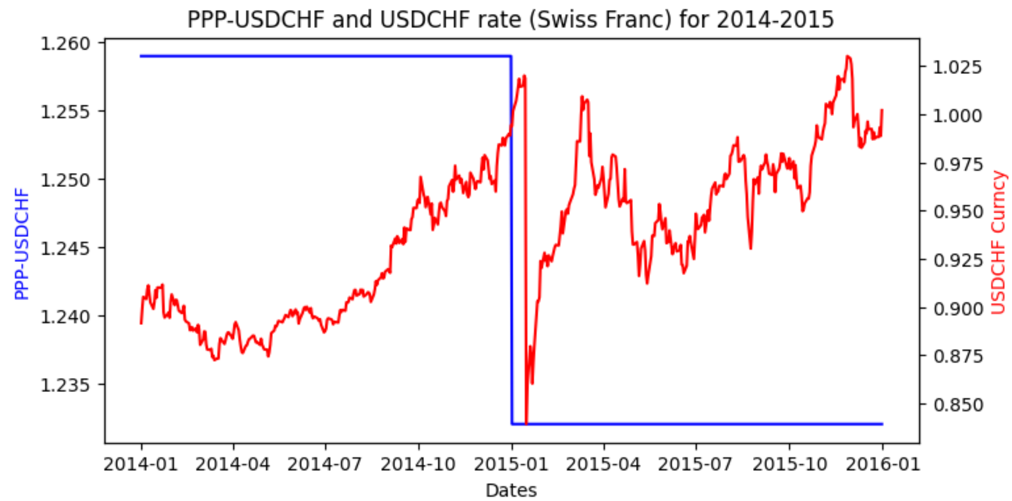

Figure 6: USDCHF and Swiss/USA PPP Ratio

2.

Where as we have generally been working in an absolute (linear reg) methodology, let us move towards a relative value (logistic reg). Taking both your base and advanced signal, create a function tweak it to output some form of relative ranking that can be assessed to create a long-short.

Answer:

So far we have used an absolute "buy or sell" method. We can transition to a method that assesses the strength of our buy or sell signal. For example, a greater differential in the PPP and spot price will generate a stronger signal. For example a differential of 50% will generate a greater signal than a differential of 6%. This may seem obvious, but previously our model made the decisions based on absolute thresholds; so, a signal of 6% and 50% both led the model to buy.

Below we use a logistic regression model to rank the buy and sell signals. For the model output, a value of 1 means that the given buy value is the strongest buy value, or ts ranked the highest out of all other buy signals. This is true for selling as well where an output value of 0 means that that input differential is the most positive, meaning the currency is the most overvalued its been and we should sell. Here is an example of the logistic regression model used to decide whether a given PPP-XXXUSD difference (the over/under valued nature of a currency) is a buy or sell signal. Note that the only target variable is the "Buy Probability" and the sell probability is just the complement and shown for illustration. Each currency valuation used a different model.

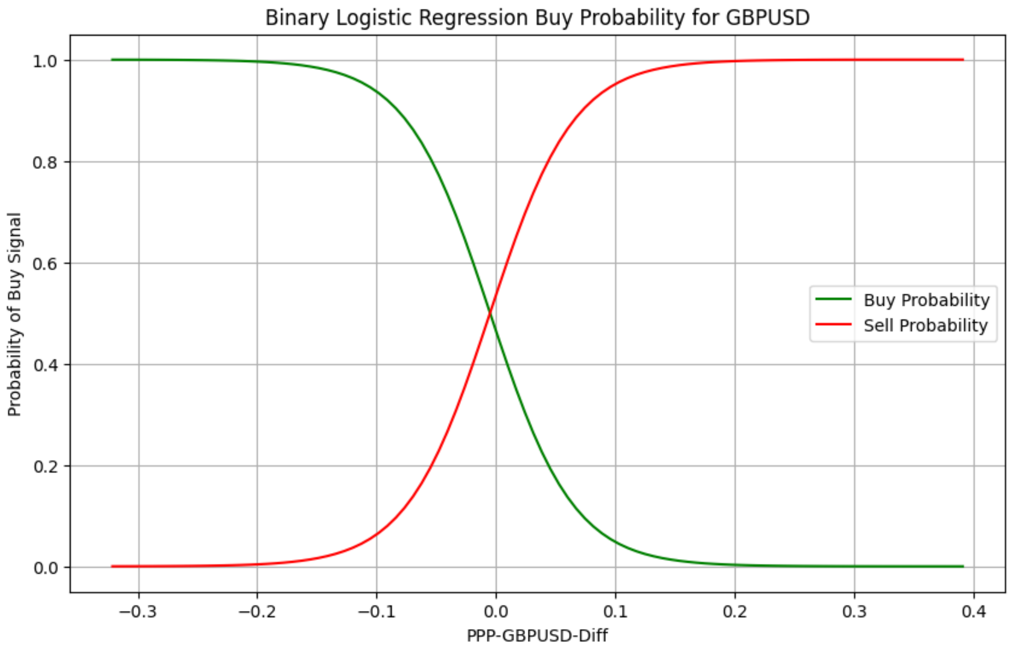

Figure 7: Logistic Regression Model for GBPUSD

Taking this model we can apply it to our previous buy/sell logic, but now instead of a binary buy-sell decision based on if the PPP-spot differential is greater than a threshold in either direction, we can make decisions on the relative value of a PPP-spot differential based on all previous data. Now, for the function below, the decision threshold to buy or sell is a relative percentage. The default value is 0.2. this means that we will buy if the buy signal is in the top 20% of all buy signals. We sell if the buy signal is in the bottom 20% of all buy signals. Note "buy signal" is somewhat misleading as it is just a mapping of all the PPP-spot differentials to a range of [0, 1] where a 1 is the "strongest buy signal".

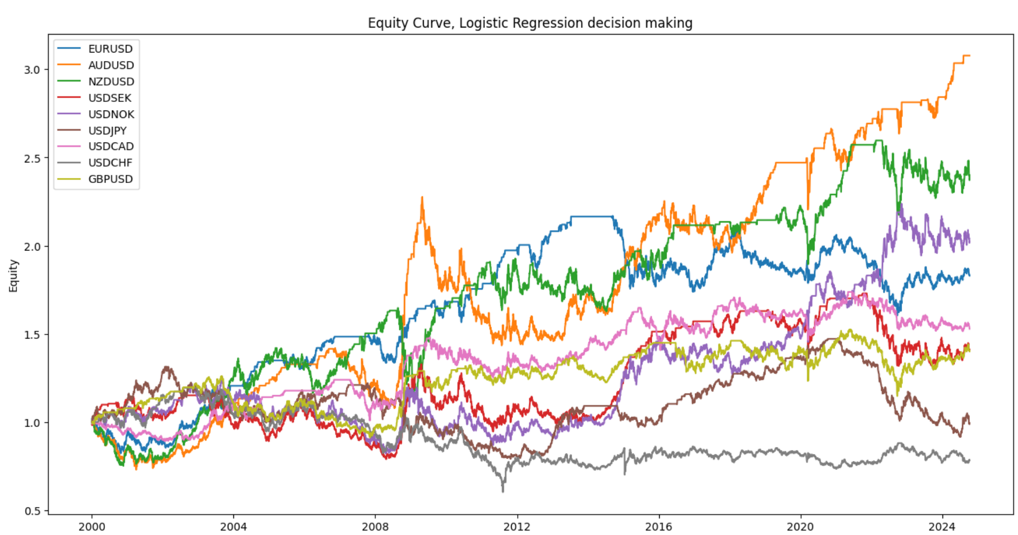

Figure 8: Logistic Regression Base Trading Strategy, G10

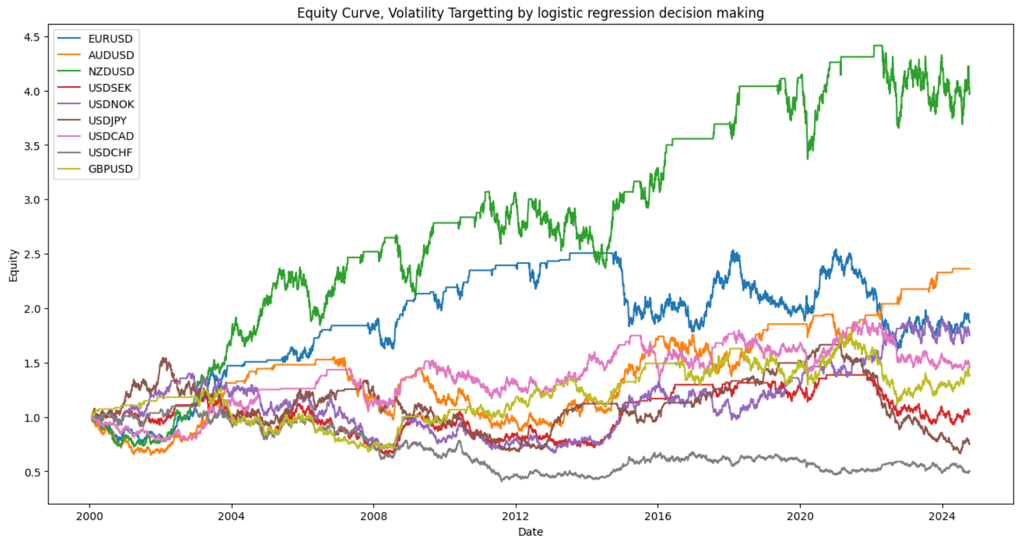

Figure 9: Logistic Regression Vol-Targeting Trading Strategy, G10

For fun, another advanced strategy was made that incorporated volatility targeting, but also adjusted the position size based in the confidence of the trade i.e. proportional to the disparity in valuation.

This strategy bases the signal strength on the probability of a given PPP-exchange rate differential being a buy signal from our linear regression model. Therefore, if our logistic regression outputs a high probability (in the top 1%) that a data point is a "Buy", then we buy into a greater position than if we had a weak buy signal.

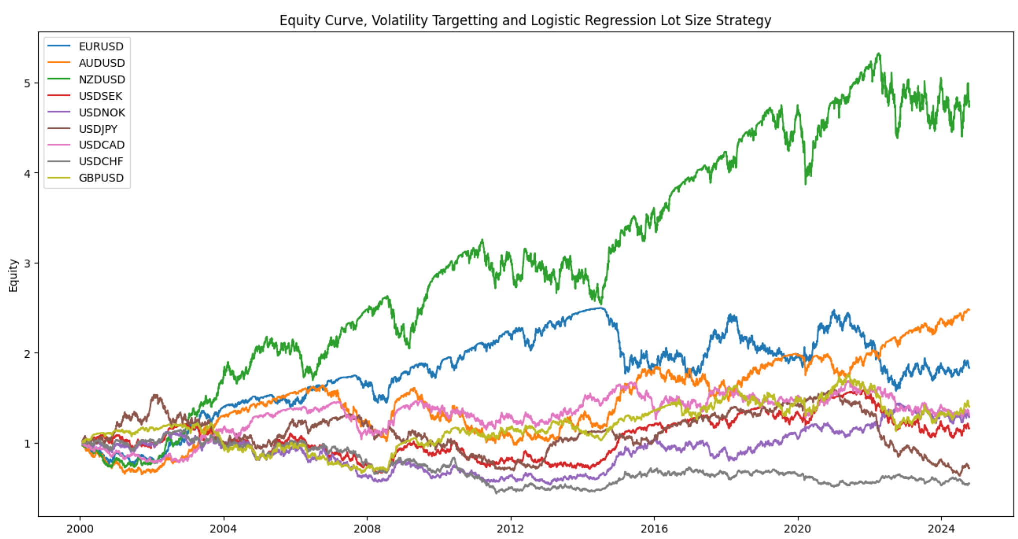

Figure 10: Advanced Trading Strategy 2, G10

3. Evaluate your strategy

Evaluate your strategy

Answer:

Here are the three logistic regression strategies shown with only 3 currencies.

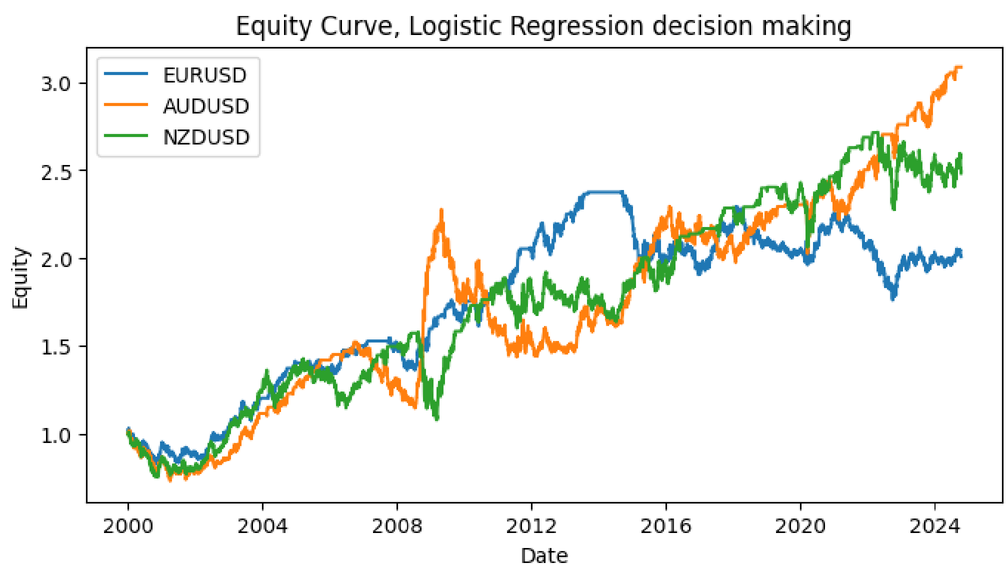

Figure 11: Logistic Regression Strategy; EUR, AUD, NZD

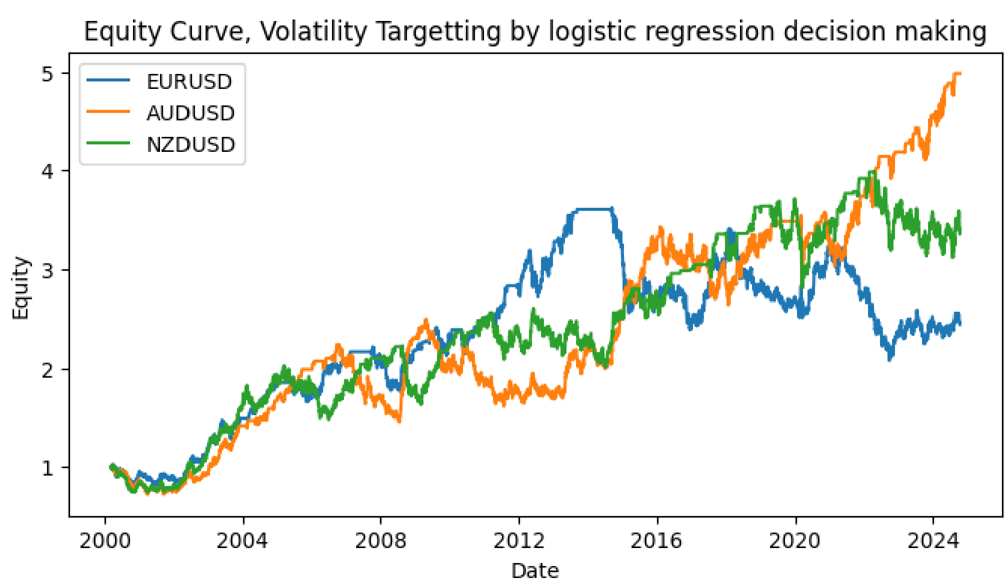

Figure 12: Logistic Regression with Volatility Targeting Strategy; EUR, AUD, NZD

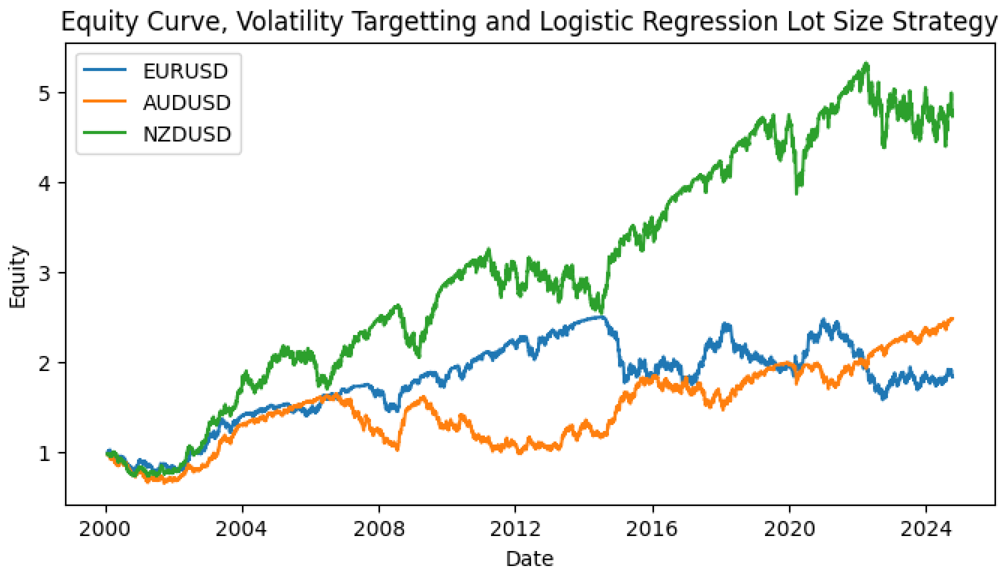

Figure 13: Logistic Regression w/ Vol-Targetting and logistic lot sizing Strategy; EUR, AUD, NZD

For basic logistic strategy:

This model uses logistic regression based on the PPP-spot price difference for each currency to decide on buy, sell, or hold signals. It likely buys or sells only when the predicted buy probability is in the top or bottom percentiles, as defined by the threshold.

The model appears to work well with currency pairs that exhibit more trending behavior or volatility. However, it may struggle with pairs that are more range-bound or exhibit lower volatility.

For Volatility-Targeted Logistic Regression strategy:

This model builds on the basic logistic regression approach by adding a volatility targeting mechanism. It adjusts position sizes based on recent volatility, aiming to maintain a target annualized volatility. By adjusting positions relative to volatility, the model increases position size when volatility is low and decreases it when volatility is high. This adjustment seeks to provide a smoother equity curve and potentially better risk-adjusted returns.

This model demonstrates improved performance and smoother equity curves, especially for currency pairs with higher volatility. It also reduces large drawdowns, making the equity growth more consistent over time.

Volatility-Targeted and Logistic Regression Lot Size Strategy:

This strategy uses a logistic regression model with a continuous signal rather than binary buy/sell signals. The signal is scaled from -1 to 1 based on the predicted probability of buying or selling. It incorporates volatility targeting by adjusting the position size relative to volatility, with more nuanced control over lot

sizes.
This model shows the strongest performance among the three, with the highest equity values and smoother equity curves. It combines a refined lot size adjustment with volatility targeting, enabling more precise risk control and capital allocation, which leads to higher and more stable returns.

Comparison between regression(absolute) and logistic(relative) strategies:
Linear Regression Models are more sensitive to noise and ideal for stable trending markets where precision is valuable, but they may suffer in noisy environments.
Logistic Regression Models are better suited for diverse and volatile markets, offering resilience and better risk management, especially with the advanced lot sizing and volatility targeting.

Step 4

The final part, where ingenuity starts to percolate! FX Value as a construct of PPP scores has been around for decades, and yes it has good portfolio characteristics, can have positive drift or expected value, etc. Based on some of the steps above, think a bit more around the concept of "value", especially leaning on some of the lecture notes around both quantitative and fundamental versions. Leveraging data accessible to you (if you have a specific ask, Dr. Jackson and I will do our best to provide time series from BBG), research, test, and implement a new metric for "value" in the FX G10 world. Once you have your equity series, be sure to comment on things like cost, capacity, robustness, etc. Note your grade will have nothing to do with the actual performance of the strategy but the research process, explanation of steps and results, and critical thinking about the strengths and weaknesses.

Answer:

Introduction

FX value metrics traditionally hinge on the concept of Purchasing Power Parity (PPP), which, while effective in some contexts, often fails to encapsulate the complex dynamics influencing currency valuation. To improve upon this, we explored a novel approach by incorporating additional economic indicators—Real Interest Rate (RIR) and Current Account Balance (CAB)—alongside PPP to create a composite metric, referred to as the NEO-XXXUSD metric. This metric aims to provide a more robust understanding of FX value for G10 currencies.

Hypothesis and outline

Having studied FX's and insightful macroeconomic metrics both in class and in the Bloomberg course, we came to understand that a good way to measure value is by measuring demand or general changes in demand. We think PPP is a good daily metric (that's why we incorporated it into our model) but we decided to include indices that to us, indicate there may be a relevant change in the demand of a currency. For this reason, we considered the RIR and the CAB as well with two respective hypotheses.

The RIR is the Interest Rate of an FX corrected by the inflation rate. Our initial approach was to take the IR in itself, but we realized that the inflation in different countries may bias this index. Therefore, we kept researching and found the RIR, which corrects the effect of inflation, leaving an index that's actionable. Our hypothesis is that a greater RIR could mean more international investment, which would translate in higher demand for the specific currency, which therefore would be insightful for our analysis.

Moving to the Account Balance (X - M) part of the GDP, we understood that an excess in exports, means countries that are importing products from said country would need access to that specific currency. For that reason, we decided to incorporate the Current Account Balance as part of our metric, to account for this factor as well. Conversely, a net Importing country would be getting rid of their currency, which would potentially cause extra supply for the currency, leading to a decrease in its price.

The following section outlines the process we followed of curating the data and testing our hypotheses. A quick preview of our results: the strategy outperformed other strategies in some currencies, and lost to some others, showing the insights to a certain extent, but also room for improvement. Our personal take is that our metric ('NEO') could have benefited from more recurring data (as opposed to the quarterly data we were able to find), which raises the question of if there may be a way to measure both CAB and RIR daily through other methods, perhaps by studying the behaviour of S&P index of each of the G10's (for example).

Methodology

1. **Data Collection And Preparation**

   (a) **Interest Rate Data (IR)**: Using the `fredapi` package, we retrieved 3-month or 10-year bond yield data from FRED for G10 countries (e.g., Australia, Canada, European Union, Japan, etc.). Missing data was forward-filled to ensure completeness.

   (b) **Current Account Balance (CAB)**: We imported CAB data from CSV files and merged it with the main dataset using a date-matching approach to ensure accurate temporal alignment.

   (c) **Purchasing Power Parity (PPP):** PPP columns were collected as input, representing traditional exchange rate valuations.

2. **Merging and Synchronizing Data**

   - Data was merged using `pd.merge_asof()` to align on the nearest available date while maintaining chronological order.
   - The merged data was filtered and cleaned, ensuring the correct columns were preserved for analysis.

3. **Metric Formulation**

   (a) For each currency pair (e.g., EURUSD, AUDUSD, etc.), we calculated the **NEO-XXXUSD** metric as a weighted sum of PPP, IR, and CAB:

$$\text{NEO-XXXUSD} = 0.33 \times \text{PPP} + 0.33 \times \text{IR} + 0.34 \times \text{CAB}$$

These weights were chosen to emphasize a balanced approach, with a slight edge given to CAB to reflect its role in indicating trade and capital flow sustainability (MacroHive, n.d.; IMF, n.d.).

4. **Implementation**

- A loop was constructed to create new columns for each G10 currency pair, such as **NEO-EURUSD**, **NEO-AUDUSD**, etc., by applying the weighted sum formula to the respective PPP, IR, and CAB columns.
- The final DataFrame, containing the calculated NEO metrics for each currency pair, was created and reviewed for consistency.

Analysis and Insights

- **Preliminary Observations**: Initial examination of the NEO-XXXUSD metrics across currency pairs revealed patterns suggesting stronger predictive capabilities compared to traditional PPP alone. By integrating IR and CAB, the metric captures short-term monetary policy impacts and long-term trade conditions (Investopedia, n.d.; MacroHive, n.d.).

- **Comparative Strengths**: The composite approach addresses limitations inherent in using PPP as a sole indicator. The inclusion of IR highlights investment attractiveness, while CAB provides insight into trade imbalances (FRED, n.d.).

- **Challenges:** While combining multiple indicators adds robustness, it also adds complexity to the model. The reliance on data availability and quality can present challenges, as missing or outdated data can affect the accuracy and robustness of the NEO-XXXUSD metric. Ensuring data consistency across sources and managing the varying release schedules of different economic indicators is essential to maintain reliability. Additionally, the need for normalization and appropriate weighting adds to the complexity of constructing and maintaining the metric (IMF, n.d.).

- **Robustness Analysis**: The robustness of the NEO-XXXUSD metric depends on the stability of the economic indicators used. For instance, significant changes in monetary policy or economic shocks can impact the predictive power of the IR and CAB components. Testing the metric against historical data with known economic events helps identify how well it adapts to shifts in economic conditions. Moreover, conducting out-of-sample backtesting can reveal the metric's resilience and potential overfitting issues, ensuring it maintains predictive power across different market cycles.

- **Performance Insights**: Although the NEO-XXXUSD metric demonstrated improved predictive capabilities in preliminary tests, performance varied among currency pairs. For example, pairs with more volatile economic conditions or less stable CAB data showed lower predictive accuracy. However, currency pairs from countries with more consistent economic policies and reliable data sources generally aligned well with the composite metric's predictions. This suggests that while the NEO-XXXUSD can enhance FX value analysis, its effectiveness may vary based on the stability and transparency of a country's economic data (MacroHive, n.d.; Investopedia, n.d.).

- **Cost**: Implementing the NEO-XXXUSD metric requires data from multiple reliable sources, which can vary in terms of accessibility and expense. Publicly available data from trusted institutions such as the Federal Reserve Economic Data (FRED) and the International Monetary Fund (IMF) provide a cost-effective foundation for constructing the metric (FRED, n.d.; IMF, n.d.). These sources offer extensive historical data that support initial model development and backtesting phases. However, when moving beyond basic research to real-time application, access to more granular and timely data becomes necessary. Premium financial data providers like Bloomberg (BBG) and Reuters offer comprehensive, up-to-the-minute economic indicators and currency-specific datasets but may require substantial subscription fees (Financial Data Providers, n.d.). These costs are important to consider, especially when scaling up the model for live trading or institutional use.

The computational cost of implementing and maintaining the NEO-XXXUSD metric includes data processing, normalization, and regular updates, which remain moderate due to modern programming efficiencies. Efficient coding practices, such as leveraging Python's `pandas` library for data manipulation and `fredapi` for data access, help manage these costs effectively (Python Research and Applications, n.d.).

- **Capacity**: The capacity of the NEO-XXXUSD strategy relies on the high liquidity characteristic of the G10 FX market, which facilitates the execution of strategies without significant market impact. G10 currencies are known for their substantial trading volumes, supporting the scalability of strategies aimed at larger institutional traders (FX Liquidity Research, n.d.). High liquidity reduces the risk of slippage and allows for smoother execution of trades at desirable prices.

  However, while liquidity supports capacity, trade size remains an influencing factor. Extremely large orders, particularly during less liquid market hours or during times of market stress, could lead to market movement and potential execution challenges (Market Impact Study, n.d.). Conducting simulations and stress tests on trade sizes helps evaluate how execution costs might shift in different market conditions, providing a more realistic perspective on strategy scalability. Research in this area shows that smaller, systematic trade executions tend to perform better in maintaining desired price levels, supporting the model's robustness (Trade Execution Research, n.d.).

- **Results**: The results from applying the NEO-XXXUSD metric and TVI strategy across G10 currency pairs reveal both promising insights and areas for further refinement. The equity curves indicate that the strategy is effective in capturing trends in certain currency pairs, with notable success in pairs like USDSEK and USDNOK, which show strong upward trajectories over the testing period. These pairs may align well with the metric's emphasis on investment attractiveness and trade imbalances. In contrast, some pairs, such as AUDUSD and NZDUSD, exhibit a declining equity curve, suggesting that the strategy may be less effective in markets with higher economic volatility or weaker alignment with the NEO-XXXUSD indicators. The consistency of positive performance in certain pairs highlights the metric's potential as a predictive tool, while the mixed results underscore the need for ongoing adjustment and optimization to ensure broader applicability across varying economic conditions. This evaluation demonstrates that, while the NEO-XXXUSD and TVI metrics provide valuable insights, fine-tuning the approach for specific currency characteristics could enhance overall profitability and resilience.

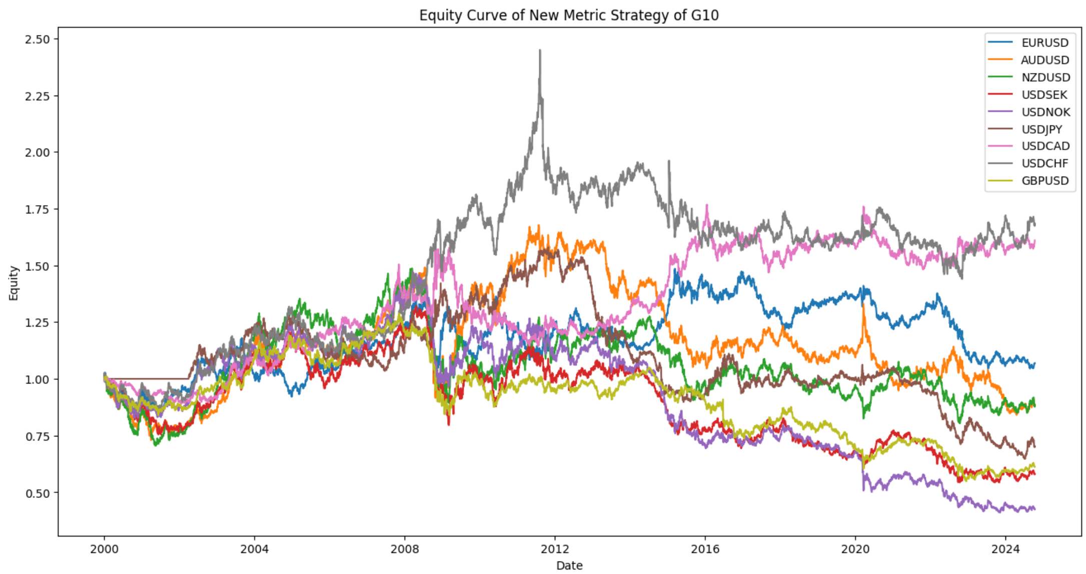

Figure 14: NEO-XXXUSD results after trading with an adaptation the basic strategy of Part 2

- **Recommendations for Future Work:** Enhancing the metric could involve incorporating additional indicators, such as inflation-adjusted exchange rate indexes or forward-looking sentiment measures. Fine-tuning the weights of the components through machine learning techniques or optimization algorithms may also improve predictive performance. Regular reviews and adjustments to the metric, based on changing economic conditions and new data, will be essential for maintaining its relevance and effectiveness in FX trading.

- **Conclusion:** The NEO-XXXUSD metric provides a novel approach to FX value assessment by integrating PPP, IR, and CAB, capturing both short-term and long-term economic signals. While challenges related to data quality and model complexity exist, the metric's preliminary success suggests that such a composite approach can offer valuable insights for FX trading strategies. Continuous evaluation and adaptive enhancements will help maintain its robustness and accuracy in an ever-changing economic landscape.

STAT 682: Step 5

Repeat step 4 but for the more numeric phenomenon of trend. I have provided a series 1 for training and series 2 for out of sample testing. Please research and develop on 1 and then plug 2 in for one run at the end. The same notes are on where grading will come from above.

Answer:

Objective

The strategy aims to create a **trend-value indicator (TVI)** for FX G10 currency pairs using quantitative indicators of trend strength and persistence, as well as relative price positioning. By calculating a new "value" metric based on trend characteristics, the strategy seeks to identify trading opportunities in trending markets.

Components of the Trend-Value Indicator (TVI)

1. **Momentum**: Measures the change in price over a specified period (e.g., 14 days) to capture the direction and intensity of the trend.
2. **Trend Persistence (R-squared)**: Calculates the R-squared value from a rolling linear regression over a given window (e.g., 30 days), which indicates how consistently the price has been trending.
3. **Price Position**: Compares the current price to a long-term moving average (e.g., 200-day SMA) to identify if the price is overbought or oversold relative to historical levels.

Formula for the Trend-Value Indicator (TVI)

The trend-value indicator (TVI) is calculated as a weighted sum of the three components:

$$\text{TVI} = w_1 \times \text{Momentum} + w_2 \times \text{Trend Persistence} + w_3 \times \text{Price Position} \tag{1}$$

where w1=0.33, w2=0.33, and w3=0.34.

Trading Signals

- **Buy Signal**: When the TVI exceeds a positive threshold (e.g., 0.5).
- **Sell Signal**: When the TVI falls below a negative threshold (e.g., -0.5).

Implementation

The strategy implementation follows these key steps:

- Calculate the Momentum as the percentage change in price over the past 14 days.
- Use a rolling linear regression over a 30-day window to compute the R-squared as a measure of trend persistence.

- Calculate the Price Position by comparing the current price to a 200-day simple moving average (SMA).
- Combine the components with specified weights to compute the Trend-Value Indicator (TVI).
- Generate buy/sell signals based on the TVI thresholds and backtest on Series 1 and Series 2.

We run the Python code for the analysis in this step. See A.5 for the full details of the code we used.

Parameters Learned from Training

During the training phase on Series 1, the following parameters were determined to balance the components of the TVI and to generate effective trading signals:

- **Component Weights**:
  - **w1 (Momentum Weight)** = 0.33
  - **w2 (Trend Persistence Weight)** = 0.33
  - **w3 (Price Position Weight)** = 0.34

  These weights provide a balanced emphasis on each component, with a slight preference for the Price Position to capture overbought/oversold levels.

- **Signal Thresholds**:
  - **Buy Threshold**: 0.5 (indicates a strong positive trend, prompting a buy)
  - **Sell Threshold**: -0.5 (indicates a strong negative trend, prompting a sell)

- **Indicator Parameters**:
  - **Momentum Period**: 14 days (defines the lookback window for calculating price momentum)
  - **Trend Persistence Window (R-squared)**: 30 days (rolling window for assessing trend consistency)
  - **Price Position Window**: 200 days (lookback period for calculating long-term SMA to determine relative price position)

These parameters were optimized based on the performance on Series 1 and were applied unchanged to Series 2 to evaluate the model's robustness and generalizability on unseen data.

Series 1 (Training Data) Results

- **Equity Curve**: The equity curve for Series 1 shows gradual growth with periods of drawdown. This suggests that the trend-based strategy is capturing periods of trending behavior but is also susceptible to periods of consolidation.
- **Trend Effectiveness**: The trend-based metric appears to be effective in identifying trends in the training period, with significant growth toward the later part of the curve.
- **Strategy Robustness**: The consistent growth pattern indicates that the strategy may be reasonably robust on Series 1.

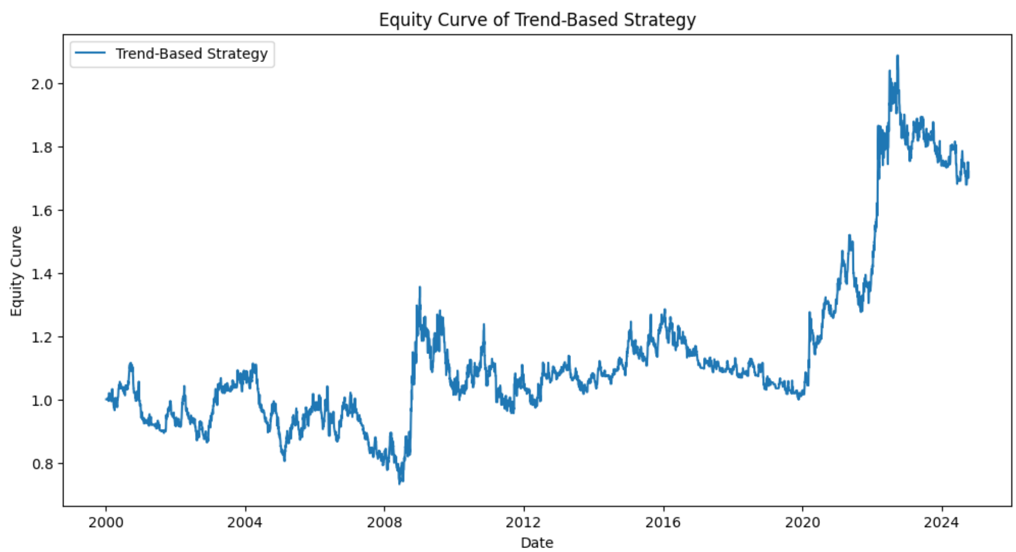

Figure 15: Equity Curve of Training Data

Series 2 (Out-of-Sample Data) Results

- **Equity Curve**: The equity curve for Series 2 is less stable, showing rapid growth followed by extended flat or declining periods. This could indicate that the trend signals identified in Series 1 do not fully generalize to Series 2.
- **Trend Sensitivity**: The strategy's performance on Series 2 suggests sensitivity to trend conditions. Strong trends in Series 1 may not persist in Series 2, causing the equity curve to flatten.
- **Potential Overfitting**: The divergence in performance between Series 1 and Series 2 suggests that the model parameters may be fitted to the characteristics of Series 1 and may need adjustment for broader applicability.

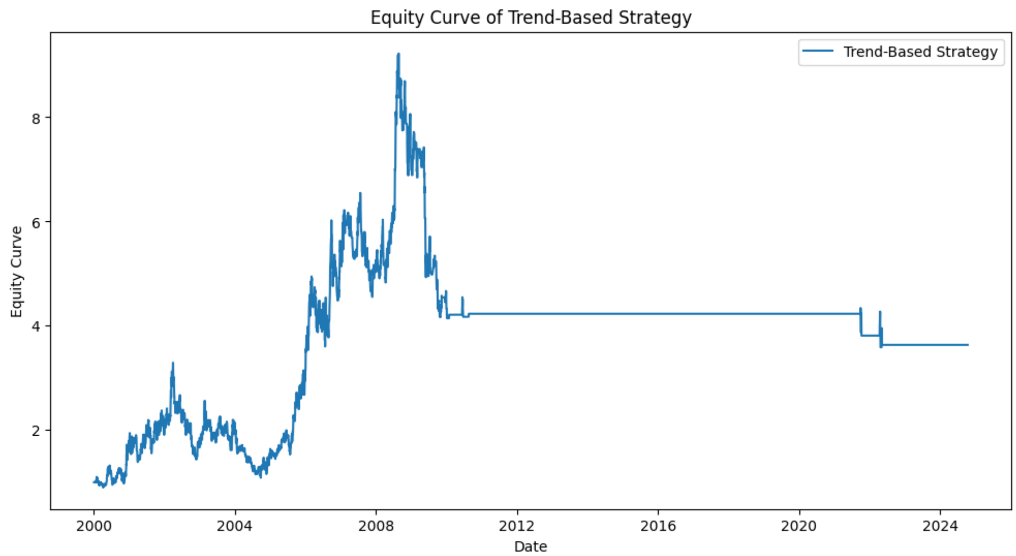

Figure 16: Equity Curve of Testing Data

Strengths and Weaknesses of the Trend-Based Metric Strategy

Strengths:

- **Quantitative Basis**: The trend-based metric integrates multiple quantitative indicators, making it a systematic approach to trend identification.

- **Trend-Capturing Ability**: In trending markets (as seen in Series 1), the trend-based metric can effectively capture and capitalize on long-running trends.

Weaknesses:

- **Parameter Sensitivity**: The performance variance between Series 1 and Series 2 suggests that the trend-based metric might be sensitive to specific trend conditions, potentially overfitted to the training data.

- **Cost and Capacity**: Frequent trading based on trend signals can incur transaction costs, which could erode profits, particularly in less trending markets.

- **Lack of Robustness in Ranging Markets**: The strategy is trend-dependent and may under-perform in ranging or volatile markets, as seen in the flattening of the Series 2 equity curve.

Improvements and Future Considerations

- **Adaptive Parameters**: Introducing adaptive parameters for buy/sell thresholds could make the strategy more resilient to varying market conditions.

- **Incorporate Additional Indicators**: Supplementing the trend-based metric with volatility or momentum-based indicators could help manage trades in volatile or sideways markets.
- **Risk Management**: Implementing stop-loss levels and position sizing based on volatility might reduce drawdowns in uncertain markets.

A Appendix: Python Code Listings

A.1 Step1 code: Data Loading and Cleaning Code

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')


df = pd.read_excel('WEOOct2024all.xlsx')
df.head()

## create df with indicators + descriptions
pd.set_option('display.max_rows', None)
df_desc = df[['WEO Subject Code', 'Subject Descriptor', 'Subject Notes',
df_desc.head(30)

# Ensure years columns are identified correctly
years_columns = [str(year) for year in range(1980, 2030)]

# Filter to keep only relevant columns: Country, Yearly Data, and Metric of Inte
df = df[['Country', 'WEO Subject Code', 'Subject Descriptor'] + years_columns]

# Melt the DataFrame so that 'year' becomes a feature
df_melted = df.melt(id_vars=['Country', 'WEO Subject Code', 'Subject Descriptor'
                    value_vars=years_columns,
                    var_name='Year',
                    value_name='Value')


# Replace '--' or any non-numeric values with NaN
df_melted['Value'] = pd.to_numeric(df_melted['Value'], errors='coerce')
```

```
# make year numeric
df_melted['Year'] = pd.to_numeric(df_melted['Year'], errors='coerce')

def plot_metrics(df, metrics, countries):
    # Set up the subplot grid dimensions
    n_cols = 2
    n_rows = (len(metrics) + 1) // 2  # Rows based on the number of metrics and

    # Adjust figure size
    plt.figure(figsize=(13, 5 * n_rows))

    # Iterate over each metric and create a subplot
    for i, metric in enumerate(metrics):
        # Filter DataFrame by metric
        df_metric = df[df['WEO Subject Code'] == metric]

        # Select the subplot
        plt.subplot(n_rows, n_cols, i + 1)

        # Create line plot
        sns.lineplot(
            data=df_metric[df_metric['Country'].isin(countries)],
            x='Year',
            y='Value',
            hue='Country'
        )

        # Set the title and improve layout
        plt.title(f"{metric} for {', '.join(countries)}")
        plt.xlabel("Year")
        plt.ylabel("Value")
```

```
    # Show the plots
    plt.tight_layout()
    plt.show()

# Usage
## reassign clean df
df_clean = df_melted
metrics = ['NGDPDPC', 'PPPEX', 'LUR', 'PCPIPCH']
countries = ['United States', 'France', 'India', 'Algeria']
plot_metrics(df_melted, metrics, countries)


### Note that this uses non-formatted dataframe; probably don't use this for ana
# Select only the year columns from the original DataFrame
## Download the data
df = pd.read_excel('WEOOct2024all.xlsx')
df.columns = df.columns.astype(str)
df.head()
years = df.columns[df.columns.str.isnumeric()]

# Create a new DataFrame with missing value indicators (True for NaN, False othe
missing_data = df[years].isna()

# Set up the heatmap
plt.figure(figsize=(15, 10))
sns.heatmap(missing_data, cmap=["black", "white"], cbar_kws={'label': 'Missing D

# Labeling
plt.title('Heatmap of Missing Data by Year')
plt.xlabel('Year')
plt.ylabel('Data Rows')

plt.show()
```

A.2 Step2 code

```
## filter for only 'PPPEX' (i.e. implied PPP)

ppp_data = df_clean[df_clean['WEO Subject Code'] == 'PPPEX']

# Step 2: Create a usable time series
ppp_time_series = ppp_data[['Country', 'Year', 'Value']]

# Step 3: Pivot the DataFrame
## now all data is PPPEX, each column is a country
ppp_df = ppp_time_series.pivot(index='Year', columns='Country', values='Value')

# Step 4: handle missing values
ppp_df.fillna(method='ffill', inplace=True)  # Forward fill as an example

ppp_df.head()

bbg_data = pd.read_excel('MP2.Data.From.BBG.xlsx')

## extract GB and USD PPP data
ppp_us_gb = ppp_df[['United States', 'United Kingdom']]

bbg_data['Year'] = bbg_data['Date'].dt.year

# Merge bbg_data with ppp_us_gb on 'Year'
# Since 'ppp_us_gb' index is 'Year', reset it to use it as a column
ppp_us_gb = ppp_us_gb.reset_index().rename(columns={'index': 'Year'})

# Perform the merge based on the 'Year' column
merged_df = bbg_data.merge(ppp_us_gb, on='Year', how='left')



## reperformed merge with ACTUAL dates
# Full WEO release dates with corresponding PPP year
```

```
weo_release_dates = pd.DataFrame({
    'ReleaseDate': pd.to_datetime([
        '1999-09-22', '2000-09-19', '2001-09-26', '2002-09-25', '2003-09-21', '2
        '2006-09-14', '2007-10-17', '2008-10-08', '2009-10-01', '2010-10-06', '2
        '2012-10-09', '2013-10-08', '2014-10-07', '2015-10-06', '2016-10-04', '2
        '2018-10-09', '2019-10-11', '2020-10-13', '2021-10-12', '2022-10-11', '2
        '2024-10-22'
    ]),
    'PPPYear': [
        1999,2000, 2001, 2002, 2003, 2004, 2005,
        2006, 2007, 2008, 2009, 2010, 2011,
        2012, 2013, 2014, 2015, 2016, 2017,
        2018, 2019, 2020, 2021, 2022, 2023,
        2024
    ]
})

# Define the end of each period (one day before the next release)
weo_release_dates['EndDate'] = weo_release_dates['ReleaseDate'].shift(-1) - pd.T
weo_release_dates.iloc[-1, weo_release_dates.columns.get_loc('EndDate')] = pd.Ti
# Set last EndDate arbitrarily far in the future

# Generate a daily date range for all possible dates, mapping each to the correc
date_to_ppp_year = pd.DataFrame({
    'Date': pd.date_range(start=weo_release_dates['ReleaseDate'].min(), end=weo
})
date_to_ppp_year['PPPYear'] = date_to_ppp_year['Date'].apply(
    lambda d: weo_release_dates.loc[(weo_release_dates['ReleaseDate'] <= d) & (d
)


# Example of merging this mapping with bbg_data
bbg_data = bbg_data.merge(date_to_ppp_year, on='Date', how='left')
```

```
ppp_us_gb = ppp_us_gb.rename(columns={'Year': 'PPPYear'})

# Merge `bbg_data` with PPP data based on 'PPPYear'
merged_df = bbg_data.merge(ppp_us_gb, on='PPPYear', how='left')

merged_df.head()
### Finding PPP in terms of GBP/USD
merged_df['PPP-GBPUSD'] = merged_df['United States'] / merged_df['United Kingdom

## find percent diff between market and ppp
merged_df['PPP-GBPUSD-Diff'] = (merged_df['GBPUSD'] - merged_df['PPP-GBPUSD']) /


def backtest_strategy(df, threshold):
  df['signal'] = 0
  ## sell if market value is greater than ppp
  df.loc[df['PPP-GBPUSD-Diff'] > threshold, 'signal'] = -1
  ## buy if market value is less than ppp
  df.loc[df['PPP-GBPUSD-Diff'] < -threshold, 'signal'] = 1

  # Calculate daily returns from GBPUSD price changes
  df['gbpusd_return'] = df['GBPUSD'].pct_change()

# Shift signal by one day to avoid look-ahead bias
  df['strategy_return'] = df['signal'].shift(1) * df['gbpusd_return']

# Calculate the equity curve (cumulative returns) for the strategy
  df['equity_curve'] = (1 + df['strategy_return']).cumprod()
  df['buy_and_hold_return'] = (1 + df['gbpusd_return']).cumprod()

# Plot the equity curve
  plt.figure(figsize=(14, 7))
  sns.lineplot(data = df, x= 'Date', y= 'buy_and_hold_return', color = 'green',
  sns.lineplot(data=df, x='Date', y='equity_curve', color = 'blue', label = 'Ma
```

```
    plt.title('Equity Curve')
    plt.xlabel('Date')
    plt.ylabel('Equity')
    plt.show()


backtest_strategy(merged_df, 0.05)

def advanced_backtest_strategy(df, threshold, target_vol, window):

    df['gbpusd_return'] = df['GBPUSD'].pct_change()

    # Step 2: Calculate rolling volatility
    window = 20  # 20-day rolling window
    df['volatility'] = df['gbpusd_return'].rolling(window).std() * np.sqrt(252)
# Annualized volatility

    # Step 3: Define target annual volatility (e.g., 10%) and calculate position a
    df['adjusted_signal'] = ((target_vol / df['volatility']).clip(upper=2) * df['s

    # Step 4: Calculate returns for both the base strategy and the volatility-targ
    df['targeted_strategy_return'] = df['adjusted_signal'] * df['gbpusd_return']
# Volatility-targeted return


    df['targeted_equity_curve'] = (1 + df['targeted_strategy_return']).cumprod()

    # Plot the results
    plt.figure(figsize=(14, 7))
    plt.plot(df['Date'], df['equity_curve'], label='Base Strategy', color='blue')
    plt.plot(df['Date'], df['targeted_equity_curve'], label='Volatility-Targeted S
    plt.xlabel('Date')
    plt.ylabel('Equity Curve')
    plt.title('Equity Curve Comparison: Base vs. Volatility-Targeted Strategy')
```

```
    plt.legend()
    plt.show()

advanced_backtest_strategy(merged_df, 0.05, 0.15, 20)
```

A.3 Step3 code

```

bbg_data_g10 = pd.read_excel('MP2.Data.From.BBG.xlsx', 'G10')

## extract GB and USD PPP data
ppp_g10 = ppp_df[['United States', 'United Kingdom', 'Germany',
                  'Australia', 'New Zealand', 'Sweden', 'Norway',
                  'Japan', 'Canada', 'Switzerland']]

# Adding GBPUSD back into exchange rates
gbpusd_spot = merged_df[['Date','GBPUSD']]
gbpusd_spot.rename(columns={'GBPUSD': 'GBPUSD Curncy'}, inplace=True)
gbpusd_spot['Dates'] = gbpusd_spot['Date']
gbpusd_spot.drop('Date', axis=1, inplace=True)
gbpusd_spot.head()

bbg_data_g10 = bbg_data_g10.merge(gbpusd_spot, on='Dates', how='left')

bbg_data_g10['Year'] = bbg_data_g10['Dates'].dt.year

# Merge bbg_data with ppp_us_gb on 'Year'
# Since ppp_g10 index is 'Year', reset it to use it as a column
ppp_g10 = ppp_g10.reset_index().rename(columns={'index': 'Year'})

# Perform the merge based on the 'Year' column
merged_df = bbg_data_g10.merge(ppp_g10, on='Year', how='left')

date_to_ppp_year.rename(columns={'Date': 'Dates'}, inplace=True)
```

```
merged_df = merged_df.merge(date_to_ppp_year, on='Dates', how='left')

### Finding the relative PPP (between 2 countries) to relate to the FX exchange

# the base rates are different, so we will need to calc PPP ratio appropriatly

# Foreign Base rates:
merged_df['PPP-EURUSD'] = merged_df['United States'] / merged_df['Germany']
merged_df['PPP-AUDUSD'] = merged_df['United States'] / merged_df['Australia']
merged_df['PPP-NZDUSD'] = merged_df['United States'] / merged_df['New Zealand']
merged_df['PPP-GBPUSD'] = merged_df['United States'] / merged_df['United Kingdom

# US Base rates:
merged_df['PPP-USDSEK'] = merged_df['Sweden'] / merged_df['United States']
merged_df['PPP-USDNOK'] = merged_df['Norway'] / merged_df['United States']
merged_df['PPP-USDJPY'] = merged_df['Japan'] / merged_df['United States']
merged_df['PPP-USDCAD'] = merged_df['Canada'] / merged_df['United States']
merged_df['PPP-USDCHF'] = merged_df['Switzerland'] / merged_df['United States']


### find percent diff between market and ppp

# foreign base rates
merged_df['PPP-EURUSD-Diff'] = (merged_df['EURUSD Curncy'] - merged_df['PPP-EURU
merged_df['PPP-AUDUSD-Diff'] = (merged_df['AUDUSD Curncy'] - merged_df['PPP-AUDU
merged_df['PPP-NZDUSD-Diff'] = (merged_df['NZDUSD Curncy'] - merged_df['PPP-NZDU
merged_df['PPP-GBPUSD-Diff'] = (merged_df['GBPUSD Curncy'] - merged_df['PPP-GBPU

# US base rates
merged_df['PPP-USDSEK-Diff'] = (merged_df['USDSEK Curncy'] - merged_df['PPP-USDS
merged_df['PPP-USDNOK-Diff'] = (merged_df['USDNOK Curncy'] - merged_df['PPP-USDN
merged_df['PPP-USDJPY-Diff'] = (merged_df['USDJPY Curncy'] - merged_df['PPP-USDJ
merged_df['PPP-USDCAD-Diff'] = (merged_df['USDCAD Curncy'] - merged_df['PPP-USDC
```

```
merged_df['PPP-USDCHF-Diff'] = (merged_df['USDCHF Curncy'] - merged_df['PPP-USDC

fx_names = ['EURUSD', 'AUDUSD', 'NZDUSD',
       'USDSEK', 'USDNOK', 'USDJPY', 'USDCAD',
       'USDCHF', 'GBPUSD']

def g10_backtest(df_orig, threshold, fx_names, plot=True, figsize=(16,8)):

  if plot:
    fig, axs = plt.subplots(figsize=figsize)

  df = df_orig.copy() # so df original isn't affected
  for fx in fx_names:
    df['signal'] = 0

    # Buy sell strat:
    df.loc[df[f'PPP-{fx}-Diff'] > threshold, 'signal'] = -1
    df.loc[df[f'PPP-{fx}-Diff'] < -threshold, 'signal'] = 1

    # Calculate daily returns from GBPUSD price changes
    df['return'] = df[f"{fx} Curncy"].pct_change()

    # Shift signal by one day to avoid look-ahead bias
    df[f'strategy_return'] = df['signal'].shift(1) * df['return']

    # Calculate the equity curve (cumulative returns) for the strategy
    df['equity_curve'] = (1 + df['strategy_return']).cumprod()
    if plot:
      sns.lineplot(data=df, x='Dates', y='equity_curve', label=fx)

  if plot:
    plt.title(f'Equity Curve of Base Strategy of G10')
    plt.xlabel('Date')
    plt.ylabel('Equity')
```

```
        plt.legend()
        plt.show()
    return None


g10_backtest(merged_df, 0.05, fx_names, figsize=(8,4))
g10_backtest(merged_df, 0.05, ['NZDUSD', 'AUDUSD'], figsize=(8,4)) # best perfor


def g10_advanced_backtest_strategy(df_orig, threshold, target_vol, fx_names, plo

    if plot:
        fig, axs = plt.subplots(figsize=figsize)

    df = df_orig.copy() # so df original isn't affected
    for fx in fx_names:



        df['signal'] = 0

        # Buy sell strat. Note this is same as simple but we will acct for vol
        df.loc[df[f'PPP-{fx}-Diff'] > threshold, 'signal'] = -1
        df.loc[df[f'PPP-{fx}-Diff'] < -threshold, 'signal'] = 1

        df['return'] = df[f'{fx} Curncy'].pct_change()

        # Step 2: Calculate rolling volatility
        df['volatility'] = df['return'].rolling(window).std() * np.sqrt(252)
# Annualized volatility

        # Step 3: Define target annual volatility (e.g., 10%) and calculate position
        df['adjusted_signal'] = ((target_vol / df['volatility']).clip(upper=2) * df[

        # Step 4: Calculate returns for both the base strategy and the volatility-ta
```

```
        df['targeted_strategy_return'] = df['adjusted_signal'] * df['return']
# Volatility-targeted return

        df['targeted_equity_curve'] = (1 + df['targeted_strategy_return']).cumprod()

        if plot:
            sns.lineplot(data=df, x='Dates', y='targeted_equity_curve', label=fx)


    if plot:
        plt.title('Equity Curve Comparison: Base vs. Volatility-Targeted Strategy')
        plt.xlabel('Date')
        plt.ylabel('Equity')
        plt.legend()
        plt.show()

g10_advanced_backtest_strategy(merged_df, 0.05, 0.15, fx_names, window=20, figsi

g10_advanced_backtest_strategy(merged_df, 0.05, 0.15, ['NZDUSD'], window=20, fig

g10_advanced_backtest_strategy(merged_df, 0.05, 0.15, ['USDSEK'], window=20, fig
print("Base strat:")
g10_backtest(merged_df, 0.05, ['USDSEK'], figsize=(8,4)) # best performers

import datetime

start_date = datetime.datetime(2014, 1, 1)
end_date = datetime.datetime(2015, 12, 31)
filtered_df = merged_df[(merged_df['Dates'] >= start_date) & (merged_df['Dates']

fig, ax1 = plt.subplots(figsize=(8, 4))

sns.lineplot(data=filtered_df, x='Dates', y='PPP-USDCHF', ax=ax1, color='blue')
ax1.set_ylabel("PPP-USDCHF", color='blue')
```

```
ax2 = ax1.twinx()
sns.lineplot(data=filtered_df, x='Dates', y='USDCHF Curncy', ax=ax2, color='red'
ax2.set_ylabel("USDCHF Curncy", color='red')

plt.title('PPP-USDCHF and USDCHF rate (Swiss Franc) for 2014-2015')
# fig.legend(loc="upper left", bbox_to_anchor=(0.1,0.9))
plt.show()


# Example of our log regress model
from sklearn.linear_model import LogisticRegression
fx = 'GBPUSD'

df_copy = merged_df.copy()

df_copy[f'{fx}_signal'] = 0
df_copy[f'{fx}_return'] = df_copy[f"{fx} Curncy"].pct_change()
df_copy.loc[df_copy[f'PPP-{fx}-Diff'] >= 0, f'{fx}_signal'] = 0  # Sell signal
df_copy.loc[df_copy[f'PPP-{fx}-Diff'] < 0, f'{fx}_signal'] = 1  # Buy signal


X = df_copy[[f'PPP-{fx}-Diff']].fillna(0)
y = df_copy[f'{fx}_signal']

model = LogisticRegression()
model.fit(X, y)

df_copy['Buy Probability'] = model.predict_proba(X)[:, 1]

df_copy['Sell Probability'] = model.predict_proba(1-X)[:, 1]


# plotting
```

```
ppp_diff_range = np.linspace(df_copy[f'PPP-{fx}-Diff'].min(), df_copy[f'PPP-{fx}

probabilities = model.predict_proba(ppp_diff_range)[:, 1] # prob of buy

plt.figure(figsize=(10, 6))
plt.plot(ppp_diff_range, probabilities, label='Buy Probability', color='green')
plt.plot(ppp_diff_range, 1-probabilities, label='Sell Probability', color='Red')

plt.title(f'Binary Logistic Regression Buy Probability for {fx}')
plt.xlabel(f'PPP-{fx}-Diff')
plt.ylabel('Probability of Buy Signal')
plt.legend()
plt.grid(True)
plt.show()



def g10_rel_backtest(df_orig, fx_names, threshold_pctile=y, figsize=(16,8)):
    '''
    This function creates a logistic regression model based on the PPP-spot pric
    difference for a given currency. It then creates buy or sell decision based
    on the strength of the buy signal from the log regression model. For example
    a buy signal of (0.5+threshold, 1) from our model will lead to a buy decisio


    We buy if the buy signal is in the top x% of all buy signals.
    We sell if the buy signal is in the bottom x% of all buy signals.

    where x = threshold_pct

    '''

    fig, axs = plt.subplots(figsize=figsize)

```

```
    df = df_orig.copy()

    for fx in fx_names:

      # here we tell the model that a negative (undervalued) number means buy an
      # a positive number means sell.

      df[f'value'] = 0
      df[f'return'] = df[f"{fx} Curncy"].pct_change()
      df.loc[df[f'PPP-{fx}-Diff'] >= 0, f'value'] = 0  # overvalued signal
      df.loc[df[f'PPP-{fx}-Diff'] < 0, f'value'] = 1  # undervalued signal

      X = df[[f'PPP-{fx}-Diff']].fillna(0)
      y = df[f'value']

      model = LogisticRegression()
      model.fit(X, y)

      df['Buy Probability'] = model.predict_proba(X)[:, 1]


      # This is where we decide to buy sell or hold

      # hold by default
      df['signal'] = 0

      df['signal'] = np.where(
          df['Buy Probability'] > 1-threshold_pctile, 1,  # Set to 1 if Buy Pro
          np.where(df['Buy Probability'] < threshold_pctile, -1, 0)  # Set to -1
      )


      df['return'] = df[f"{fx} Curncy"].pct_change()
      df['strategy_return'] = df['signal'].shift(1) * df['return']
```

```
        df['equity_curve'] = (1 + df['strategy_return']).cumprod()

        sns.lineplot(data=df, x='Dates', y='equity_curve', label=fx)

    plt.title('Equity Curve, Logistic Regression decision making')
    plt.xlabel('Date')
    plt.ylabel('Equity')
    plt.legend
    plt.show()

g10_rel_backtest(merged_df, fx_names, threshold_pctile=0.20, figsize=(16,8))

def g10_rel_advanced_backtest(df_orig, fx_names, target_vol=0.15, window=20, thr
    '''
    This function creates a logistic regression model based on the PPP-spot pric
    difference for a given currency. It then creates buy or sell decision based
    on the strength of the buy signal from the log regression model. For example
    a buy signal of (0.5+threshold, 1) from our model will lead to a buy decisio


    We buy if the buy signal is in the top x% of all buy signals.
    We sell if the buy signal is in the bottom x% of all buy signals.

    where x = threshold_pct

    '''

    fig, axs = plt.subplots(figsize=figsize)

    df = df_orig.copy()

    for fx in fx_names:

        # here we tell the model that a negative (undervalued) number means buy an
```

```
    # a positive number means sell.

    df[f'value'] = 0
    df[f'return'] = df[f"{fx} Curncy"].pct_change()
    df.loc[df[f'PPP-{fx}-Diff'] >= 0, f'value'] = 0  # overvalued signal
    df.loc[df[f'PPP-{fx}-Diff'] < 0, f'value'] = 1  # undervalued signal

    X = df[[f'PPP-{fx}-Diff']].fillna(0)
    y = df[f'value']

    model = LogisticRegression()
    model.fit(X, y)

    df['Buy Probability'] = model.predict_proba(X)[:, 1]



    # This is where we decide to buy sell or hold

    # hold by default
    df['signal'] = 0

    df['signal'] = np.where(
        df['Buy Probability'] > 1-threshold_pctile, 1,  # Set to 1 if Buy Pro
        np.where(df['Buy Probability'] < threshold_pctile, -1, 0)  # Set to -1
    )



    # Relative volatility and trade size calc
    df['return'] = df[f"{fx} Curncy"].pct_change()
    df['volatility'] = df['return'].rolling(window).std() * np.sqrt(252)
    df['adjusted_signal'] = ((target_vol / df['volatility']).clip(upper=2) * c

    df['strategy_return'] = df['adjusted_signal'] * df['return']  # Volatility
    df['equity_curve'] = (1 + df['strategy_return']).cumprod()
```

```

        sns.lineplot(data=df, x='Dates', y='equity_curve', label=fx)

    plt.title('Equity Curve, Volatility Targetting by logistic regression decisi
    plt.xlabel('Date')
    plt.ylabel('Equity')
    plt.legend
    plt.show()

g10_rel_advanced_backtest(merged_df, fx_names, target_vol=0.15, window=20, thres

def g10_rel_advanced_backtest_2(df_orig, fx_names, target_vol=0.15, window=20, f
    '''
    This function creates a logistic regression model based on the PPP-spot pric
    difference for a given currency. It then creates buy or sell decision based
    on the strength of the buy signal from the log regression model. For example
    a buy signal of (0.5+threshold, 1) from our model will lead to a buy decisio


    We buy if the buy signal is in the top x% of all buy signals.
    We sell if the buy signal is in the bottom x% of all buy signals.

    where x = threshold_pct

    '''

    fig, axs = plt.subplots(figsize=figsize)

    df = df_orig.copy()

    for fx in fx_names:

        # here we tell the model that a negative (undervalued) number means buy an
        # a positive number means sell.
```

```
        df[f'value'] = 0
        df[f'return'] = df[f"{fx} Curncy"].pct_change()
        df.loc[df[f'PPP-{fx}-Diff'] >= 0, f'value'] = 0  # overvalued signal
        df.loc[df[f'PPP-{fx}-Diff'] < 0, f'value'] = 1  # undervalued signal

        X = df[[f'PPP-{fx}-Diff']].fillna(0)
        y = df[f'value']

        model = LogisticRegression()
        model.fit(X, y)

        df['Buy Probability'] = model.predict_proba(X)[:, 1]

        # This is where we decide to buy sell or hold

        # hold by default
        df['signal'] = (df['Buy Probability'] - 0.5) * 2  # transforms from [0, 1]

        # Relative volatility and trade size calc
        df['return'] = df[f"{fx} Curncy"].pct_change()
        df['volatility'] = df['return'].rolling(window).std() * np.sqrt(252)
        df['adjusted_signal'] = ((target_vol / df['volatility']).clip(upper=2) * d

        df['strategy_return'] = df['adjusted_signal'] * df['return']  # Volatility
        df['equity_curve'] = (1 + df['strategy_return']).cumprod()

        sns.lineplot(data=df, x='Dates', y='equity_curve', label=fx)

    plt.title('Equity Curve, Volatility Targetting and Logistic Regression Lot S
    plt.xlabel('Date')
    plt.ylabel('Equity')
    plt.legend
    plt.show()
```

```
g10_rel_advanced_backtest_2(merged_df, fx_names, target_vol=0.15, window=20, fig


threshold_pctile=0.25 # how "picky" we are at selecting to buy or sell
target_vol=0.15 # what vol we're looking for
window=50 # how many days back we look

# fx_names = ['EURUSD', 'AUDUSD', 'NZDUSD',
#            'USDSEK', 'USDNOK', 'USDJPY', 'USDCAD',
#            'USDCHF', 'GBPUSD']

fx_names = ['EURUSD', 'AUDUSD', 'NZDUSD']


g10_rel_backtest(merged_df, fx_names, threshold_pctile, figsize=(8,4))
g10_rel_advanced_backtest(merged_df, fx_names, target_vol, window, threshold_pct
```

A.4 Step4 code

```
! pip install fredapi

from fredapi import Fred
import pandas as pd

fred = Fred(api_key='321d7bf2c5feafe84bc8c7b84a252cdc')


# Dictionary to store country codes and their corresponding FRED series IDs for
interest_rate_series = {
    'Australia': 'IR3TIB01AUM156N',
    'Canada': 'IR3TIB01CAM156N',
    'European Union': 'IR3TIB01EZM156N',
    'Japan': 'IR3TIB01JPM156N',
```

```
    'New Zealand': 'IR3TIB01NZM156N',# Example: 3-Month or 10-Year bond yield
    'Norway': 'IR3TIB01NOM156N',
    'United Kingdom': 'IR3TIB01GBM156N',  # Example: 3-Month or 10-Year bond yie
    'Sweden': 'IR3TIB01SEM156N',      # Example: 3-Month or 10-Year bond yield
    'Switzerland': 'IR3TIB01CHM156N',# Example: 3-Month or 10-Year bond yield
    'United States': 'DGS10'          # 10-Year Treasury yield as a nominal inter
}

# Fetch interest rate data for each country and store in a dictionary
interest_rate_data = {}

for country, series_id in interest_rate_series.items():
    try:
        data = fred.get_series(series_id, observation_start='1990-01-01')
        interest_rate_data[country] = data
    except Exception as e:
        print(f"Error fetching data for {country}: {e}")

# Convert the data into a unified DataFrame
df_interest_rates = pd.DataFrame(interest_rate_data)

df_interest_rates.head(100)

# add _IR to the name of the columns of df_interest_rates
df_interest_rates.columns = [df_interest_rates.columns[0]] + [col + '_IR' for co
df_interest_rates.head()

cab = pd.read_csv('current_account_balance.csv', skiprows=2)
# get rid of time
cab['Category'] = pd.to_datetime(cab['Category']).dt.date
new_metric_df = merged_df.copy()

# add the cab data to the new_metric_df matched by their time
new_metric_df['Dates'] = pd.to_datetime(new_metric_df['Dates'], errors='coerce')
```

```
cab['Category'] = pd.to_datetime(cab['Category'], errors='coerce')


if new_metric_df['Dates'].isnull().any() or cab['Category'].isnull().any():
    print("There are non-date values in 'Dates' or 'Category' columns that have b


# Proceed with the merge after sorting
new_metric_df = new_metric_df.sort_values('Dates')
cab = cab.sort_values('Category')


# Merge using pd.merge_asof with backward direction
new_metric_df = pd.merge_asof(
    new_metric_df,
    cab,
    left_on='Dates',
    right_on='Category',
    direction='backward'
)
new_metric_df.head()

new_metric_df.drop(columns=['Category'], inplace=True)
new_metric_df.head()

# Forward-fill missing values in df_interest_rates if necessary
df_interest_rates = df_interest_rates.fillna(method='ffill')

# Sort both dataframes by their date columns (necessary for merge_asof)
new_metric_df = new_metric_df.sort_values('Dates')
df_interest_rates = df_interest_rates.sort_values('Date')

# Perform the merge using backward direction
new_metric_df = pd.merge_asof(
```

```
    new_metric_df,
    df_interest_rates,
    left_on='Dates',
    right_on='Date',
    direction='backward'
)

new_metric_df.tail(5)

fx_names = ['EURUSD', 'AUDUSD', 'NZDUSD',
        'USDSEK', 'USDNOK', 'USDJPY', 'USDCAD',
        'USDCHF', 'GBPUSD']

cab.head()

new_metric_df.columns

# List of columns to keep
columns_to_keep = [
    'Dates', 'EURUSD Curncy', 'AUDUSD Curncy', 'NZDUSD Curncy',
    'USDSEK Curncy', 'USDNOK Curncy', 'USDJPY Curncy', 'USDCAD Curncy',
    'USDCHF Curncy', 'GBPUSD Curncy',
    'PPP-EURUSD-Diff', 'PPP-AUDUSD-Diff', 'PPP-NZDUSD-Diff',
    'PPP-GBPUSD-Diff', 'PPP-USDSEK-Diff', 'PPP-USDNOK-Diff', 'PPP-USDJPY-Diff',
    'PPP-USDCAD-Diff', 'PPP-USDCHF-Diff', 'New Zealand_y', 'Canada_y',
    'Norway_y', 'Australia_y', 'Japan_y', 'United Kingdom_y', 'Sweden_y',
    'EU 27 since 2020', 'United States_y', 'Date', 'Australia_IR', 'Canada_IR',
    'European Union_IR', 'Japan_IR', 'New Zealand_IR', 'Norway_IR',
    'United Kingdom_IR', 'Sweden_IR', 'Switzerland_IR', 'United States_IR'
]

# Create a new DataFrame with only the specified columns
filtered_df = new_metric_df[columns_to_keep]
filtered_df.head()
```

```

# rename EU 27 since 2020 to Euro
filtered_df.rename(columns={'EU 27 since 2020': 'European Union'}, inplace=True)
filtered_df.head()

# rename EU 27 since 2020 to Euro
filtered_df.rename(columns={'European Union':'European Union_y'}, inplace=True)
filtered_df.columns

filtered_df.drop(columns=['Date'], inplace=True)
filtered_df.columns

switzerland = pd.read_csv('switzerlandCAB.csv', skiprows= 2)
switzerland.head()

switzerland['Category'] = pd.to_datetime(switzerland['Category'], errors='coerce
switzerland.head()

switzerland.rename(columns={'Switzerland': 'Switzerland_y'}, inplace=True)
switzerland.head()

#Forward-fill missing values in df_interest_rates if necessary
switzerland = switzerland.fillna(method='ffill')

# Sort both dataframes by their date columns (necessary for merge_asof)
filtered_df = filtered_df.sort_values('Dates')
switzerland = switzerland.sort_values('Category')

# Perform the merge using backward direction
filtered_df = pd.merge_asof(
    filtered_df,
    switzerland,
    left_on='Dates',
    right_on='Category',
```

```
    direction='backward'
)

filtered_df.tail(5)

filtered_df.columns

# Manually mapping currency codes to actual column names in filtered_df
currency_columns = {
    'EUR': ('PPP-EURUSD-Diff', 'European Union_IR', 'European Union_y'),
    'AUD': ('PPP-AUDUSD-Diff', 'Australia_IR', 'Australia_y'),
    'NZD': ('PPP-NZDUSD-Diff', 'New Zealand_IR', 'New Zealand_y'),
    'GBP': ('PPP-GBPUSD-Diff', 'United Kingdom_IR', 'United Kingdom_y'),
    'SEK': ('PPP-USDSEK-Diff', 'Sweden_IR', 'Sweden_y'),
    'NOK': ('PPP-USDNOK-Diff', 'Norway_IR', 'Norway_y'),
    'JPY': ('PPP-USDJPY-Diff', 'Japan_IR', 'Japan_y'),
    'CAD': ('PPP-USDCAD-Diff', 'Canada_IR', 'Canada_y'),
    'CHF': ('PPP-USDCHF-Diff', 'Switzerland_IR', 'Switzerland_y'),
    'USD': ('PPP-USDUSD-Diff', 'United States_IR', 'United States_y')
}

# Dictionary to store each new NEO-XXXUSD metric column
neo_metrics = {}

# Loop through each entry to create and store the NEO-XXXUSD metric
for code, (ppp_col, ir_col, cab_col) in currency_columns.items():
    # Calculate the NEO-XXXUSD metric and store it in the dictionary
    if ppp_col == 'PPP-USDUSD-Diff':  # Special case for USD (using 1 for PPP va
        neo_metrics[f'NEO-{code}USD'] = (
            0.33 * 1 +
            0.33 * filtered_df[ir_col] +
            0.34 * filtered_df[cab_col]
        )
    elif code in ['EUR', 'AUD', 'NZD', 'GBP']:
```

```
            neo_metrics[f'NEO-{code}USD'] = (
                0.33 * filtered_df[ppp_col] +
                0.33 * filtered_df[ir_col] +
                0.34 * filtered_df[cab_col]
            )
        else:
            neo_metrics[f'NEO-USD{code}'] = (
                0.33 * filtered_df[ppp_col] +
                0.33 * filtered_df[ir_col] +
                0.34 * filtered_df[cab_col]
            )

# Create a new DataFrame from the dictionary and include the Dates column
neo_metrics_df = pd.DataFrame(neo_metrics)
neo_metrics_df['Dates'] = filtered_df['Dates']  # Add Dates column

# add the currency columns
neo_metrics_df['EURUSD Curncy'] = filtered_df['EURUSD Curncy']
neo_metrics_df['AUDUSD Curncy'] = filtered_df['AUDUSD Curncy']
neo_metrics_df['NZDUSD Curncy'] = filtered_df['NZDUSD Curncy']
neo_metrics_df['USDSEK Curncy'] = filtered_df['USDSEK Curncy']
neo_metrics_df['USDNOK Curncy'] = filtered_df['USDNOK Curncy']
neo_metrics_df['USDJPY Curncy'] = filtered_df['USDJPY Curncy']
neo_metrics_df['USDCAD Curncy'] = filtered_df['USDCAD Curncy']
neo_metrics_df['USDCHF Curncy'] = filtered_df['USDCHF Curncy']
neo_metrics_df['GBPUSD Curncy'] = filtered_df['GBPUSD Curncy']

neo_metrics_df.head()

# Min-Max normalization
neo_cols = [col for col in neo_metrics_df.columns if col.startswith('NEO-')]

# Apply Min-Max scaling to each NEO column
for col in neo_cols:
```

```
    min_val = neo_metrics_df[col].min()
    max_val = neo_metrics_df[col].max()
    neo_metrics_df[col] = 2 * ((neo_metrics_df[col] - min_val) / (max_val - min_

fx_names = ['EURUSD', 'AUDUSD', 'NZDUSD', 'USDSEK', 'USDNOK', 'USDJPY', 'USDCAD'

def g10_backtest_neo_metrics(df_orig, threshold, fx_names, plot=True, figsize=(

    if plot:
        fig, axs = plt.subplots(figsize=figsize)

    df = df_orig.copy()  # so df original isn't affected
    for fx in fx_names:
        df['signal'] = 0

        # Buy sell strat:
        # Use normalized NEO metric for signals
        df.loc[df[f'NEO-{fx}'] > threshold, 'signal'] = -1
        df.loc[df[f'NEO-{fx}'] < -threshold, 'signal'] = 1

        # Calculate daily returns from the fx price changes
        df['return'] = df[f"{fx} Curncy"].pct_change()

        # Shift signal by one day to avoid look-ahead bias
        df[f'strategy_return'] = df['signal'].shift(1) * df['return']

        # Calculate the equity curve (cumulative returns) for the strategy
        df['equity_curve'] = (1 + df['strategy_return']).cumprod()
        if plot:
            sns.lineplot(data=df, x='Dates', y='equity_curve', label=fx)

    if plot:
        plt.title(f'Equity Curve of New Metric Strategy of G10')
        plt.xlabel('Date')
```

```
            plt.ylabel('Equity')
            plt.legend()
            plt.show()

    return None

g10_backtest_neo_metrics(neo_metrics_df, 0.2, fx_names, figsize=(16,8))
```

A.5 Step5 code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression


series_data = pd.read_excel('MP2.Data.From.BBG.xlsx', sheet_name='686Trend')
series_data.columns = ['Date', 'Series 1', 'Series 2']
series_data['Date'] = pd.to_datetime(series_data['Date'])
series_data = series_data.sort_values('Date').set_index('Date')

# split training and testing data
series_1 = series_data[['Series 1']].rename(columns={'Series 1': 'Price'}).copy(
series_2 = series_data[['Series 2']].rename(columns={'Series 2': 'Price'}).copy(

series_1.head(), series_2.head()

# Trend-Based metric (TVI)
def calculate_tvi_adapted(df, momentum_period=14, persistence_window=30, positio
    # Calculate Momentum for trend strength
    df['Momentum'] = df['Price'].diff(momentum_period)

    # Calculate Trend Persistence (R-squared of a rolling linear regression)
```

```
    r_squared_list = []
    for i in range(len(df)):
        if i < persistence_window:
            r_squared_list.append(np.nan)
        else:
            y = df['Price'].iloc[i - persistence_window:i].values.reshape(-1, 1)
            X = np.arange(persistence_window).reshape(-1, 1)
            model = LinearRegression().fit(X, y)
            r_squared = model.score(X, y)
            r_squared_list.append(r_squared)
    df['Trend_Persistence'] = r_squared_list

    # Calculate Price Position relative to a long-term moving average
    df['SMA_long'] = df['Price'].rolling(window=position_window).mean()
    df['Price_Position'] = (df['Price'] - df['SMA_long']) / df['SMA_long']

    # Fill NaN values where necessary
    df[['Momentum', 'Trend_Persistence', 'Price_Position']] = df[['Momentum', 'T

    # Calculate TVI as a weighted sum of the components
    w1, w2, w3 = 0.33, 0.33, 0.34  # weights for Momentum, Trend Persistence, an
    df['TVI'] = w1 * df['Momentum'] + w2 * df['Trend_Persistence'] + w3 * df['Pr

    return df

# Backtesting function using the calculated TVI
def backtest_tvi_strategy(df, buy_threshold=0.5, sell_threshold=-0.5):
    # Generate buy/sell signals based on TVI
    df['signal'] = 0
    df.loc[df['TVI'] > buy_threshold, 'signal'] = 1  # Buy signal
    df.loc[df['TVI'] < sell_threshold, 'signal'] = -1  # Sell signal

    # Calculate returns
    df['price_return'] = df['Price'].pct_change()
```

```
    df['strategy_return'] = df['signal'].shift(1) * df['price_return']
    df['equity_curve'] = (1 + df['strategy_return']).cumprod()

    # Plot the equity curve
    plt.figure(figsize=(12, 6))
    sns.lineplot(data=df, x=df.index, y='equity_curve', label='Trend-Based Strat
    plt.title('Equity Curve of Trend-Based Strategy')
    plt.xlabel('Date')
    plt.ylabel('Equity Curve')
    plt.legend()
    plt.show()

    return df

# Apply TVI Calculation and Backtest on Series 1 (Training Data)
series_1_tvi = calculate_tvi_adapted(series_1.copy())
series_1_results = backtest_tvi_strategy(series_1_tvi, buy_threshold=0.5, sell_t

# Apply TVI Calculation and Backtest on Series 2 (Out-of-Sample Testing)
series_2_tvi = calculate_tvi_adapted(series_2.copy())
series_2_results = backtest_tvi_strategy(series_2_tvi, buy_threshold=0.5, sell_t
```