The Influence of Macroeconomic Indicators on Money Velocity

Video link: https://drive.google.com/file/d/1bWukthUfzJhvWKvsbcRfdKPQBOl9Ng-p/view?usp=drive_link

Permissions

Place an `X` in the appropriate bracket below to specify if you would like your group's project to be made available to the public. (Note that student names will be included (but PIDs will be scraped from any groups who include their PIDs).

- [X] YES - make available
- [ ] NO - keep private

Names

- Jingwei Guo
- Sinclair Lim
- Alejandro Vazquez
- Alisa Vu
- David Yonemura

Abstract

Our team chose to work on this topic in order to determine which factors has the most influence on money velocity in the United States between the years 2000 and 2020. This helps us better understand the dynamics of our economy and gain valuable insights into the drivers of economic performance during the period. It also reinforces the importance of considering diverse macroeconomic indicators when analyzing money velocity. This can be meaningful data to various stakeholders such as policymakers, investors and economists so that they are able to make a more informed decision and better understand the macroeconomics in play in shaping the US economy over these two

decades. By employing techniques such as linear regression and block bootstrapping to analyze historical data, we were able to analyze a range of variables. We discovered that the Economic Policy Uncertainty Index (EPU) was the most statistically significant variable in modeling money velocity, revealing the relationship between heightened economic policy uncertainty and decreased money velocity. Through this research and comparison between a variety of variables, we can highlight the interplay between economic policies, public perception, and media influences in shaping money velocity.

Research Question

Which of the following factors had the most influence on money velocity in the United States from 2000-2020: unemployment rate, interest rate, corporate profits after tax, the S&P 500, or economic policy uncertainty index?

Background and Prior Work

Money velocity is a crucial indicator of economic activity and represents the frequency with which a unit of money is used to buy domestically-produced goods and services within a specific period. For example, a money velocity of 2 measured over one quarter means that each dollar in the economy is used for an average of 2 transactions per quarter. There are a variety of factors that may influence the velocity of money, from the unemployment and inflation rates to stock performance and political stability. Understanding which factors most significantly impact money velocity in the U.S. can provide valuable insights for policymakers, investors, economists, and consumers alike. More specifically, this metric can help us understand the effectiveness of monetary policy changes, presence of liquidity traps, inflationary pressures, consumer confidence, shifts in savings habits, and overall efficiency of the financial system. It is a valuable indicator because it offers deeper insights into societal well-being, a dimension that other indicators such as Gross Domestic Product (GDP) often fail to account for.

Research shows that money velocity might be influenced by the "wealth effect", which suggests that consumers spend more money when their wealth increases even if their income does not. Poterba (2000) explored this effect by examining how fluctuations in stock prices influenced consumer spending. They found that an appreciating stock market increases the net worth of households, leading to potential increases in spending, which implies a higher velocity of money[1]. Moreover, a 2003 study by Stock and Watson found that there is a modest relationship between stock returns and output

growth but little evidence that asset prices can predict inflation, giving us insight into the effect of the stock market on output and inflation (two key components of money velocity)[2].

Further research has been done on how the interest rate affects money velocity. Hetzel (1993) found that money velocity became more sensitive to changes in the interest rate during the 1980s, concluding that the velocity of money increased with higher interest rates because the opportunity cost of holding non-interest-bearing assets was higher[3]. Another more recent study by Roman Kozlov (2023) confirmed that interest rates and consumption are inversely related[4]. This has important implications for money velocity because if consumers are spending less, then velocity will be lower all else being equal. This allows stakeholders to uncover their precise impact on money velocity, advancing our understanding of economics indicators. With the various factors affecting the velocity of money at play, our group aims to quantify how significant these influences are.

1. ^ Anderson, R., Bordo, M., & Duca, J. (2016). Money and velocity during financial crises: From the great depression to the great recession (w22100; p. w22100). National Bureau of Economic Research. https://doi.org/10.3386/w22100
2. ^ Ireland, P. N. (1991). Financial evolution and the long-run behavior of velocity: New evidence from us regional data [SSRN Scholarly Paper]. https://papers.ssrn.com/abstract=2126169
3. ^ Stock, J. H., & Watson, M. W. (2003). Forecasting output and inflation: The role of asset prices. Journal of Economic Literature, 41(3), 788–829. https://www.princeton.edu/~mwatson/papers/Stock_Watson_JEL_2003.pdf
4. ^ Kozlov, R. (2023). The effect of interest rate changes on consumption: An age-structured approach. Economies, 11(1), 23. https://doi.org/10.3390/economies11010023

Hypothesis

Among unemployment rate, interest rate, corporate profits after tax, the S&P 500, and economic policy uncertainty index, we hypothesize that interest rate will have the most influence on money velocity in the United States between the years 2000-2020. This hypothesis is formulated based on the understanding that interest rates can directly impact borrowing costs, which can incentivize or disincentivize consumer spending and investment, and therefore significantly affect the speed at which money circulates within the economy. For instance, in a case where interest rates are higher, we would expect

decreased consumer spending in addition to a lower money velocity.

Data

The ideal dataset would be one large dataset, formatted in a semi-structured form such as csv or JSON, that included the factors of interest: unemployment rate, interest rate, and economic policy uncertainty index and the dependent variable, money velocity. The rows would be multi-indexed first by years then by quarter, so the variables of the observations can easily be compared with each other. For observation count, we are hoping there are observations for each year and each quarter within the year. That means we are ideally looking for 100 observations with no missing values. This dataset would preferably be already collected by a trustworthy source, such as the United States Department of the Treasury, since this type of information would already be of interest to this department. This also entails that it is free for the public to download and analyze, as it would be simpler than trying to webscrape.

While searching for real datasets, we came across several trustworthy websites, such as but not limited to: the Federal Reserve Economic Data or The World Bank. Given that across these websites, we have found all the data necessary for our research, our data is certainly obtainable and in the format desired. However, this data spans across different tables, which means the data will have to be merged together on our part, rather than simply found in one large dataset. Additionally, some datasets are not all in the same time series, where the frequency of the data is collected monthly, daily, quarterly, or even yearly, which means we will also have to tidy this data to keep all the variables in the same time series. Besides this, our data is mostly ideal.

Data overview

Dataset 1: Money Velocity

- **Calculation**: Money velocity of M2 Money Stock= Quarterly Nominal GDP / Quarterly Average of M2 Money Stock
- **Definition of Money velocity**: Money Velocity measures how frequently currency is used for domestic purchases. A rising velocity suggests increased economic transactions, indicating active spending. Monitoring velocity provides insights into consumer and business spending behavior (*Federal Reserve Bank of St. Louis*).
- **Definition of Nominal GDP**: Nominal Gross Domestic Product is the total market

value of all goods and services produced in a country's economy over a given period. Unlike other GDP measurements, nominal GDP is not adjusted to account for price changes from inflation and deflation (*Corporate Finance Institute*).

- **Definition of M2 Money Stock**: M1, M2, and MZM represent the spectrum of money supply. M2, since May 2020, includes M1 plus small-denomination time deposits and balances in retail MMFs. Seasonally adjusted M2 combines savings deposits, small-denomination time deposits, and retail MMFs. M1, the narrowest component, encompasses currency in circulation, traveler's checks, demand, and checkable deposits. A declining M1 velocity suggests reduced short-term consumption transactions (*Federal Reserve Bank of St. Louis*).
- **Dataset Name**: Velocity of M2 Money Stock
- **Link to the dataset**: https://fred.stlouisfed.org/series/M2V (*Federal Reserve Bank of St. Louis*).
- **Number of observations**: 259
- **Number of variables**: 2

Dataset 2: Federal Funds Effective Rate (Interest Rate)

- **Definition of The Federal Funds Effective Rate (Interest Rate)**: The Federal Funds Effective Rate is the weighted average interest rate at which depository institutions trade federal funds overnight, reflecting the cost of borrowing for banks. Determined through negotiations between lending and borrowing banks, it is influenced by the Federal Reserve's open market operations, impacting broader interest rates and serving as a key indicator in the U.S. financial market (*Board of Governors of the Federal Reserve System*).
- **Dataset Name**: Federal Funds Effective Rate (Interest Rate)
- **Link to the dataset**: https://fred.stlouisfed.org/series/FEDFUNDS (*Board of Governors of the Federal Reserve System*).
- **Number of observations**: 832
- **Number of variables**: 2

Dataset 3: Unemployment Rate

- **Definition of Unemployment Rate**: The unemployment rate represents the number of unemployed as a percentage of the labor force. Labor force data are restricted to people 16 years of age and older, who currently reside in 1 of the 50 states or the District of Columbia, who do not reside in institutions (e.g., penal and mental facilities, homes for the aged), and who are not on active duty in the Armed Forces (*U.S. Bureau of Labor Statistics*).

- **Dataset Name**: Unemployment Rate
- **Link to the dataset**: https://fred.stlouisfed.org/series/UNRATE (*U.S. Bureau of Labor Statistics*).
- **Number of observations**: 302
- **Number of variables**: 2

Dataset 4: Corporate Profits After Tax in Billions

- **Definition of Corporate Profits After Tax**: Profits after tax without IVA and CCAdj is equal to Corporate Profit Before Tax (Book Profit) less taxes on corporate income. It consists of net dividends and undistributed corporate profits (U.S. Bureau of Economic Analysis).
- **Dataset Name**: Corporate Profits After Tax (without IVA and CCAdj)
- **Link to the dataset**: https://fred.stlouisfed.org/series/CP (*U.S. Bureau of Economic Analysis*).
- **Number of observations**: 306
- **Number of variables**: 2

Dataset 5: Economic Policy Uncertainty Index

- **Definition of Economic Policy Uncertainty Index**: The Economic Policy Uncertainty Index is an index measuring the uncertainty level of economic policies founded by economic professors Scott R. Baker at Northwestern University, Nick Bloom at Stanford University, and Steven J. Davis at University of Chicago (*Economic Policy Uncertainty*).
- **Construction of Economic Policy Uncertainty Index**: The index is constructed using three components. The first component measures newspaper coverage of policy-related economic uncertainty, utilizing search results from 10 major U.S. newspapers. The second component incorporates data from the Congressional Budget Office on expiring federal tax code provisions. The third component utilizes the Federal Reserve Bank of Philadelphia's Survey of Professional Forecasters, gauging dispersion among forecasters' predictions on key macroeconomic variables to create indices of uncertainty about policy-related factors (*Economic Policy Uncertainty*).
- **Dataset Name**: Economic Policy Uncertainty Index
- **Link to the dataset**: https://www.policyuncertainty.com/index.html (*Economic Policy Uncertainty*).
- **Number of observations**: 467
- **Number of variables**: 4

Dataset 6: Quarterly Change in S&P500

- **Definition of Quarterly Change in S&P500**: The S&P 500 is a stock market index that tracks the performance of 500 large companies listed on stock exchanges in the United States. The Quarterly Change in S&P 500, a leading U.S. stock market index, reflects the percentage difference in its value every quarter.
- **Dataset Name**: Quarterly Change in S&P500
- **Link to the dataset**: https://www.nasdaq.com/market-activity/index/spx/historical?page=1&rows_per_page=10&timeline=y10 (*National Association of Securities Dealers Automated Quotations*).
- **Number of observations**: 246
- **Number of variables**: 2

Each dataset holds crucial information about our project because they are our independent (Money Velocity) and dependent variables (the remaining) that we want to study. All of our datasets follow a straightforward structure, since every data point includes both a date and a numerical value.

To process the data, we sourced the datasets from various reliable sources as shown above. We first ensured the numerical data adhered to the correct data type, such as integer or float. Then, we cleaned up the data by imputing them with binomial random values, leveraging the data of the surrounding neighbours.

We intend to utilise these datasets of the dependent variables to compare how significant they are to the independent variable and whether our hypothesis is true through time series analysis. We also decided to combine these data using a common time variable and in our case, it will be quartely. Because of this, we aggregated the data across days or months (if needed) into quarterly points so that each dataset is indexed by the same set of dates, taking the mean of the aggregated values.

Imports

In [ ]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from matplotlib.colors import LinearSegmentedColormap
import statsmodels.api as sm
from scipy import stats
import statsmodels.formula.api as smf
```

Money Velocity

In [ ]:
```
# data downloaded from: https://fred.stlouisfed.org/series/M2V
df_money_velocity = pd.read_csv('M2V.csv')

# convert date column to datetime format
df_money_velocity['DATE'] = pd.to_datetime(df_money_velocity['DATE'])

# ensure data is sorted by date
df_money_velocity_mv = df_money_velocity.sort_values(by='DATE')

# Filter data to only include data from 2000 to 2020
df_money_velocity = df_money_velocity[(df_money_velocity['DATE'].dt.year >=
                                       & (df_money_velocity['DATE'].dt.year <
df_money_velocity.reset_index(drop=True, inplace=True)

cleaned_df_money_velocity = df_money_velocity.set_index('DATE')

cleaned_df_money_velocity
```

Out[ ]:

DATE	M2V
2000-01-01	2.135
2000-04-01	2.151
2000-07-01	2.141
2000-10-01	2.133
2001-01-01	2.085
...	...
2019-10-01	1.438
2020-01-01	1.391
2020-04-01	1.128
2020-07-01	1.176
2020-10-01	1.163

84 rows × 1 columns

Federal Funds Effective Rate (Interest Rate)

In [ ]:
```
df_fedfunds = pd.read_csv('FEDFUNDS.csv')

# convert string to date time format, then filter for years before 2020 afte
df_fedfunds['DATE'] = pd.to_datetime(df_fedfunds['DATE'])
df_fedfunds = df_fedfunds[(df_fedfunds['DATE'].dt.year >= 1999) & (df_fedfur

# a function to clean up by quarters
def filterbyquarters(df_fedfunds_Q, start_month, end_month):

    df_fedfunds_QX = df_fedfunds_Q[df_fedfunds_Q['DATE'].dt.month.between(st
    df_fedfunds_QX = df_fedfunds_QX.groupby(df_fedfunds_QX['DATE'].dt.year).
    df_fedfunds_QX.reset_index(drop=True, inplace=True)
    df_fedfunds_QX['DATE'] = df_fedfunds_QX['DATE'].dt.date # change to date
    df_fedfunds_QX['DATE'] = df_fedfunds_QX['DATE'].apply(lambda x: x.replac
    df_fedfunds_QX['DATE'] = df_fedfunds_QX['DATE'].apply(lambda x: x.replac
    df_fedfunds_QX = df_fedfunds_QX.round(2) # round the value off to 2 deci

    return df_fedfunds_QX

# perform cleaning up using above function on all four quarters
df_Q1 = filterbyquarters(df_fedfunds, 1, 3)
df_Q2 = filterbyquarters(df_fedfunds, 4, 6)
df_Q3 = filterbyquarters(df_fedfunds, 7, 9)
df_Q4 = filterbyquarters(df_fedfunds, 10, 12)

# merge the dataframes vertically
df_fedfunds = pd.concat([df_Q1, df_Q2, df_Q3, df_Q4])

# sort the df by year
df_fedfunds = df_fedfunds.sort_values(by='DATE')
df_fedfunds['DATE'] = pd.to_datetime(df_fedfunds['DATE'])
cleaned_df_fedfunds = df_fedfunds.set_index('DATE')
cleaned_df_fedfunds.rename(columns = {'FEDFUNDS':'Federal Funds Effective Ra

cleaned_df_fedfunds = cleaned_df_fedfunds.drop(cleaned_df_fedfunds.index[:3]
cleaned_df_fedfunds
```

Out[ ]:

DATE	Federal Funds Effective Rate (Interest Rate) (%)
1999-10-01	5.31
2000-01-01	5.68
2000-04-01	6.27
2000-07-01	6.52
2000-10-01	6.47
...	...
2019-10-01	1.64
2020-01-01	1.26
2020-04-01	0.06
2020-07-01	0.09
2020-10-01	0.09

85 rows × 1 columns

Unemployment Rate

In [ ]:
```
## Load data from https://fred.stlouisfed.org/series/UNRATE
unemployment_rate = pd.read_csv('unemployment_rate.csv')

## Convert DATE column to pandas datetime object
unemployment_rate['DATE'] = pd.to_datetime(unemployment_rate['DATE'])

## Round UNRATE column to three decimal places
unemployment_rate = unemployment_rate.round(3)

## Filter data to years between 2000 and 2020 (inclusive)
unemployment_rate = unemployment_rate[(unemployment_rate['DATE'].dt.year >=
                                       (unemployment_rate['DATE'].dt.year <=

## Reset index so first date starts at index 0
unemployment_rate.reset_index(drop = True, inplace = True)

## Rename UNRATE column to unemployment rate
unemployment_rate.columns = ['DATE', 'Unemployment Rate (%)']

cleaned_unemployment_rate = unemployment_rate.set_index('DATE')
```

```
cleaned_unemployment_rate
```

Out[ ]:

DATE	Unemployment Rate (%)
2000-01-01	4.033
2000-04-01	3.933
2000-07-01	4.000
2000-10-01	3.900
2001-01-01	4.233
...	...
2019-10-01	3.600
2020-01-01	3.800
2020-04-01	12.967
2020-07-01	8.833
2020-10-01	6.767

84 rows × 1 columns

Corporate Profits After Tax in Billions

In [ ]:

```
## Load File
corp_profit_after_tax = pd.read_csv('corporate_profits_after_tax.csv')

## Data Cleaning
corp_profit_after_tax['DATE'] = pd.to_datetime(corp_profit_after_tax['DATE']

## Filtering
cleaned_corp_profits = corp_profit_after_tax[(corp_profit_after_tax['DATE'].

## Formatting
corp_profits_after_tax = cleaned_corp_profits.set_index('DATE')
corp_profits_after_tax.rename(columns = {'CP':'Corporate Profits After Tax (

corp_profits_after_tax = corp_profits_after_tax.drop(corp_profits_after_tax.
corp_profits_after_tax
```

Out[ ]:

DATE	Corporate Profits After Tax (USD in Billions)
1999-10-01	590.992
2000-01-01	565.043
2000-04-01	559.322
2000-07-01	554.170
2000-10-01	533.362
...	...
2019-10-01	2120.145
2020-01-01	2009.635
2020-04-01	1837.522
2020-07-01	2534.949
2020-10-01	2367.466

85 rows × 1 columns

Economic Policy Uncertainty Index

In [ ]:
```
# data downloaded from: https://www.policyuncertainty.com/us_monthly.html
df_policy_uncert = pd.read_csv('US_Policy_Uncertainty_Data.csv')

# Function to determine the start date of the quarter
def get_quarter_start(year, month):
    if pd.isna(year) or pd.isna(month):
        return None
    year = int(year)
    if month <= 3:
        return f"{year}-01-01"
    elif month <= 6:
        return f"{year}-04-01"
    elif month <= 9:
        return f"{year}-07-01"
    else:
        return f"{year}-10-01"

# Apply the function to each row
df_policy_uncert['DATE'] = df_policy_uncert.apply(lambda x:
                           get_quarter_start(x['Year'], x['Month']), axis=1
```

```
# Convert the QUARTER column to datetime, handling None values
df_policy_uncert['DATE'] = pd.to_datetime(df_policy_uncert['DATE'], errors='

# ensure data is sorted by date
df_policy_uncert = df_policy_uncert.sort_values(by='DATE')

# Filter data to only include data from 2000 to 2020
df_policy_uncert = df_policy_uncert[(df_policy_uncert['DATE'].dt.year >= 200
                                     & (df_policy_uncert['DATE'].dt.year <=
df_policy_uncert.reset_index(drop=True, inplace=True)

# Group by the DATE column and calculate the mean of the Economic Policy Unc
df_policy_uncert = df_policy_uncert.groupby('DATE')['Three_Component_Index']
df_policy_uncert = df_policy_uncert.reset_index()

# Rename the columns for clarity (EPU = Economic Policy Uncertainty)
df_policy_uncert.columns = ['DATE', 'Avg_EPU_Index']

# Formatting
cleaned_df_policy_uncert = df_policy_uncert.set_index('DATE')

cleaned_df_policy_uncert
```

Out[ ]:

DATE	Avg_EPU_Index
2000-01-01	81.666667
2000-04-01	89.000000
2000-07-01	73.666667
2000-10-01	108.333333
2001-01-01	112.666667
...	...
2019-10-01	131.000000
2020-01-01	191.666667
2020-04-01	283.333333
2020-07-01	249.333333
2020-10-01	247.333333

84 rows × 1 columns

Quarterly Change in S&P 500

In [ ]:
```
df_stockprices = pd.read_csv('S&P500.csv')
df_stockprices = df_stockprices[["Date", "Close*"]]
df_stockprices.columns = ["Date", "Close"]

# convert string to date time format, then filter for years before 2020 afte
df_stockprices['Date'] = pd.to_datetime(df_stockprices['Date'])
df_stockprices['Close'] = pd.to_numeric(df_stockprices['Close'].str.replace(
df_stockprices = df_stockprices[(df_stockprices['Date'].dt.year >= 1999) & (

# a function to clean up by quarters
def filterbyquarters(df_stockprices_Q, start_month, end_month):

    df_stockprices_QX = df_stockprices_Q[df_stockprices_Q['Date'].dt.month.b
    df_stockprices_QX = df_stockprices_QX.groupby(df_stockprices_QX['Date'].
    df_stockprices_QX.reset_index(drop=True, inplace=True)
    df_stockprices_QX['Date'] = df_stockprices_QX['Date'].dt.date # change t
    df_stockprices_QX['Date'] = df_stockprices_QX['Date'].apply(lambda x: x.
    df_stockprices_QX['Date'] = df_stockprices_QX['Date'].apply(lambda x: x.
    df_stockprices_QX = df_stockprices_QX.round(2) # round the value off to

    return df_stockprices_QX

# perform cleaning up using above function on all four quarters
df_stockprices_Q1 = filterbyquarters(df_stockprices, 1, 3)
df_stockprices_Q2 = filterbyquarters(df_stockprices, 4, 6)
df_stockprices_Q3 = filterbyquarters(df_stockprices, 7, 9)
df_stockprices_Q4 = filterbyquarters(df_stockprices, 10, 12)

# merge them back and sort by date
merged_df = pd.concat([df_stockprices_Q1, df_stockprices_Q2, df_stockprices_
merged_df.sort_values(by='Date', inplace=True)
merged_df.reset_index(drop=True, inplace=True)

# cleaning up
merged_df['Close_Percentage_Difference'] = merged_df['Close'].pct_change() *
merged_df = merged_df.drop(index=[0, 1]) # drop the first two unnecessary ro
merged_df = merged_df[['Date', 'Close_Percentage_Difference']]
merged_df = merged_df.rename(columns={'Date': 'DATE', 'Close_Percentage_Diff
merged_df['DATE'] = pd.to_datetime(merged_df['DATE'])
merged_df = merged_df.round(2) # round the value off to 2 decimal places
merged_df.set_index('DATE', inplace=True)
cleaned_df_stockchange = merged_df

cleaned_df_stockchange
```

Out[ ]:

DATE	Quarterly Change in S&P500 (%)
2000-01-01	-0.65
2000-04-01	1.39
2000-07-01	1.50
2000-10-01	-9.83
2001-01-01	-4.72
...	...
2019-10-01	6.90
2020-01-01	-8.37
2020-04-01	7.52
2020-07-01	7.62
2020-10-01	9.21

84 rows × 1 columns

Data Cleaning

For all the datasets besides, Economic Policy Uncertainty Index and Quarterly Change in S&P500, data cleaning was necessary, but was very minute. The data cleaning excluding these two revolved around standardizing the date column to be all the same to make a merge easier, and changing the name of the column of interest to make it clear what the value represented. It also included truncating the data to fit the year of interest we have indicated in our research question. For Economic Policy Uncertainty Index, more work was correctly formatting the quarters for a proper merge, and also taking the monthly data provided by the dataset and combining them to fit the quarterly format. We used mean as our combined format as we wanted to make sure we captured all the monthly data to make our analysis slightly more precise. This variable would also include the previous cleaning techniques. For Quarterly Change in S&P500, we had to again change the DATE column to fit the format of the other datasets, while differencing the quarter of interest and the past interest to get the quarterly change. Change was decided as to prevent autocorrelation, which this concept will be discussed later. Again, the previous cleaning techniques were used as well. However, given the data we used, imputation and missing values did not have to be handled as all the data we needed was given through

the datasets.

Results

Exploratory Data Analysis

Data merge

In [ ]:
```
# Concatenate dataframes into one single dataframe
finance_df = pd.concat([cleaned_df_fedfunds, cleaned_unemployment_rate, corp
                        cleaned_df_policy_uncert, cleaned_df_stockchange, cl

finance_df = finance_df.iloc[1: , :]

# Variables for convenience
fed_funds_rate = finance_df.columns[finance_df.columns.get_loc("Federal Func
unemployment_rate = finance_df.columns[finance_df.columns.get_loc("Unemploym
corp_profits = finance_df.columns[finance_df.columns.get_loc("Corporate Prof
avg_EPU_index = finance_df.columns[finance_df.columns.get_loc("Avg_EPU_Index
delta_sp = finance_df.columns[finance_df.columns.get_loc("Quarterly Change i

# Variable for Line Charts
reset_index_df = finance_df.reset_index()

finance_df = finance_df.merge(corp_profit_after_tax, left_on='DATE', right_c
finance_df = finance_df.drop(columns=['CP'])

finance_df
```

Out[ ]:

	DATE	Federal Funds Effective Rate (Interest Rate) (%)	Unemployment Rate (%)	Corporate Profits After Tax (USD in Billions)	Avg_EPU_Index	Quarterly Change in S&P500 (%)	M2V
0	2000-01-01	5.68	4.033	565.043	81.666667	-0.65	2.135
1	2000-04-01	6.27	3.933	559.322	89.000000	1.39	2.151
2	2000-07-01	6.52	4.000	554.170	73.666667	1.50	2.141
3	2000-10-01	6.47	3.900	533.362	108.333333	-9.83	2.133
4	2001-01-01	5.59	4.233	562.885	112.666667	-4.72	2.085
...	...	...	...	...	...	...	...
79	2019-10-01	1.64	3.600	2120.145	131.000000	6.90	1.438
80	2020-01-01	1.26	3.800	2009.635	191.666667	-8.37	1.391
81	2020-04-01	0.06	12.967	1837.522	283.333333	7.52	1.128
82	2020-07-01	0.09	8.833	2534.949	249.333333	7.62	1.176
83	2020-10-01	0.09	6.767	2367.466	247.333333	9.21	1.163

84 rows × 7 columns

Section 1 of EDA - Descriptive Analysis

Boxplots are used to display the distribution of numerical data and skewness by displaying the data quartiles (or percentiles) and averages 1. It visually shows the five-number summary of a set of data: including the minimum score, first (lower) quartile, median, third (upper) quartile, and maximum score. Most importantly, it indicates outliers which will be important in our descriptive analysis.

Line charts are used to how a variable changes over time, which is paramount in our analysis since we are looking at data between a certain time period.

In [ ]:
```
finance_description = finance_df.loc[:, finance_df.columns != 'DATE'].descri
finance_description
```

Out[ ]:

	Federal Funds Effective Rate (Interest Rate) (%)	Unemployment Rate (%)	Corporate Profits After Tax (USD in Billions)	Avg_EPU_Index	Quarterly Change in S&P500 (%)	M
count	84.000000	84.000000	84.000000	84.000000	84.000000	84.0000
mean	1.716905	5.986833	1480.976048	121.440476	1.347857	1.7296
std	1.902200	1.972044	525.224623	42.154213	6.465957	0.2537
min	0.060000	3.600000	512.713000	63.000000	-21.030000	1.1280
25%	0.147500	4.550250	1105.866750	91.833333	-1.390000	1.5155
50%	1.085000	5.433000	1565.871000	112.666667	2.595000	1.7270
75%	2.400000	7.008000	1883.646500	140.583333	5.565000	1.9697
max	6.520000	12.967000	2534.949000	283.333333	16.590000	2.1510

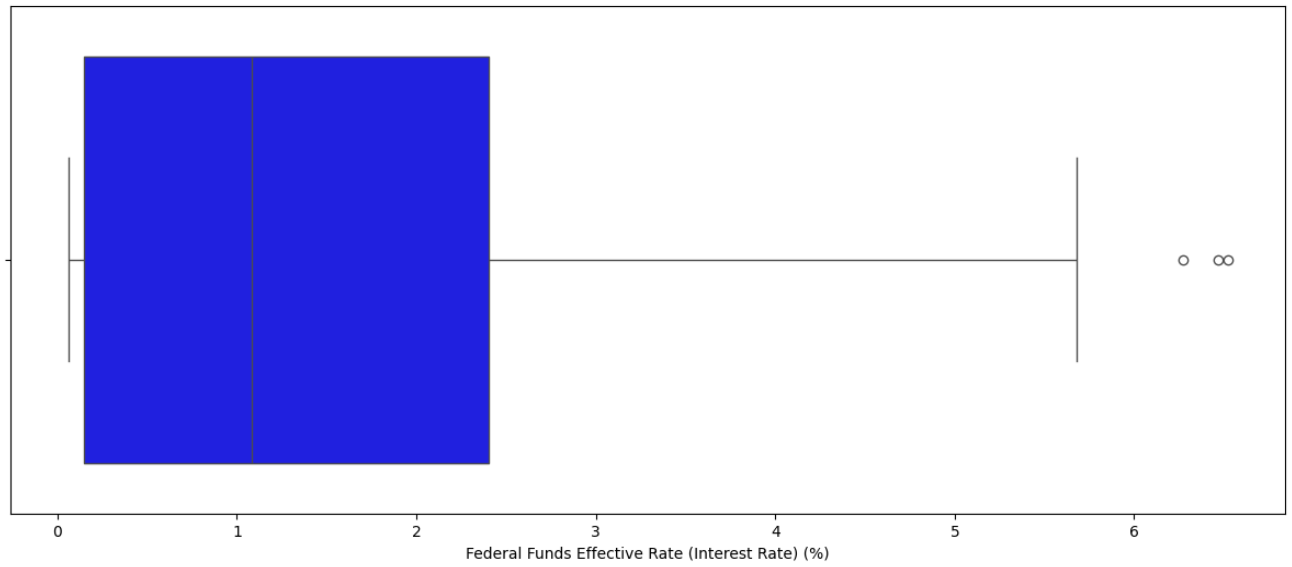
Here we can see some important statistical information about our the data we have chosen. We can see that besides Corporate Profits After Tax, Unemployment Rate, and Average EPU Index, the standard deviation or spread from the mean is relatively low. This tell us two things. First that for most of the variables, their data points are relatively close to the mean and do not have much variance. Second, it shows that these variables may not have outliers, or more conservatively, they have outliers that have little effect on the standard deviation. This can also be reversed, as those with high standard deviations may have many outliers or outliers that have a big effect. Thus we will create boxplots to check the general spread of values and outliers.

Outlier Search

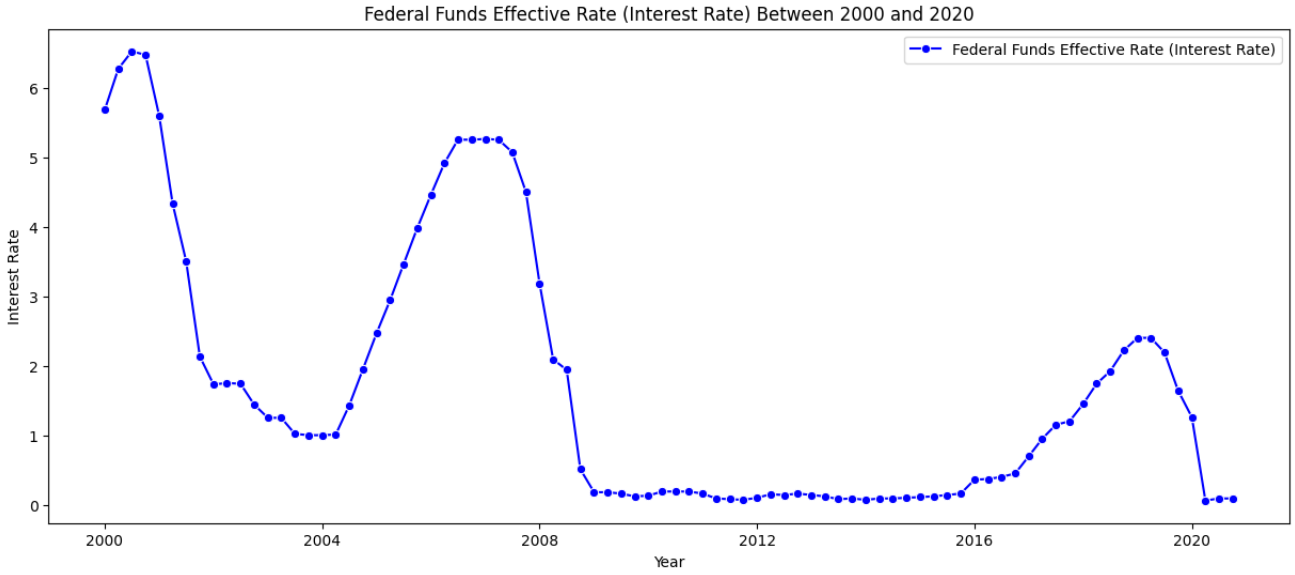
Interest Rate Outliers

In [ ]:
```
plt.figure(figsize=(15, 6))
sns.boxplot(x=finance_df[fed_funds_rate],color='blue')
```

Out[ ]: <Axes: xlabel='Federal Funds Effective Rate (Interest Rate) (%)'>



In [ ]:
```
#Size the figure
plt.figure(figsize=(15, 6))

#Plot the lineplot
Interest_Rate = sns.lineplot(data=reset_index_df, x='DATE', y=fed_funds_rate

#Add other graph elements
plt.title('Federal Funds Effective Rate (Interest Rate) Between 2000 and 202
plt.xlabel('Year')
plt.ylabel('Interest Rate')
plt.legend()
plt.show()
```



In [ ]:
```
# finding the outliers from Federal Funds Effective Rate (Interest)
```

```
data = finance_df[fed_funds_rate]
Q1 = data.quantile(0.25)
Q3 = data.quantile(0.75)
IQR = Q3 - Q1
outlier_threshold = 1.5
outliers = (data < Q1 - outlier_threshold * IQR) | (data > Q3 + outlier_thre
print(data[outliers])
```

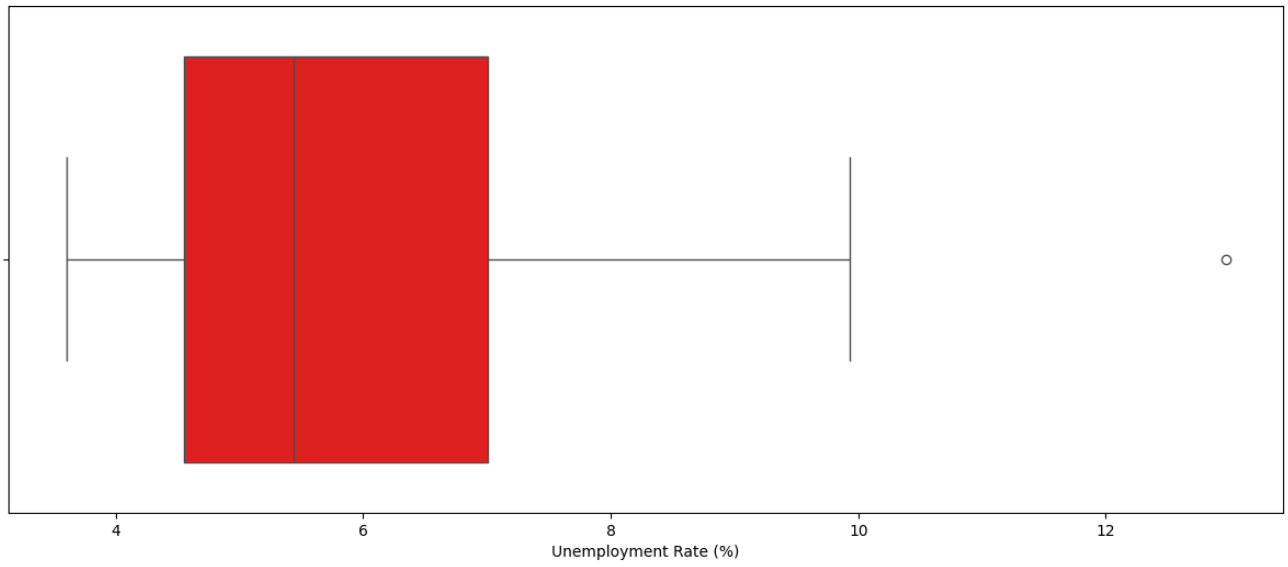
```
1    6.27
2    6.52
3    6.47
Name: Federal Funds Effective Rate (Interest Rate) (%), dtype: float64
```

From both our boxplot and the above code, we can see that there are three main outliers in the year 2000.

Prior to early 2000s, the prevalence of low interest rates facilitated an increase in startup companies. The inevitable burst of the Dot-Com Bubble led to an uncontrollable inflation within the economy. It is likely that in response, the Federal Reserve hiked interest rates in order to counter inflation to maintain affordability for the general public.

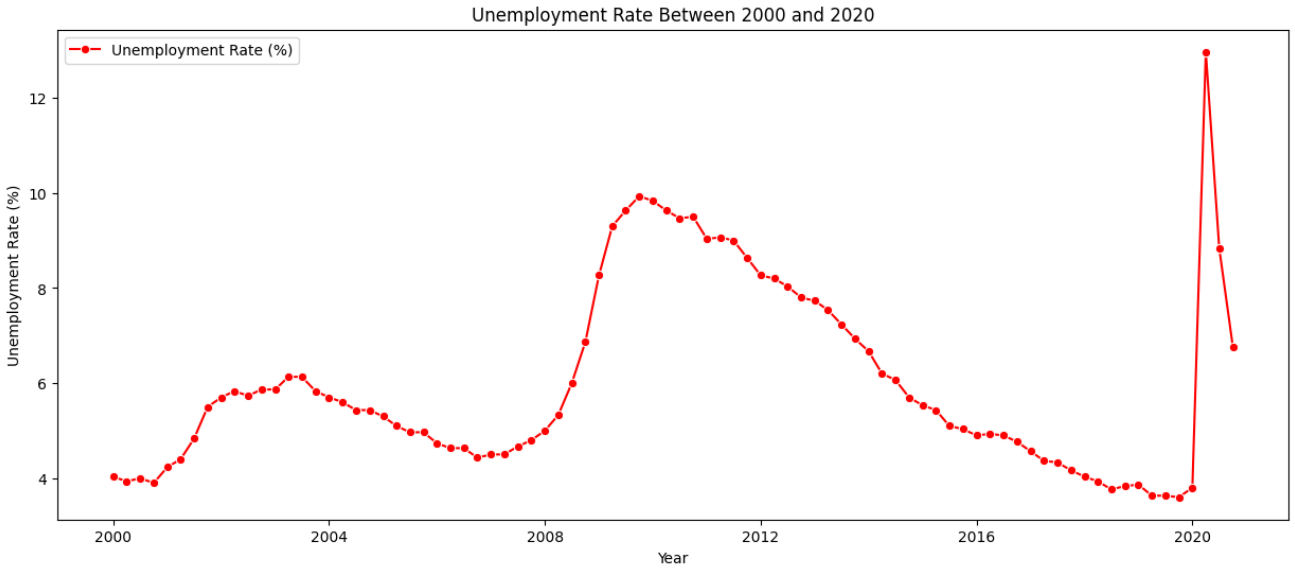
Unemployment Rate Outlier

In [ ]:
```
plt.figure(figsize=(15, 6))
sns.boxplot(x=finance_df[unemployment_rate], color = 'red')
plt.xlabel(unemployment_rate)
plt.show()
```



In [ ]:
```
#Size the figure
plt.figure(figsize=(15, 6))

#Plot the lineplot
```

```
Unemployment_Rate = sns.lineplot(data=reset_index_df, x='DATE', y=unemployme

#Add other graph elements
plt.title('Unemployment Rate Between 2000 and 2020')
plt.xlabel('Year')
plt.ylabel(unemployment_rate)
plt.legend()
plt.show()
```



In [ ]:
```
# finding the outliers from Unemployment Rate
data = finance_df[unemployment_rate]
Q1 = data.quantile(0.25)
Q3 = data.quantile(0.75)
IQR = Q3 - Q1
outlier_threshold = 1.5
outliers = (data < Q1 - outlier_threshold * IQR) | (data > Q3 + outlier_thre
print(data[outliers])
```

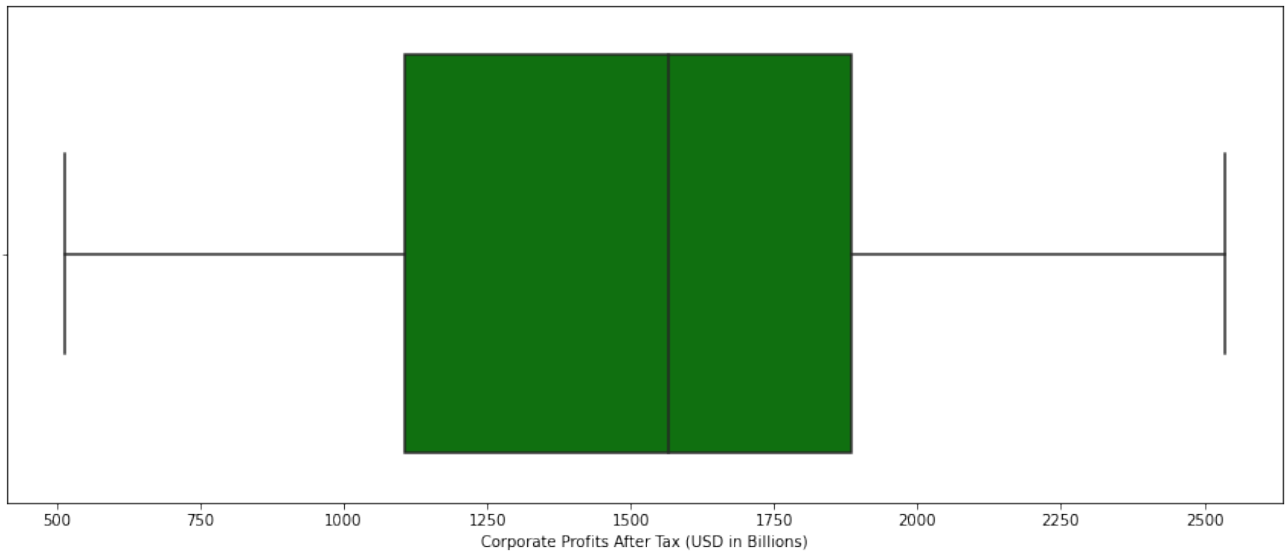
```
81    12.967
Name: Unemployment Rate (%), dtype: float64
```

Due to the global financial crisis in 2008, the unemployment rate increased heavily because there were lower demands for goods and because labor is a key economic input, companies had to lay off workers. In 2020, the unemployment rate also peaked due to the COVID-19 outbreak. Because companies and businesses had to shutdown in order to contain the outbreak, millions of people lost their jobs.

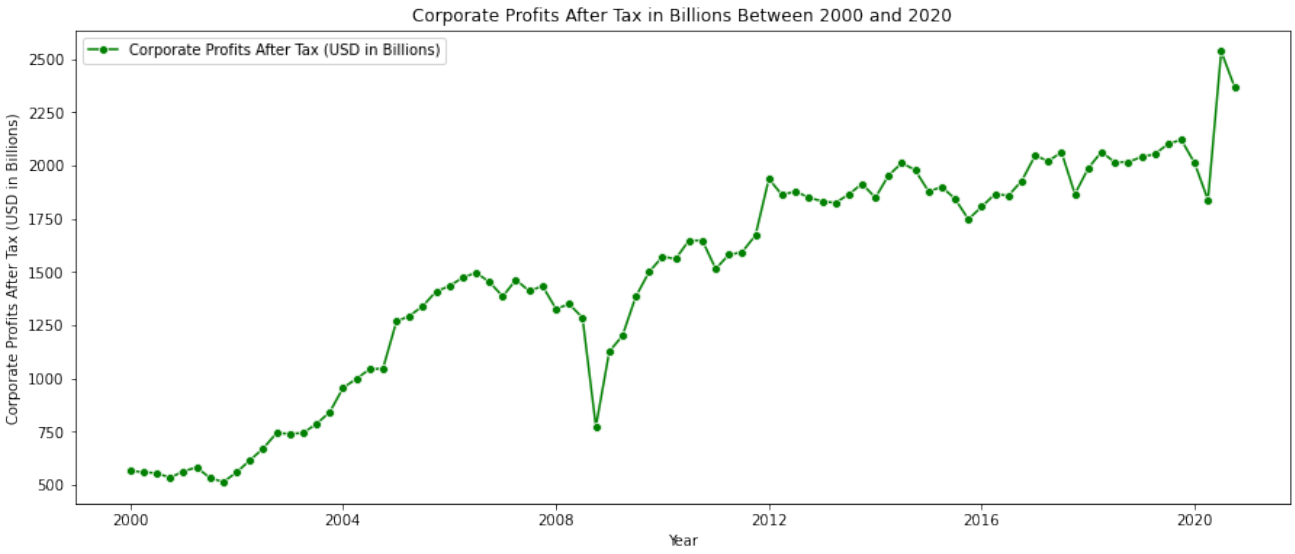
Corporate Profits Outliers

In [ ]:
```
plt.figure(figsize=(15, 6))
sns.boxplot(x=finance_df[corp_profits], color = 'green')
plt.xlabel(corp_profits)
```

```
plt.show()
```



In [ ]:
```
#Size the figure
plt.figure(figsize=(15, 6))

#Plot the lineplot
Corporate_Profits = sns.lineplot(data=reset_index_df, x='DATE', y=corp_profi

#Add other graph elements
plt.title('Corporate Profits After Tax in Billions Between 2000 and 2020')
plt.xlabel('Year')
plt.ylabel(corp_profits)
plt.legend()
plt.show()
```

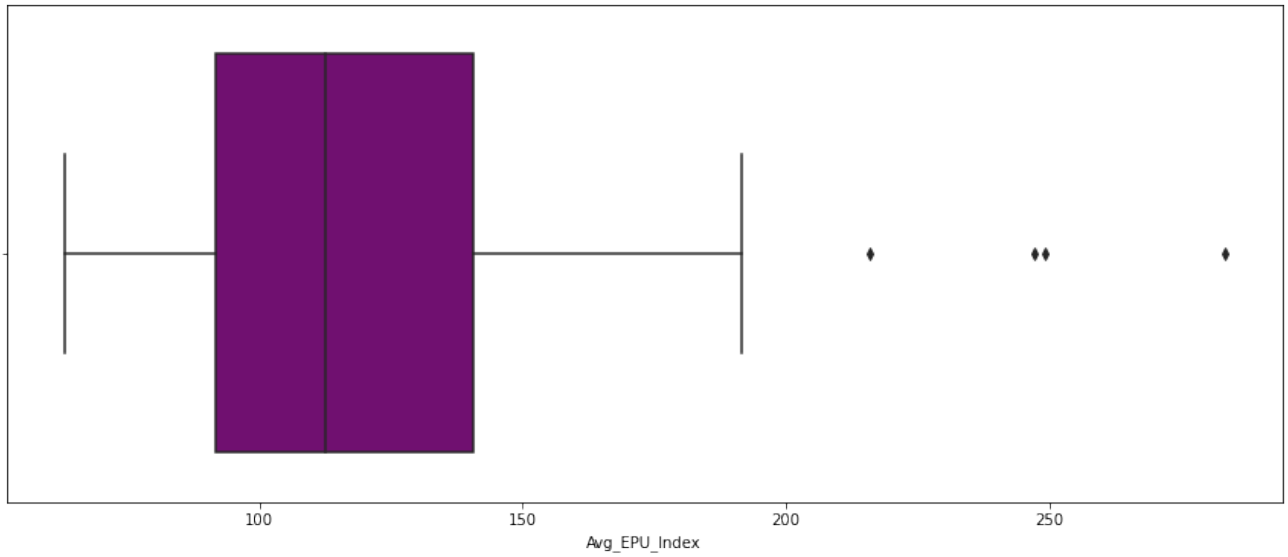


The 2008 global financial crisis also disrupted the regular pattern we would expect to see. Due to the mortgage crisis, financial institution losses, reduced consumer spending,

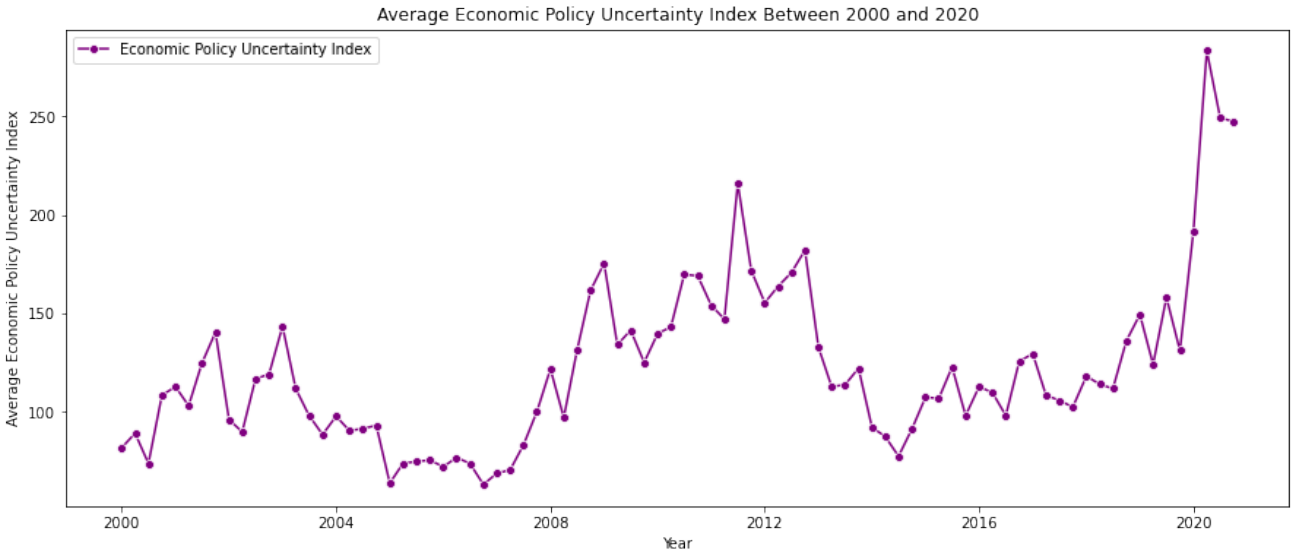
and overall global economic slowdown, there was a significant decline in corporate profits. However, there doesn't appear to be any significant outliers according to our boxplot.

Economic Policy Uncertainty Index Outliers

In [ ]:
```
plt.figure(figsize=(15, 6))
sns.boxplot(x=finance_df[avg_EPU_index], color = 'purple')
plt.xlabel("Avg_EPU_Index")
plt.show()
```



In [ ]:
```
# Size the figure
plt.figure(figsize=(15, 6))

# Plot the lineplot
Economic_Policy_Uncertainty = sns.lineplot(data=reset_index_df, x='DATE', y=

# Add other graph elements
plt.title('Average Economic Policy Uncertainty Index Between 2000 and 2020')
plt.xlabel('Year')
plt.ylabel('Average Economic Policy Uncertainty Index')
plt.legend()
plt.show()
```



In [ ]:
```
# finding the outliers from Average Economic Policy Uncertainty (EPU) Index
Q1 = finance_df[avg_EPU_index].quantile(0.25)
Q3 = finance_df[avg_EPU_index].quantile(0.75)
IQR = Q3 - Q1
outlier_threshold = 1.5
outliers = (finance_df[avg_EPU_index] < Q1 - outlier_threshold * IQR) | (fir
print(finance_df[avg_EPU_index][outliers])
```

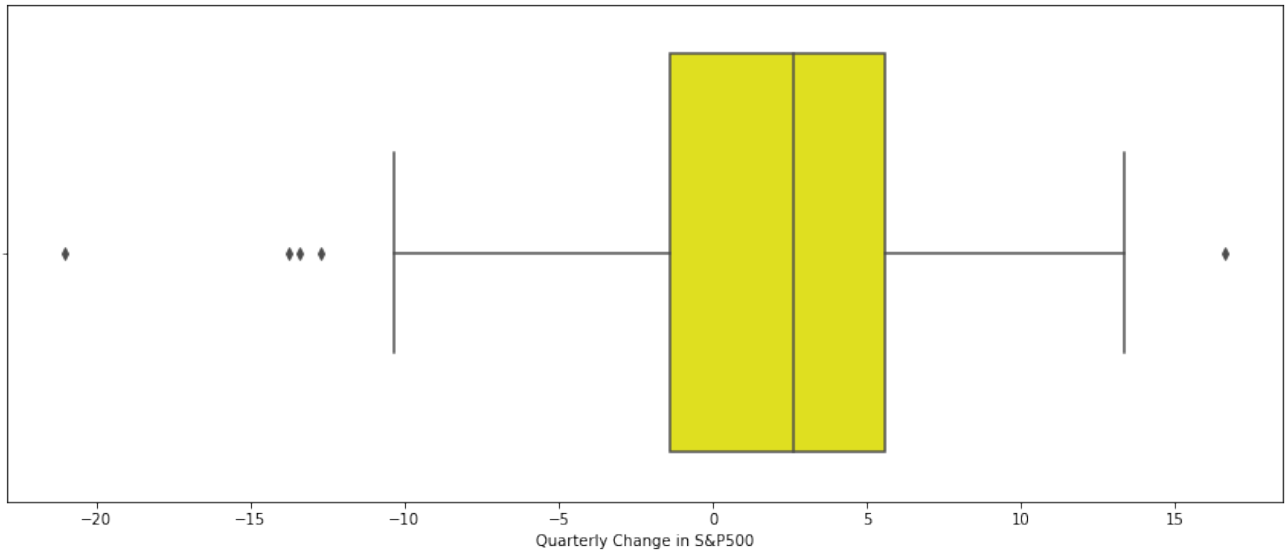
```
46    216.000000
81    283.333333
82    249.333333
83    247.333333
Name: Avg_EPU_Index, dtype: float64
```

We can see that we have a single outlier in 2011 and three outliers in the year 2020.

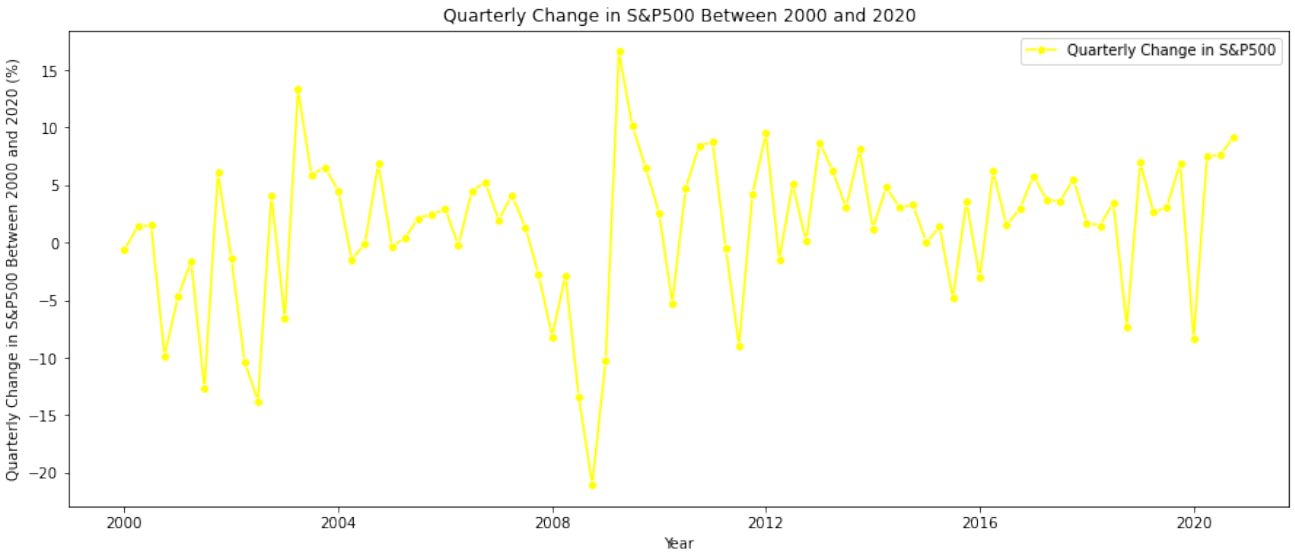
The higher than usual average EPU in 2011 can be attributed to the simultaneous crises of the U.S. debt ceiling and the European Union debt crisis. These issues fostered stark contrasting views within governments. Similarly, in 2020, the onset of a global pandemic, COVID-19, left governments with uncertainty and challenged to determine their next course of action. Both these events resulted in widespread doubts about government and economic stability and naturally led to a higher average EPU.

Quarterly Change in S&P500

In [ ]:
```
plt.figure(figsize=(15, 6))
sns.boxplot(x=finance_df[delta_sp], color = 'yellow')
plt.xlabel("Quarterly Change in S&P500")
plt.show()
```



In [ ]:
```
# Size the figure
plt.figure(figsize=(15, 6))

# Plot the lineplot
Economic_Policy_Uncertainty = sns.lineplot(data=reset_index_df, x='DATE', y=

# Add other graph elements
plt.title('Quarterly Change in S&P500 Between 2000 and 2020')
plt.xlabel('Year')
plt.ylabel('Quarterly Change in S&P500 Between 2000 and 2020 (%)')
plt.legend()
plt.show()
```



In [ ]:
```
# finding the outliers from Quarterly Change in S&P500
Q1 =finance_df[delta_sp].quantile(0.25)
Q3 = finance_df[delta_sp].quantile(0.75)
IQR = Q3 - Q1
```

```
outlier_threshold = 1.5
outliers = (finance_df[delta_sp] < Q1 - outlier_threshold * IQR) | (finance_
print(finance_df.loc[outliers, [delta_sp]])
```

```
    Quarterly Change in S&P500 (%)
6                           -12.72
10                          -13.74
34                          -13.42
35                          -21.03
37                           16.59
```

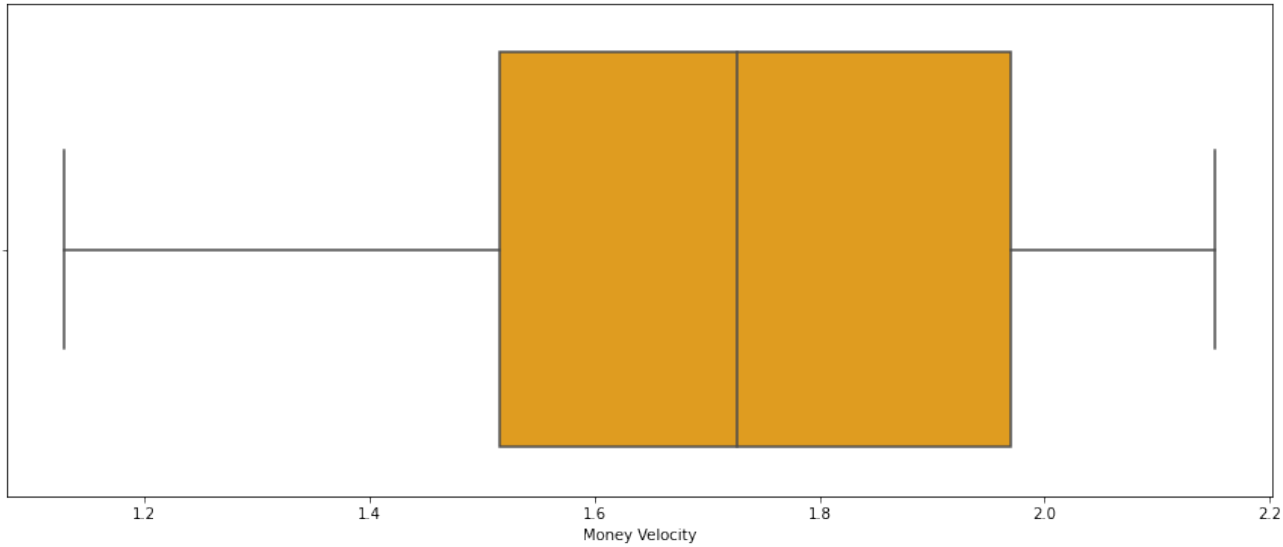
There are two outliers in the early 2000s and three outliers in the late 2000s.

Expanding on the above section (Federal Funds Effective Rate (Interest)), the Dot-Com Bubble in the early 2000s precipitated a sharp fall in the S&P500 prices because of the gargantuan increase in startup companies. In 2008, there was the most severe global financial crises since the great depression triggered by the collapse of the U.S. housing market. These factors led to economic uncertainty within the markets, explaining the sharp decline in the S&P500.

Naturally, the outlier in 2009 was the effect of a rebound in the economy caused previously by the 2008 financial crisis.

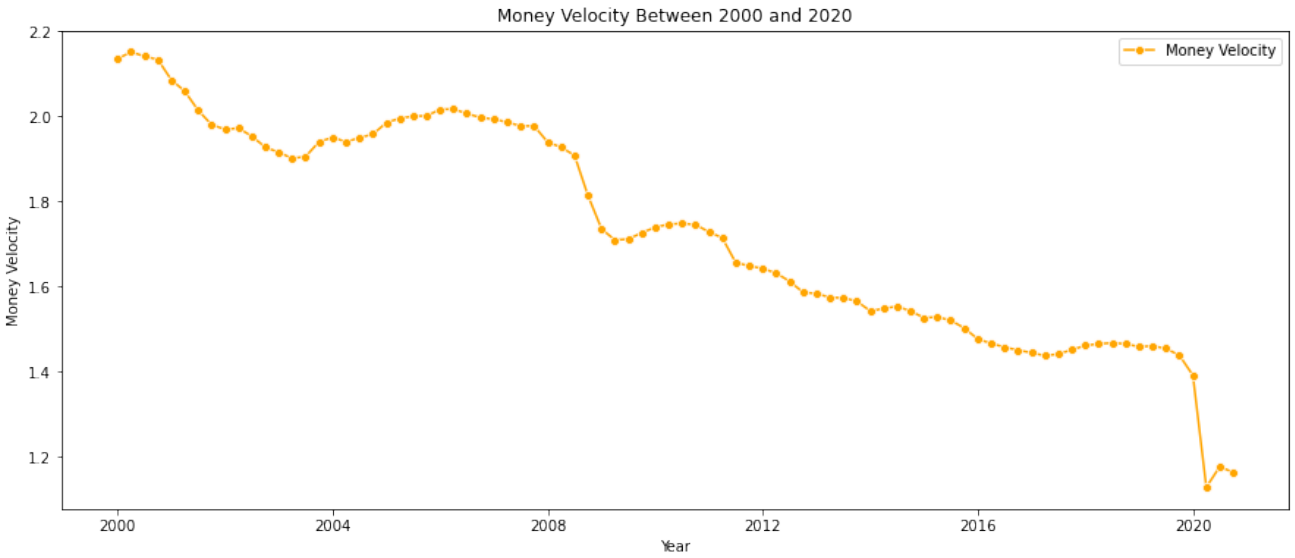
Money Velocity Outliers

In [ ]:
```
plt.figure(figsize=(15, 6))
sns.boxplot(x=finance_df['M2V'], color = 'orange')
plt.xlabel("Money Velocity")
plt.show()
```



In [ ]:
```
#Size the figure
```

```
plt.figure(figsize=(15, 6))

#Plot the lineplot
Money_Velocity = sns.lineplot(data=reset_index_df, x='DATE', y='M2V', marker

#Add other graph elements
plt.title('Money Velocity Between 2000 and 2020')
plt.xlabel('Year')
plt.ylabel('Money Velocity')
plt.legend()
plt.show()
```



Like we've seen throughout the other graphs, the two events that significantly impacted the regular patterns of the graph were the global financial crisis in 2008 and the COVID-19 outbreak in 2020. This is also reflected in this graph, as the decline in money velocity is more significant at 2008 and 2020.

Section 2 of EDA - Pairplot, Trendlines, and Heatmaps

Pairplots are used to visualize the relationship between multiple variables in a dataset. It is a matrix of scatterplots, where each variable in the dataset is paired with every other variable. We use these to determine linear correlations between variables.

Since we will be running OLS regression to explore correlations between independent variables and the dependent variable Money Velocity. We need to first test whether the assumptions are satisfied or not. We use the pairplot to comprehensively visualize the relationships between variables.

1. **Linearity.** the relationship between the dependent and independent variables is

linear. This ensures that changes in the independent variables have a consistent effect on the dependent variable, encompassing linearity in both coefficients and error terms.

2. **No multicollinearity.** Independent variables are not perfectly correlated. Perfect collinearity hinders unique estimation of coefficients, making the model unstable.

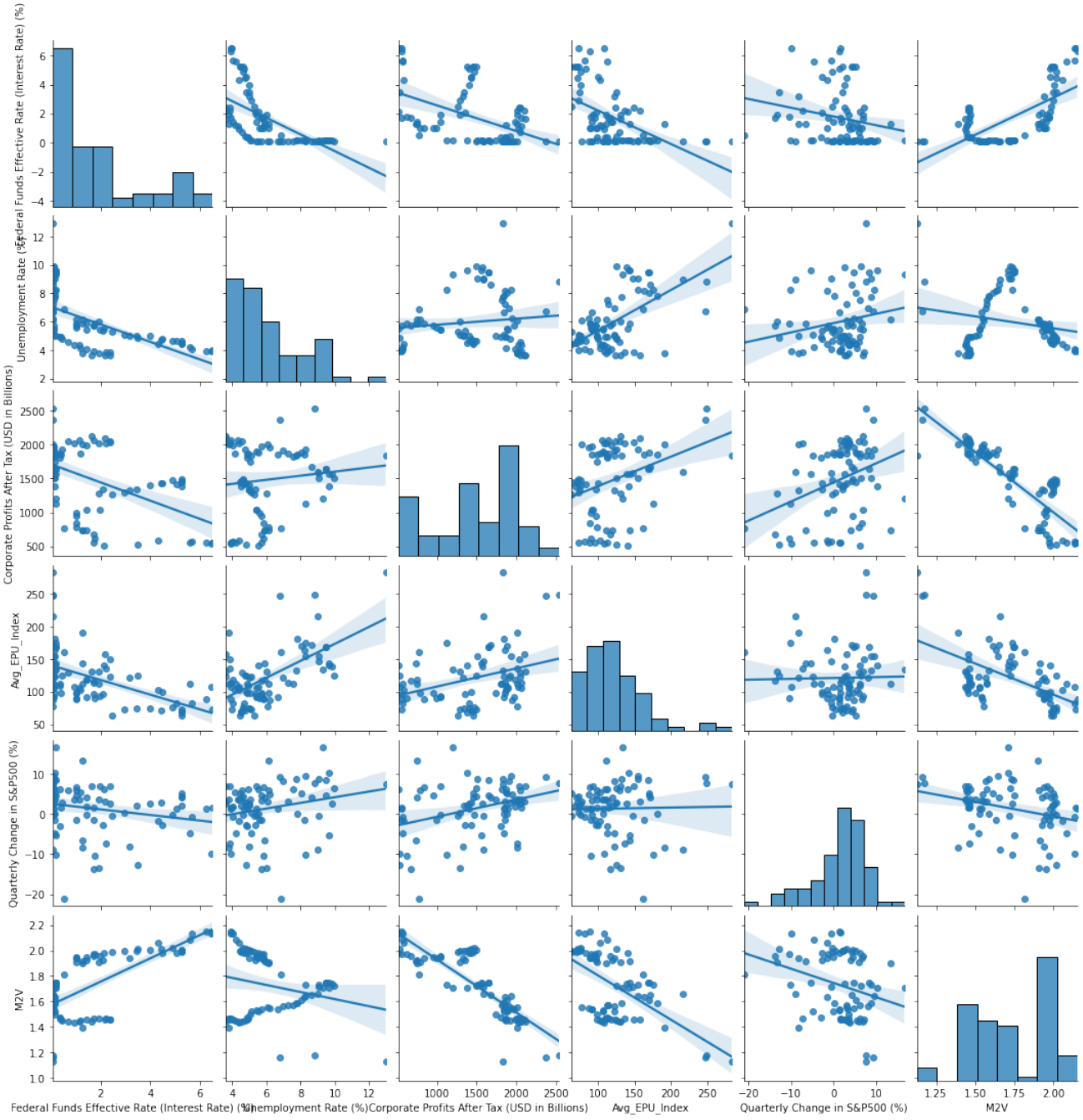
3. **No auto-correlation.** Errors are not systematically correlated over time. It implies that the residuals from the model are independent, and the occurrence of an error at one point in time does not predict errors at subsequent points, ensuring unbiased parameter estimates.

4. **Homoscedasticity.** The variance of the error term is constant across all levels of the independent variables, preventing heteroskedasticity that can affect the efficiency of estimates.

In this section 2 of EDA, we will be checking assumption 1 (linearity), assumption 2 (no multicollinearity), and assumption 4 (homoscedasticity).

In [ ]:
```
plt.figure(figsize=(15, 6))
sns.pairplot(finance_df,kind="reg")
plt.show()
```

```
<Figure size 1080x432 with 0 Axes>
```



Assumption 1 (Linearity): From the scatterplots of independent variables on the dependent variable, we identify **a strong linear negative relationship between Corporate Profits After Tax and M2V.** A rise in Corporate Profits After Tax is strongly correlated with a decrease in M2V money velocity. **A moderate linear negative relationship between Avg_EPU_Index and M2V.** A rise in EPU index is moderately associated with a decrease in M2V money velocity. **A weak linear negative relatinship between Change in S&P and M2V.** A rise in Change in S&P is weakly associated with a decrease in M2V money velocity. Federal Funds Effective Rate (Interest Rate) & M2V and Unemployment Rate & M2V, however, do not display linear relationships.

Assumption 2 (No Multicollinearity): From the scatterplots of independent variables on independent variables and also intuitively we identify that **there exist no perfect multicollinearity between independent variables. None of the independent variables is a linear combinition of the other.**

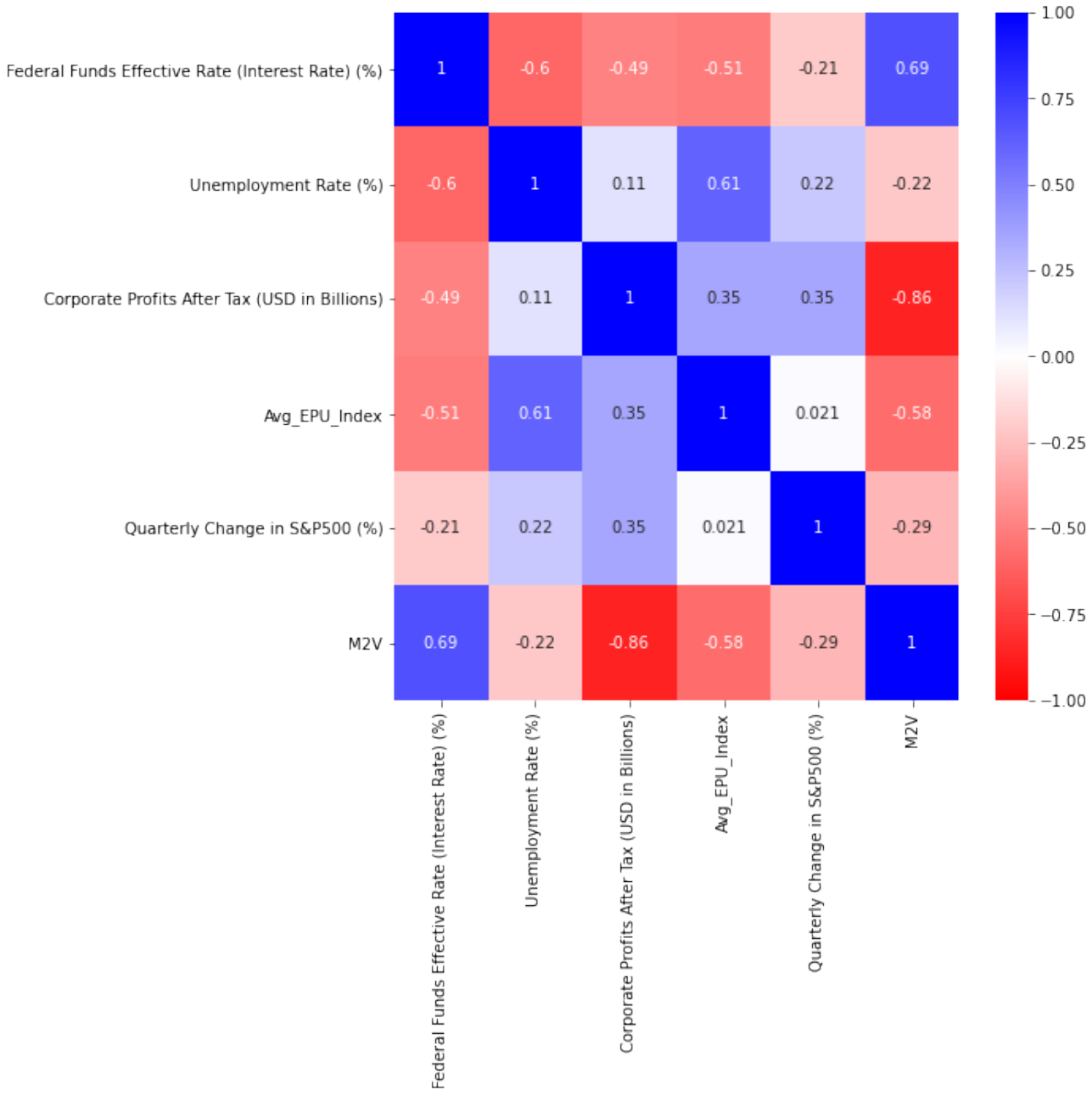
Assumption 4 (Homoscedasticity): From the scatterplots of independent variables on the dependent variable, we identify that the residuals vary across the independent variable, **we cannot assume homoscedasticity of the standard error.** Therefore, we will use the heteroscedastic-robust standard error for the OLS Regression.

Heatmap

Furthermore, in the heatmap below we show the pearson correlation coefficient between all of our variables. We can see that none of our independent variables appear to be strongly correlated, except for unemployment rate and federal funds rate.

Because of the non-linear relationship between unemployment rate and money velocity, and the strong correlation between unemployment rate and federal funds rate, we will remove unemployment rate from our analysis moving forward. Using unemployment rate as an independent variable in our linear regression rate will give us results that are not a good fit.

In [ ]:
```
plt.figure(figsize=(8, 8))
cmap = LinearSegmentedColormap.from_list('custom_red_white_blue', ['red', 'w
heatmap=sns.heatmap(finance_df.drop(columns=['DATE']).corr(),vmin=-1,vmax=1,
```



Adding New Variables

Because we are removing a variable from our analysis, we are going to add in two additional variables to take its place: **United States Gini Coefficient** and **majority party in congress** data.

Import Gini Coefficient Data

Put simply, the gini coefficient is a measure of wealth inequality from 0 to 1. 0 represents perfect wealth equality, and 1 represents perfect inequality. We are using this variable

because it may have a positive relationship with money velocity. A country with high wealth inequality might have lower money velocity because the wealth is being hoarded by a few individuals who are not spending their money, whereas a country with low wealth inequality might have a higher money velocity because the wealth is distributed across more individuals.

Dataset Information:

- **Definition of Gini Coefficient**: The Gini coefficient is a measure of income distribution within a population, commonly used in economics to gauge wealth inequality. It is represented as a number between 0 and 1, where 0 represents perfect wealth equality (everyone has the same income) and 1 signifies perfect inequality (all income is held by a single individual).
- **Dataset Name**: United States Gini Index
- **Link to the dataset**: https://data.worldbank.org/indicator/SI.POV.GINI?locations=US (*The World Bank*).
- **Number of observations**: 58
- **Number of variables**: 2 (Gini Coefficient and year)

In [ ]:
```
# Read in gini coefficient data
gini_coef = pd.read_csv('gini_coefficient.csv')

# Filter to only include United States data
us_gini_coef = gini_coef.iloc[251:252].drop(columns=['Country Code', 'Indica

# Drop null columns
unnamed_cols = [col for col in us_gini_coef.columns if 'Unnamed' in col]
unnamed_null_cols = [col for col in unnamed_cols if us_gini_coef[col].isna()
us_gini_coef = us_gini_coef.drop(columns=unnamed_null_cols)

# Transpose the dataframe
us_gini_coef = us_gini_coef.T
us_gini_coef.columns = us_gini_coef.iloc[0]
us_gini_coef = us_gini_coef[1:]

# Rename columns
us_gini_coef.columns = ['US Gini Coefficient']

# Filter data to only include data from 2000-2020
us_gini_coef.index = us_gini_coef.index.astype(int)
us_gini_coef = us_gini_coef[(us_gini_coef.index >= 2000) & (us_gini_coef.inc

# Convert yearly data to quarterly
quarterly_data = []
for year in us_gini_coef.index:
```

```
    for quarter_start in ['01-01', '04-01', '07-01', '10-01']:
        date = f"{year}-{quarter_start}"
        gini_value = us_gini_coef.loc[year, 'US Gini Coefficient']
        quarterly_data.append({'DATE': date, 'US Gini Coefficient': gini_val
us_gini_coef = pd.DataFrame(quarterly_data)

us_gini_coef
```

Out[ ]:

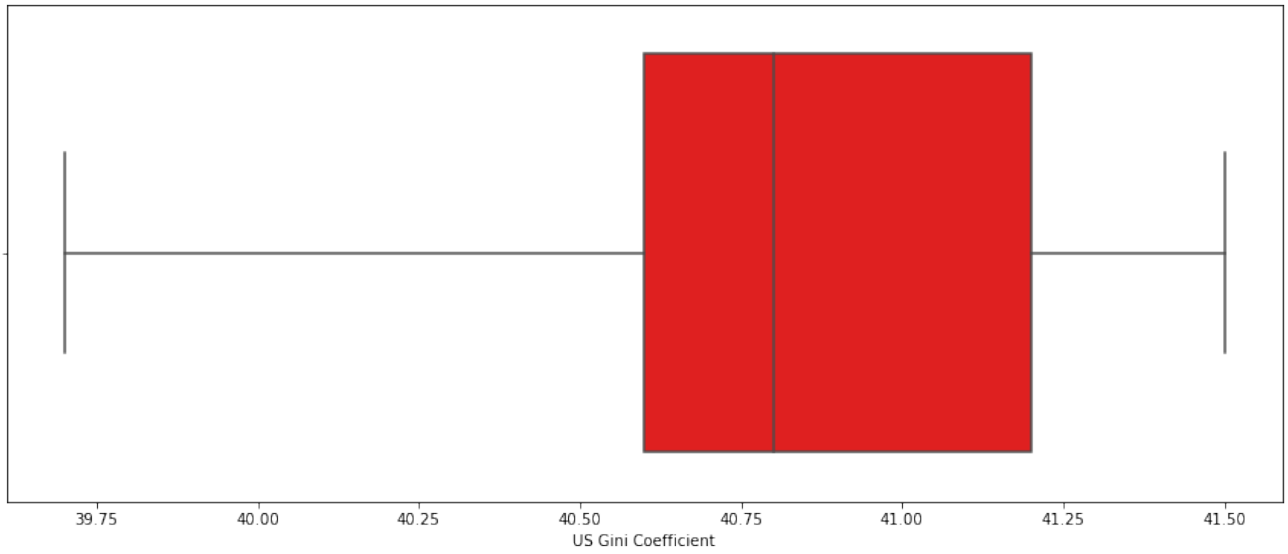
	DATE	US Gini Coefficient
0	2000-01-01	40.1
1	2000-04-01	40.1
2	2000-07-01	40.1
3	2000-10-01	40.1
4	2001-01-01	40.6
...	...	...
79	2019-10-01	41.5
80	2020-01-01	39.7
81	2020-04-01	39.7
82	2020-07-01	39.7
83	2020-10-01	39.7

84 rows × 2 columns

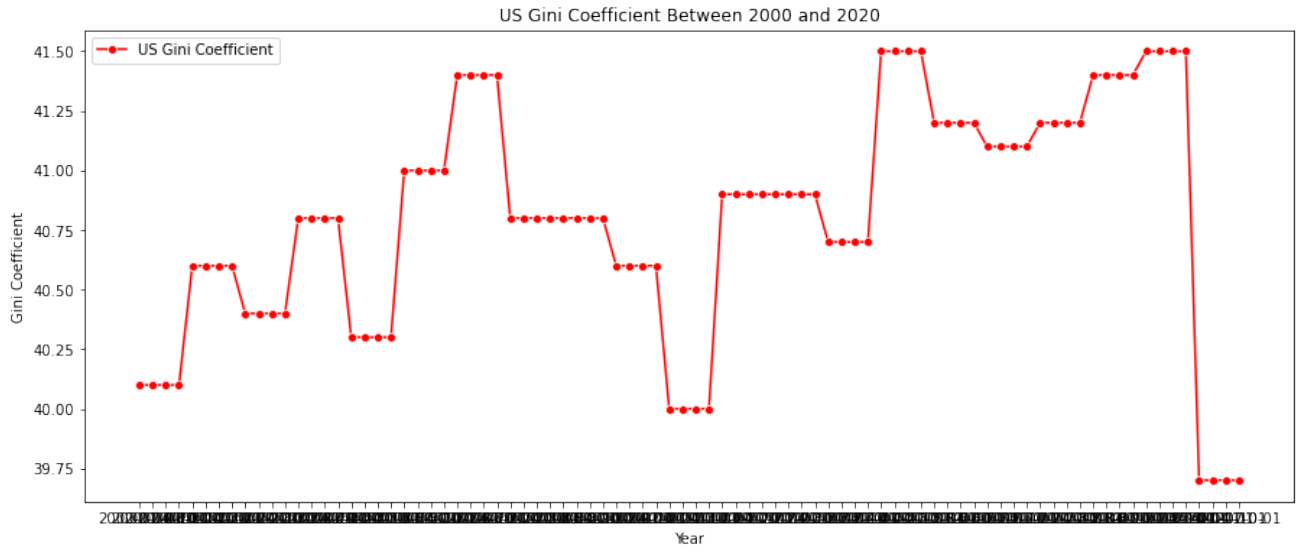
US Gini Outliers

In [ ]:
```
plt.figure(figsize=(15, 6))
sns.boxplot(x=us_gini_coef['US Gini Coefficient'], color='red')
```

Out[ ]:
```
<AxesSubplot:xlabel='US Gini Coefficient'>
```



In [ ]:
```
#Size the figure
plt.figure(figsize=(15, 6))

#Plot the lineplot
Gini_Coef = sns.lineplot(data=us_gini_coef, x='DATE', y='US Gini Coefficient

#Add other graph elements
plt.title('US Gini Coefficient Between 2000 and 2020')
plt.xlabel('Year')
plt.ylabel('Gini Coefficient')
plt.legend()
plt.show()
```



Import Majority Party in Congress Data

Data pertaining to the majority political party in the house of represetitives for the U.S.

Congress. The house was chosen as they are responsible for making and passing federal laws. With different parties having different views of the economy, a difference in party could lead to a difference in money velocity.

Dataset Information:

- **Definition**: Majority Party in the House of Representitives for the U.S. Congress; 1 is republican and 0 is democrat.
- **Dataset Name**: 1-20 Political Parties of Senators and Representatives, 34th – 117th Congresses, 1855 – 2021
- **Link to the dataset**: https://www.brookings.edu/articles/vital-statistics-on-congress/
- **Number of observations**: 840
- **Number of variables**: 5

In [ ]:
```
#Import data
df_party = pd.read_csv('1-20.csv')

# Format Data
dem_rep = df_party[(df_party['PartyStatus'] == 'Republican') | (df_party['Pa
dem_rep.loc[:, 'Seats'] = dem_rep['Seats'].replace('.', np.NaN)
dem_rep.loc[:, 'Seats'] = pd.to_numeric(dem_rep['Seats'])

#Code in block used to determine the majority party.
############
dem_rep_copy = dem_rep.copy()
dem_rep_copy['Years'] = pd.to_datetime(dem_rep_copy['Years'].str.split('-',

# Group by 'Congress' and find the party with the maximum seats for each yea
idx = dem_rep_copy.groupby(['Congress'])['Seats'].transform(max) == dem_rep_
max_seats = dem_rep_copy[idx]
years_of_interest = max_seats[(max_seats['Years'].dt.year >= 1999) & (max_se

years_of_interest.set_index('Years', inplace=True)

# Resample the data to quarterly frequency
df_quarterly = years_of_interest.resample('Q').ffill()

# Sum the values for each quarter-year combination
df_quarterly['PartyStatus'] = df_quarterly['PartyStatus'].ffill()
df_quarterly.reset_index(inplace=True)

start_date = 2019
end_date = 2020
years_of_interest = df_quarterly[(df_quarterly['Years'].dt.year >= start_dat
#######
```

```
temp_df = cleaned_df_money_velocity.reset_index()
date_copy = temp_df.copy()
party_df = date_copy['DATE'].to_frame()

#Manually fill the party that was the majority in the quarter
party_lst = ['Republican', 'Republican', 'Republican', 'Republican', 'Republ
party_df['Party'] = party_lst

#Function to turn party into binary
def party_binary(party):
    if party == 'Republican':
        return 1
    else:
        return 0

party = party_df['Party'].apply(party_binary)
party
```

```
/tmp/ipykernel_488/13131794.py:12: UserWarning: Could not infer format, so e
ach element will be parsed individually, falling back to `dateutil`. To ensu
re parsing is consistent and as-expected, please specify a format.
  dem_rep_copy['Years'] = pd.to_datetime(dem_rep_copy['Years'].str.split('-
', expand=True)[0])
```

Out[ ]:
```
0     1
1     1
2     1
3     1
4     1
     ..
79    0
80    0
81    0
82    0
83    0
Name: Party, Length: 84, dtype: int64
```

Concatenate new data to main dataframe

In [ ]:
```
# Concatenate gini coefficient and party data to main dataframe
finance_df_updated = pd.concat([finance_df, us_gini_coef, party], axis=1)
finance_df_updated = finance_df_updated.loc[:,~finance_df_updated.columns.du
finance_df_updated
```

Out[ ]:

	DATE	Federal Funds Effective Rate (Interest Rate) (%)	Unemployment Rate (%)	Corporate Profits After Tax (USD in Billions)	Avg_EPU_Index	Quarterly Change in S&P500 (%)	M2V	C
0	2000-01-01	5.68	4.033	565.043	81.666667	-0.65	2.135	
1	2000-04-01	6.27	3.933	559.322	89.000000	1.39	2.151	
2	2000-07-01	6.52	4.000	554.170	73.666667	1.50	2.141	
3	2000-10-01	6.47	3.900	533.362	108.333333	-9.83	2.133	
4	2001-01-01	5.59	4.233	562.885	112.666667	-4.72	2.085	
...	...	...	...	...	...	...	...	
79	2019-10-01	1.64	3.600	2120.145	131.000000	6.90	1.438	
80	2020-01-01	1.26	3.800	2009.635	191.666667	-8.37	1.391	
81	2020-04-01	0.06	12.967	1837.522	283.333333	7.52	1.128	
82	2020-07-01	0.09	8.833	2534.949	249.333333	7.62	1.176	
83	2020-10-01	0.09	6.767	2367.466	247.333333	9.21	1.163	

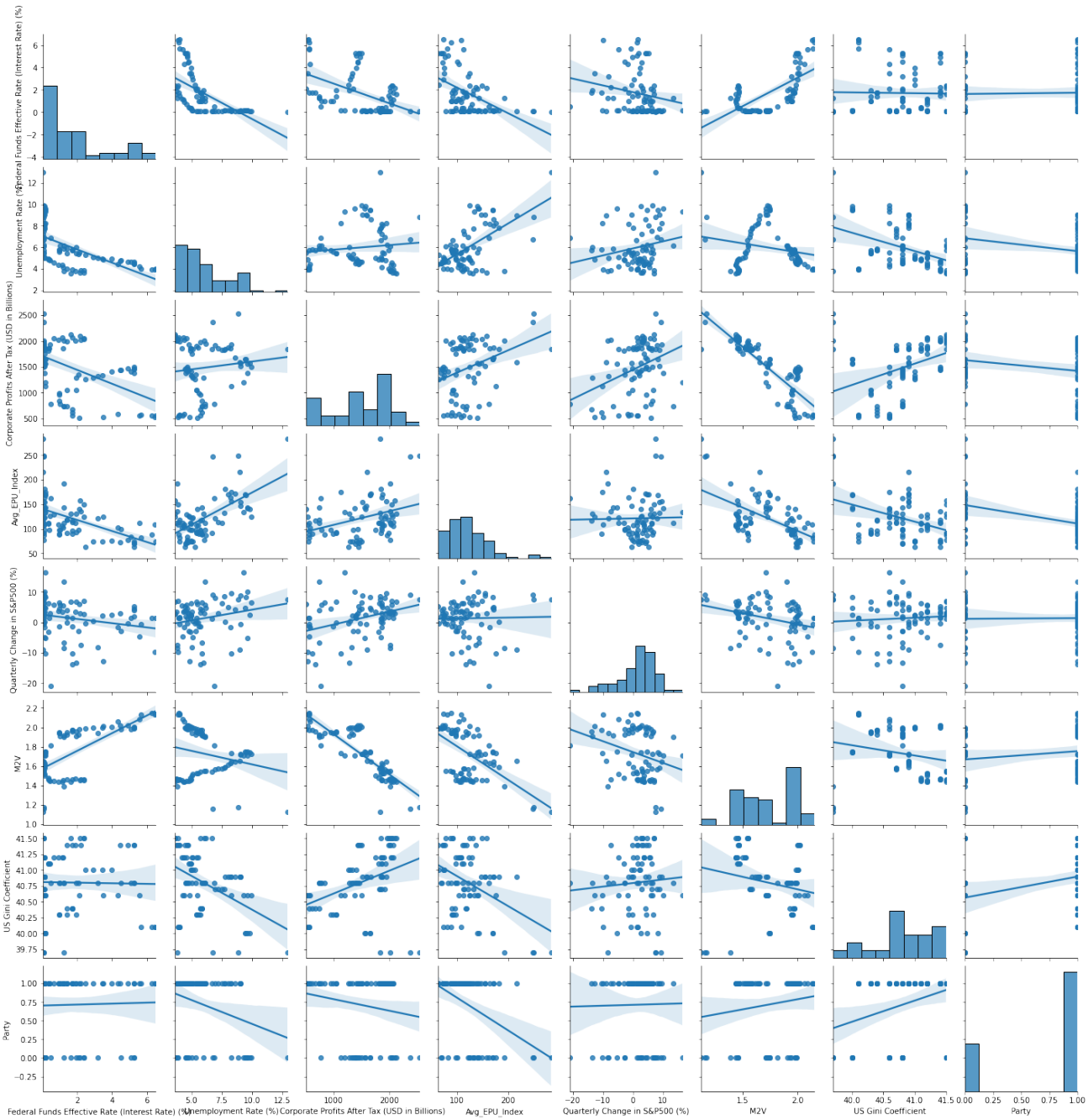
84 rows × 9 columns

Create New Pair Plot

In this pair plot we are looking for relationships between the newly added variables, US gini coefficient and majority party in congress, and the other independent variables. The gini coefficient data seems to have a somewhat linear relationship with money velocity.

In [ ]:
```
plt.figure(figsize=(15, 6))
sns.pairplot(finance_df_updated,kind="reg")
plt.show()
```

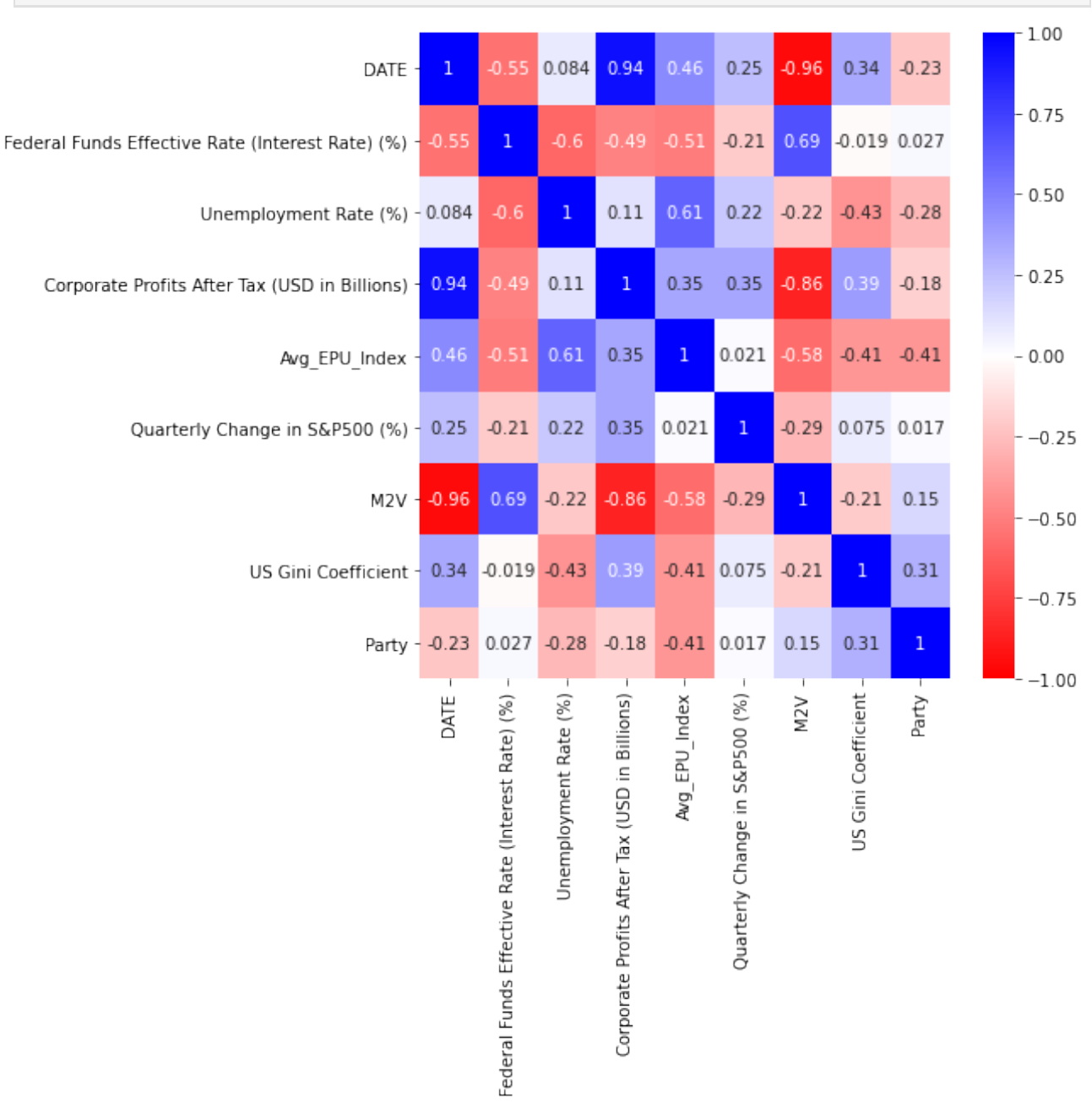
```
<Figure size 1080x432 with 0 Axes>
```



New Heatmap

In this heatmap we are checking the pearson correlation coefficient between the newly added variables and the other independent variables. Fortunately for our analysis, neither appears to be too strongly correlated with any of our other independent variables.

In [ ]:
```
plt.figure(figsize=(7, 7))
heatmap2=sns.heatmap(finance_df_updated.corr(),vmin=-1,vmax=1,annot=True, cm
```

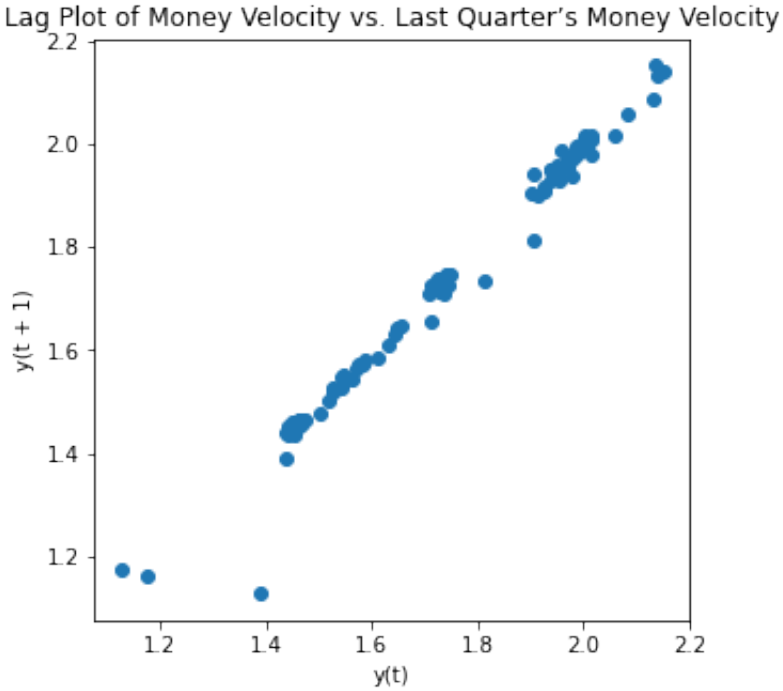


Section 3 of EDA - Autocorrelation Analysis

In this section, we check **Assumption 3 (no auto-correlation).** We graph lag plots for each of our variables to check for autocorrelation and seasonality. A lag plot is a tool used to examine whether time series data is random or has some sort of inherent structure or pattern. In a lag plot, data points from the original time series are plotted against their values at a previous time (lagged values). Knowing if there is autocorrelation present in our data will help us select the correct models for analysis and better interpret the validity of our results.

Money Velocity Lag Plot (Quarterly)

In [ ]:
```
# the lag_plot function requires the data to be a Series type
M2V_series = pd.Series(finance_df_updated['M2V'])

# set figure size
plt.figure(figsize=(5,5))

# lag=1 means we are comparing each data point with the previous data point
# since our data is quarterly, we are comparing each quarter with the previc
pd.plotting.lag_plot(M2V_series, lag=1)

plt.title('Lag Plot of Money Velocity vs. Last Quarter's Money Velocity')
plt.show()
```

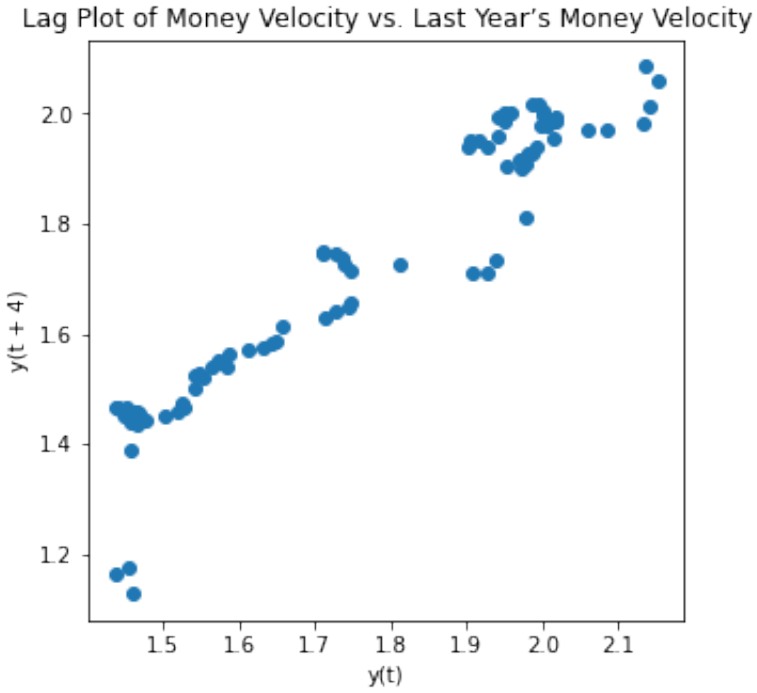


From the above plot we can see there is a strong positive autocorellation with 1-quarter differences in the data. Let's graph it again with the lag set to 4 to see the relationship of money velocity with its corresponding values from the previous year.

Money Velocity Lag Plot (Yearly)

In [ ]:
```
# set figure size
plt.figure(figsize=(5,5))

# lag=4 means we are comparing each quarter's value with the previous years
pd.plotting.lag_plot(M2V_series, lag=4)

plt.title('Lag Plot of Money Velocity vs. Last Year's Money Velocity')
```

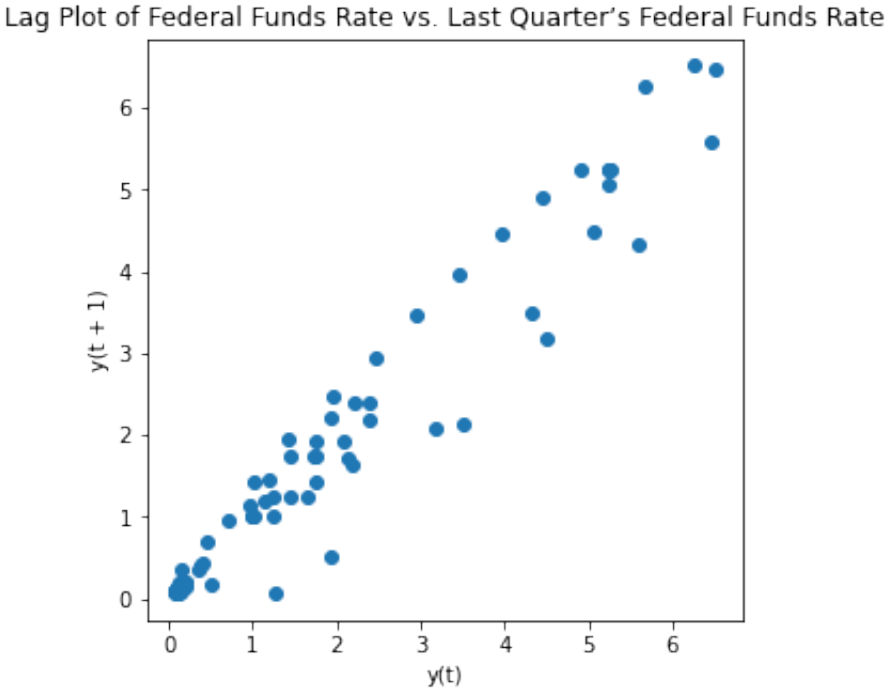
```
plt.show()
```



Once again, we can see that there is strong positive autocorrelation for the money velocity of one quarter compared to the money velocity from the same quarter of the previous year.

Federal Funds Rate Lag Plot (Quarterly)

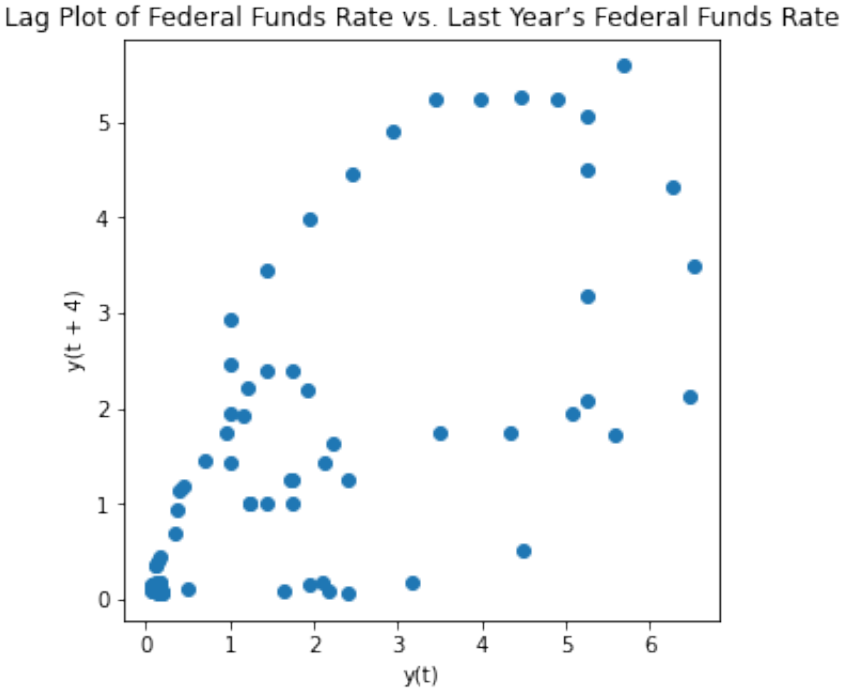
In [ ]:
```
# create Series for federal funds rate data
fed_funds_rate_series = pd.Series(finance_df_updated[fed_funds_rate])

# create plot
plt.figure(figsize=(5,5))
pd.plotting.lag_plot(fed_funds_rate_series, lag=1)
plt.title('Lag Plot of Federal Funds Rate vs. Last Quarter's Federal Funds R
plt.show()
```



Like the previous graphs, the federal funds rate also exhibits strong positive autocorrelation with lag set to 1. Intuitively this makes sense because the Federal Reserve does not drastically change the interest rate across months or quarters, instead it makes very small adjustments over a long period of time so as not to shock the economy. This means the interest rate for one quarter will likely be only a small change from the previous quarter's interest rate, if not the same. Lets take a look at a graph of the federal funds rate versus the previous years rate.

Federal Funds Rate Lag Plot (Yearly)

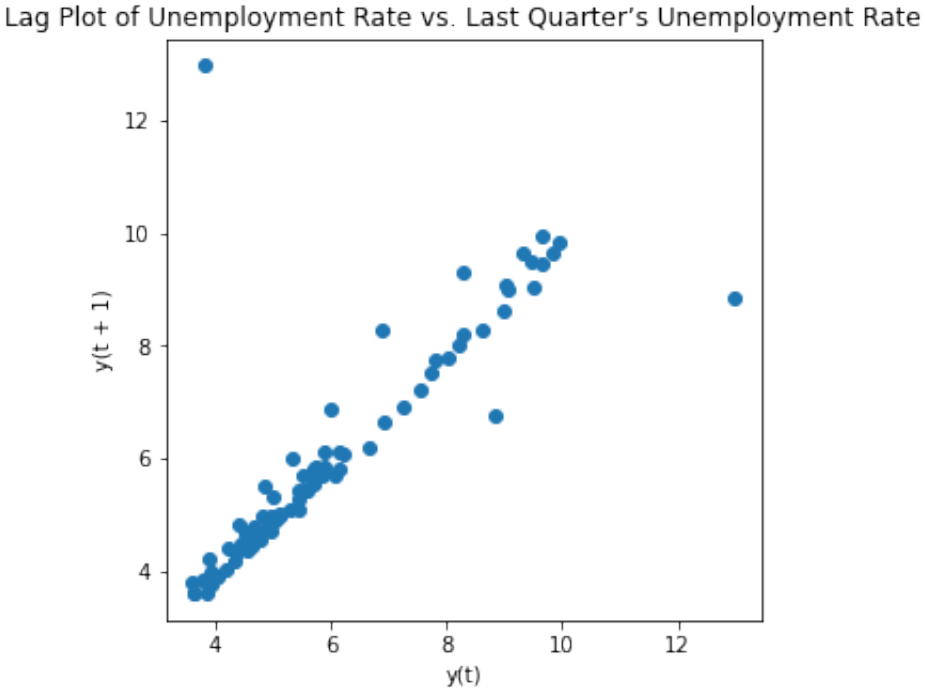
In [ ]:
```
# create plot
plt.figure(figsize=(5,5))
pd.plotting.lag_plot(fed_funds_rate_series, lag=4)
plt.title('Lag Plot of Federal Funds Rate vs. Last Year's Federal Funds Rate
plt.show()
```



Interestingly, this plot is a lot more randomly scattered and exhibits much less of a pattern than the previous graph. Although it still appears that there is a weak positive correlation or even slight cyclical nature to the data given the vague circular shape. But this data is a bit tricky to visualize because a good portion of the points are clustered at (0,0), which represents when the Federal Reserve kept the interest rate close to 0 from 2009 to 2015.

Unemployment Rate Lag Plot (Quarterly)

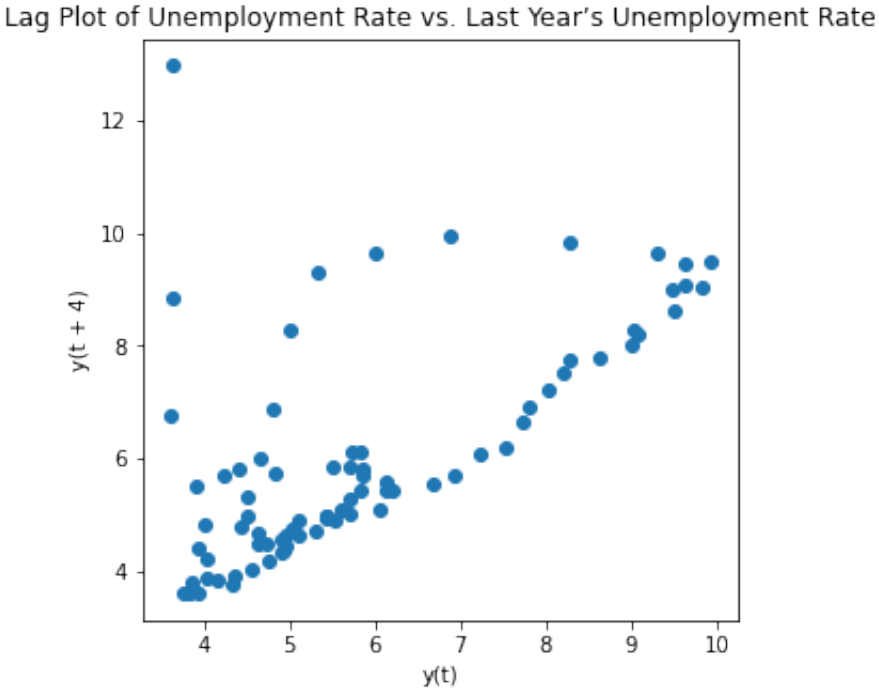
In [ ]:
```
# create Series for unemployment rate data
unemployment_rate_series = pd.Series(finance_df_updated[unemployment_rate])

# create plot
plt.figure(figsize=(5,5))
pd.plotting.lag_plot(unemployment_rate_series, lag=1)
plt.title('Lag Plot of Unemployment Rate vs. Last Quarter's Unemployment Rat
plt.show()
```



It seems we have a few outliers where the previous quarter's unemployment rate was much higher or lower than the quarter being looked at. But there is still strong positive autocorrelation in unemployment rate with a 1-quarter difference. Now let's look at 1-year difference.

Unemployment Rate Lag Plot (Yearly)

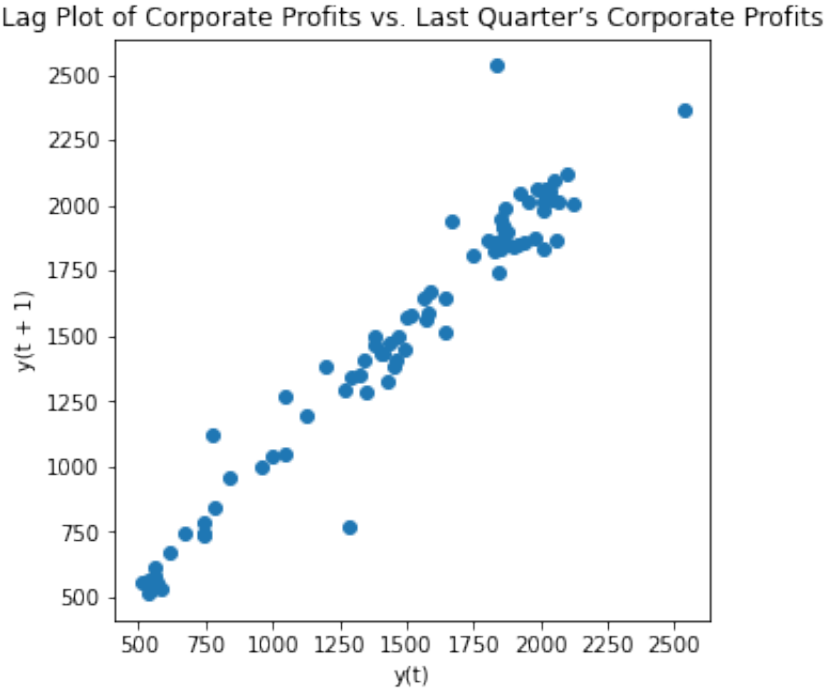
In [ ]:
```
# create plot
plt.figure(figsize=(5,5))
pd.plotting.lag_plot(unemployment_rate_series, lag=4)
plt.title('Lag Plot of Unemployment Rate vs. Last Year's Unemployment Rate')
plt.show()
```



While there is a bit more spread in the data points, there is still a strong positive autocorrelation.

Corporate Profits Lag Plot

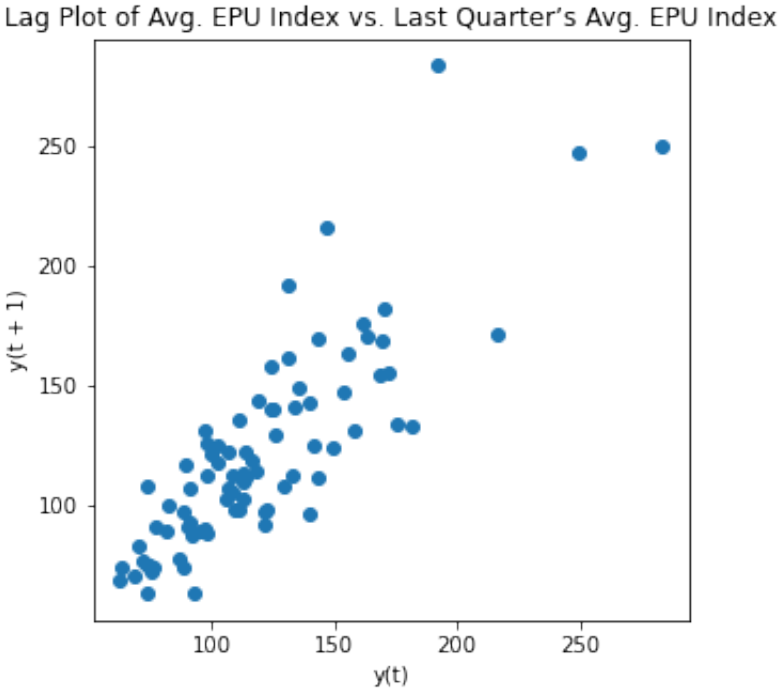
In [ ]:
```
# create Series for corporate profits data
corp_profit_series = pd.Series(finance_df_updated[corp_profits])

# create plot
plt.figure(figsize=(5,5))
pd.plotting.lag_plot(corp_profit_series, lag=1)
plt.title('Lag Plot of Corporate Profits vs. Last Quarter's Corporate Profit
plt.show()
```



We can see that there is strong positive autocorrelation in our corporate profits data. This also makes sense intuitively because corporate profits generally don't change drastically between periods, unless under extreme economic circumstances like a recession. I will skip the yearly plot for this data as it looks essentially the same but the points are slightly more spread out.

Average Economic Policy Uncertainty Index Lag Plot (Quarterly)

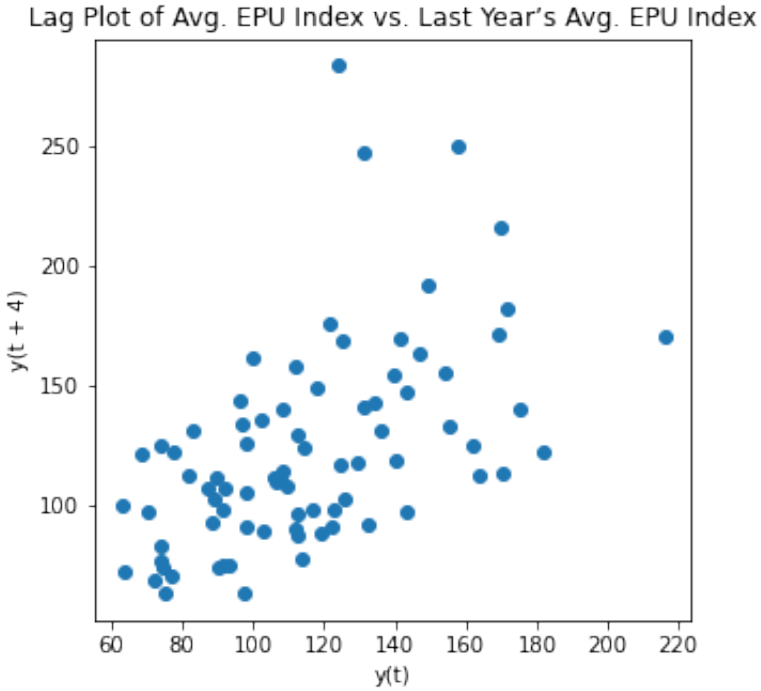
In [ ]:
```
# create Series for EPU data
EPU_series = pd.Series(finance_df_updated[avg_EPU_index])

# create plot
plt.figure(figsize=(5,5))
pd.plotting.lag_plot(EPU_series, lag=1)
plt.title('Lag Plot of Avg. EPU Index vs. Last Quarter's Avg. EPU Index')
plt.show()
```



Interestingly, this data is a bit more scattered compared to the previous lag plots, but it still shows strong positive autocorrelation. This makes sense because uncertainty regarding economic policy likely won't shift drastically between one day and the next (or one quarter and the next in our case). It takes time for people to develop their stance on new economic policy, hence the correlation between average EPU index values and their previous values.

Average Economic Policy Uncertainty Index Lag Plot (Yearly)

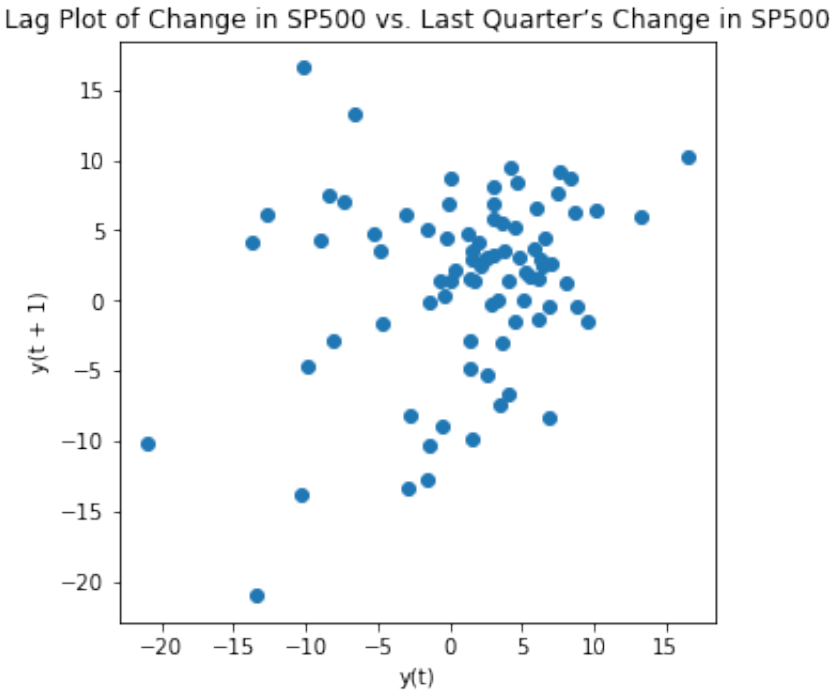
In [ ]:
```
# create plot
plt.figure(figsize=(5,5))
pd.plotting.lag_plot(EPU_series, lag=4)
plt.title('Lag Plot of Avg. EPU Index vs. Last Year's Avg. EPU Index')
plt.show()
```



When we plot average EPU values against the previous year's values, it looks somewhat like a "shotgun" where the points are more scattered. While there is a lot more scatter than in previous graphs, there is still a weak positive autocorrelation.

Change in S&P 500 (Quarterly)

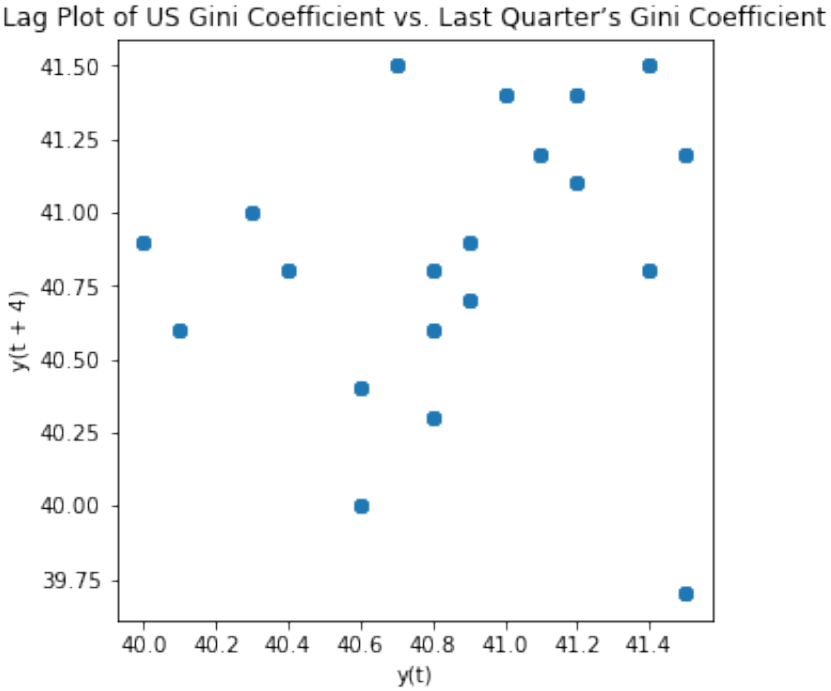
In [ ]:
```
# create Series for Change in S&P 500 data
delta_sp_series = pd.Series(finance_df_updated[delta_sp])

# create plot
plt.figure(figsize=(5,5))
pd.plotting.lag_plot(delta_sp_series, lag=1)
plt.title('Lag Plot of Change in SP500 vs. Last Quarter's Change in SP500')
plt.show()
```



There seems to be no autocorrelation in the change in S&P 500 data. This makes sense because stocks tend to follow a "random walk" where the movements in any given short period of time are mostly random.

US Gini Coefficient (Yearly)

In [ ]:
```
# create Series for US Gini Coefficient data
gini_series = pd.Series(finance_df_updated['US Gini Coefficient'])

# create plot
plt.figure(figsize=(5,5))
pd.plotting.lag_plot(gini_series, lag=4)
plt.title('Lag Plot of US Gini Coefficient vs. Last Quarter's Gini Coefficie
plt.show()
```



There doesn't appear to be any autocorrelation between the gini coefficient in one year versus the previous year. However, if we were to change the lag to be previous quarter in instead of previous year there would be strong autocorrelation since we had to copy the yearly values across each quarter in that given year.

Block Bootstrapping & Multivariate Regression Inference

Block bootstrapping is a resampling technique used in the context of time series data regression when there is autocorrelation in the independent variables. Autocorrelation implies that the observations at different time points are correlated with each other, violating the assumption of independence typically required by standard bootstrap methods. Block bootstrapping is a way to account for this correlation structure in time series data.

The key idea behind block bootstrapping is to resample contiguous blocks or segments of observations from the time series instead of individual observations. By doing so, it preserves the temporal structure and dependencies within each block, making it a more suitable method for time series data.

Data Preparation: Convert time series data to a numpy array (data = finance_df.values).

Block Size: Define the block size as the total number of observations (block_size = data.shape[0]).

Bootstrap Samples: Specify 1000 bootstrap samples to be generated (num_bootstrap_samples = 1000).

Bootstrapping Loop: Iterate through bootstrap samples. Randomly choose indices with replacement to create a bootstrap sample using np.random.choice. Form contiguous blocks of observations based on the selected indices. Store each bootstrap sample in an array (bootstrapped_samples).

Calculate Bootstrap Statistics:Compute the mean for each variable across bootstrap samples (bootstrap_means = np.mean(bootstrapped_samples, axis=1)).

Regression Analysis:Utilize bootstrapped means for regression. Extract the dependent variable (y) and independent variables (X) from bootstrapped means.

Regression Model Fitting: Fit a multivariate regression model with heteroskedastic-robust standard errors using sm.OLS(cov_type='HC3').Present regression results using results.summary().

In [ ]:
```
finance_df_updated = finance_df_updated.set_index('DATE')
finance_df_updated
```

Out[ ]:

DATE	Federal Funds Effective Rate (Interest Rate) (%)	Unemployment Rate (%)	Corporate Profits After Tax (USD in Billions)	Avg_EPU_Index	Quarterly Change in S&P500 (%)	M2V	US Coeffi
2000-01-01	5.68	4.033	565.043	81.666667	-0.65	2.135	
2000-04-01	6.27	3.933	559.322	89.000000	1.39	2.151	
2000-07-01	6.52	4.000	554.170	73.666667	1.50	2.141	
2000-10-01	6.47	3.900	533.362	108.333333	-9.83	2.133	
2001-01-01	5.59	4.233	562.885	112.666667	-4.72	2.085	
...	...	...	...	...	...	...	
2019-10-01	1.64	3.600	2120.145	131.000000	6.90	1.438	
2020-01-01	1.26	3.800	2009.635	191.666667	-8.37	1.391	
2020-04-01	0.06	12.967	1837.522	283.333333	7.52	1.128	
2020-07-01	0.09	8.833	2534.949	249.333333	7.62	1.176	
2020-10-01	0.09	6.767	2367.466	247.333333	9.21	1.163	

84 rows × 8 columns

Original Regression Model

We originally performed the multivariate regression with the dependent variable M2V and independent variables: Federal Funds Effective Rate (Interest Rate), Unemployment Rate, Corporate Profits After Tax, Avg_EPU_Index, and Change in S&P. The following is the regression result.

In [ ]:
```
# Set seed for reproducability
np.random.seed(4321)

#Standardize to compare vars
df_z = finance_df_updated.select_dtypes(include=[np.number]).dropna().apply(


# The code of this function is mostly generated by GPT-4 from the prompt "wr
# Edits were made to covariance type to work with heteroskedasticity
# Set X and Y Vals
X_vals = df_z.drop(columns=['M2V','US Gini Coefficient','Party'])
Y_vals = df_z['M2V']

# Function that performs block bootstrapping
def block_bootstrap(X, y, block_size):
    n = len(X)
    num_blocks = int(np.ceil(n / block_size))
    indices = np.arange(n)
    bootstrap_indices = np.zeros((num_blocks, block_size), dtype=int)
    for i in range(num_blocks):
        bootstrap_indices[i] = np.random.choice(indices, size=block_size, re
    return X.iloc[bootstrap_indices.flatten()], y.iloc[bootstrap_indices.fla

# Perform block bootstrapping on the data
X_boot, y_boot = block_bootstrap(X_vals, Y_vals, block_size=10)

# Fit a linear regression model to the bootstrapped data with heteroskedasti
model = sm.OLS(y_boot, X_boot).fit(cov_type='HC3')

model.summary()
```

Out[ ]:

OLS Regression Results

Dep. Variable:	M2V	**R-squared (uncentered):**	0.935
Model:	OLS	**Adj. R-squared (uncentered):**	0.931
Method:	Least Squares	**F-statistic:**	257.9
Date:	Thu, 14 Dec 2023	**Prob (F-statistic):**	8.40e-50
Time:	03:37:17	**Log-Likelihood:**	-9.3552
No. Observations:	90	**AIC:**	28.71
Df Residuals:	85	**BIC:**	41.21
Df Model:	5		
Covariance Type:	HC3		

	coef	std err	z	P>|z|	[0.025	0.975]
Federal Funds Effective Rate (Interest Rate) (%)	0.3949	0.066	5.954	0.000	0.265	0.525
Unemployment Rate (%)	0.2928	0.081	3.636	0.000	0.135	0.451
Corporate Profits After Tax (USD in Billions)	-0.5818	0.028	-20.439	0.000	-0.638	-0.526
Avg_EPU_Index	-0.3713	0.056	-6.616	0.000	-0.481	-0.261
Quarterly Change in S&P500 (%)	-0.0770	0.032	-2.387	0.017	-0.140	-0.014

Omnibus:	17.958	**Durbin-Watson:**	2.218
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	30.072
Skew:	-0.803	**Prob(JB):**	2.95e-07
Kurtosis:	5.332	**Cond. No.**	3.09

Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors are heteroscedasticity robust (HC3)

Before adjusting variables for the OLS regression, it is important to note that including Interest Rate, Unemployment Rate, Corporate Profits After Tax, Average_EPU_Index, and

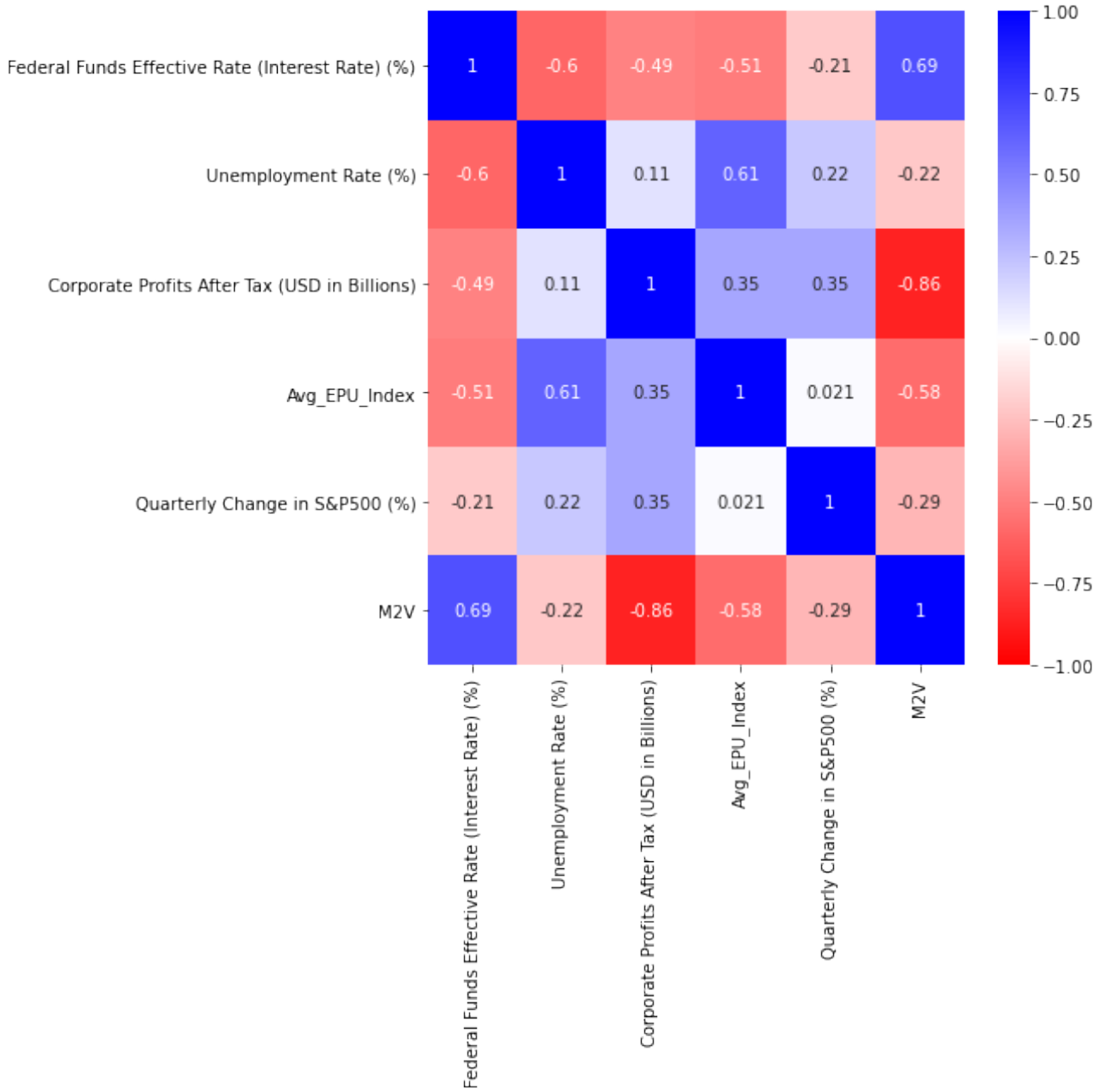
Change in S&P may lead to a strong multicollinearity problem. This issue arises due to significant correlations among these independent variables.

Also the use of Unemployment Rate and Interest Rate also leads to incorrect linear regression analysis as it does not have a linear relationship with money velocity as seen by the pair plot graph.

The heatmap analysis reveals that Avg_EPU_Index exhibits a relatively high positive correlation with Unemployment Rate, contributing to multicollinearity. Additionally, Federal Funds Effective Rate (Interest Rate) demonstrates strong correlations with Unemployment Rate, Corporate Profits After Tax, and Avg_EPU_Index.

To address this multicollinearity concern, the Unemployment Rate variable will be removed, and the Change in Interest Rate will be used instead of the Interest Rate variable. This strategic adjustment aims to mitigate the multicollinearity issue and enhance the reliability of the regression analysis.

In [ ]:
```
plt.figure(figsize=(7, 7))
heatmap=sns.heatmap(finance_df.drop(columns=['DATE']).corr(),vmin=-1,vmax=1,
```



Adjusted Regression Model

To address the challenge of multicollinearity, we addressed it by excluding the Unemployment Rate and replacing the Interest Rate variable with its change over time. Additionally, we substituted the Corporate Profit variable with its change. While removing the Unemployment Rate could result in a reduction of information available for the regression model, leading to less explained variation in the dependent variable (Y), we sought to enhance precision by introducing additional variables—specifically, the Gini Coefficient and the Party variable representing the majority party in Congress.

Subsequently, we conducted a multivariate regression with M2V as the dependent variable and the following independent variables: Avg_EPU_Index, Change in S&P, Change in Interest Rate, Change in Corporate Profit, US Gini Coefficient, and Party.

Calculating Changes

Change in Interest Rate

In [ ]:
```
cleaned_df_fedfunds_change  = cleaned_df_fedfunds.diff()
cleaned_df_fedfunds_change = cleaned_df_fedfunds_change.drop(cleaned_df_fedf
cleaned_df_fedfunds_change = cleaned_df_fedfunds_change.rename({'Federal Fur
temp_df = cleaned_df_fedfunds_change.reset_index()
temp_df['DATE'] = pd.to_datetime(temp_df['DATE'])
temp_df = temp_df.set_index('DATE')
cleaned_df_fedfunds_change = temp_df

cleaned_df_fedfunds_change
```

Out[ ]:

DATE	Change in Interest Rate (%)
2000-01-01	0.37
2000-04-01	0.59
2000-07-01	0.25
2000-10-01	-0.05
2001-01-01	-0.88
...	...
2019-10-01	-0.55
2020-01-01	-0.38
2020-04-01	-1.20
2020-07-01	0.03
2020-10-01	0.00

84 rows × 1 columns

Change in Corporate Profits After Tax

In [ ]:
```
corp_profits_after_tax_change = corp_profits_after_tax.diff()
```

```
corp_profits_after_tax_change = corp_profits_after_tax_change.drop(corp_prof
corp_profits_after_tax_change = corp_profits_after_tax_change.rename({'Corpo
corp_profits_after_tax_change.reset_index(inplace=True)
corp_profits_after_tax_change['DATE'] = pd.to_datetime(corp_profits_after_ta
corp_profits_after_tax_change = corp_profits_after_tax_change.set_index('DAT

corp_profits_after_tax_change
```

Out[ ]:

DATE	Change in Corporate Profits After Tax (USD in Billions)
2000-01-01	-25.949
2000-04-01	-5.721
2000-07-01	-5.152
2000-10-01	-20.808
2001-01-01	29.523
...	...
2019-10-01	20.664
2020-01-01	-110.510
2020-04-01	-172.113
2020-07-01	697.427
2020-10-01	-167.483

84 rows × 1 columns

In [ ]:

```
finance_df_updated_final = pd.concat([finance_df_updated, cleaned_df_fedfunc
finance_df_updated_final = finance_df_updated_final.loc[:,~finance_df_update
finance_df_updated_final
```

Out[ ]:

DATE	Federal Funds Effective Rate (Interest Rate) (%)	Unemployment Rate (%)	Corporate Profits After Tax (USD in Billions)	Avg_EPU_Index	Quarterly Change in S&P500 (%)	M2V	US Coeffi
2000-01-01	5.68	4.033	565.043	81.666667	-0.65	2.135	
2000-04-01	6.27	3.933	559.322	89.000000	1.39	2.151	
2000-07-01	6.52	4.000	554.170	73.666667	1.50	2.141	
2000-10-01	6.47	3.900	533.362	108.333333	-9.83	2.133	
2001-01-01	5.59	4.233	562.885	112.666667	-4.72	2.085	
...	...	...	...	...	...	...	
2019-10-01	1.64	3.600	2120.145	131.000000	6.90	1.438	
2020-01-01	1.26	3.800	2009.635	191.666667	-8.37	1.391	
2020-04-01	0.06	12.967	1837.522	283.333333	7.52	1.128	
2020-07-01	0.09	8.833	2534.949	249.333333	7.62	1.176	
2020-10-01	0.09	6.767	2367.466	247.333333	9.21	1.163	

84 rows × 10 columns

In [ ]:
```
# Set seed for reproducability
np.random.seed(4321)

# standardizing dataframe to compare coefficients (Compute the Z-score of ea
df_z = finance_df_updated_final.select_dtypes(include=[np.number]).dropna().

# The code of this function is mostly generated by GPT-4 from the prompt "wr
# Edits were made to covariance type to work with heteroskedasticity
X_vals = df_z.drop(columns=['M2V', unemployment_rate, fed_funds_rate, 'Corpo
```

```
Y_vals = df_z['M2V']

# Function that performs block bootstrapping
def block_bootstrap(X, y, block_size):
    n = len(X)
    num_blocks = int(np.ceil(n / block_size))
    indices = np.arange(n)
    bootstrap_indices = np.zeros((num_blocks, block_size), dtype=int)
    for i in range(num_blocks):
        bootstrap_indices[i] = np.random.choice(indices, size=block_size, re
    return X.iloc[bootstrap_indices.flatten()], y.iloc[bootstrap_indices.fla

# Perform block bootstrapping on the data
X_boot, y_boot = block_bootstrap(X_vals, Y_vals, block_size=10)

# Fit a linear regression model to the bootstrapped data with heteroskedasti
model = sm.OLS(y_boot, X_boot).fit(cov_type='HC3')

model.summary()
```

Out[ ]:

OLS Regression Results

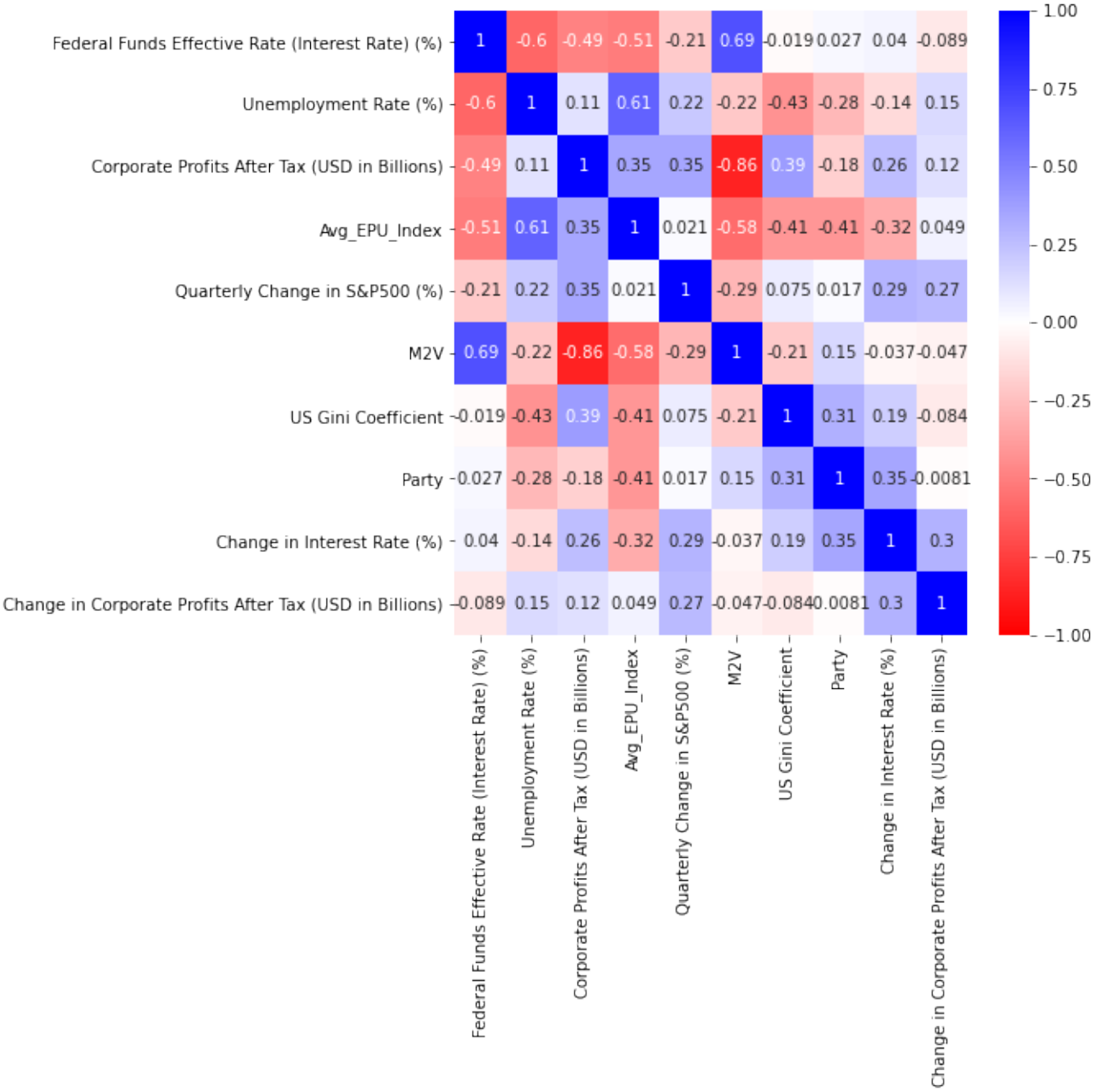
Dep. Variable:	M2V	**R-squared (uncentered):**	0.699
Model:	OLS	**Adj. R-squared (uncentered):**	0.677
Method:	Least Squares	**F-statistic:**	74.41
Date:	Thu, 14 Dec 2023	**Prob (F-statistic):**	1.64e-31
Time:	03:37:17	**Log-Likelihood:**	-78.061
No. Observations:	90	**AIC:**	168.1
Df Residuals:	84	**BIC:**	183.1
Df Model:	6		
Covariance Type:	HC3		

	coef	std err	z	P>|z|	[0.025	0.975]
Avg_EPU_Index	-0.9216	0.104	-8.870	0.000	-1.125	-0.718
Quarterly Change in S&P500 (%)	-0.2744	0.073	-3.758	0.000	-0.418	-0.131
US Gini Coefficient	-0.5126	0.063	-8.073	0.000	-0.637	-0.388
Party	0.0122	0.072	0.168	0.867	-0.130	0.154
Change in Interest Rate (%)	-0.2051	0.082	-2.516	0.012	-0.365	-0.045
Change in Corporate Profits After Tax (USD in Billions)	0.1171	0.090	1.305	0.192	-0.059	0.293

Omnibus:	1.570	**Durbin-Watson:**	1.974
Prob(Omnibus):	0.456	**Jarque-Bera (JB):**	1.600
Skew:	0.261	**Prob(JB):**	0.449
Kurtosis:	2.607	**Cond. No.**	2.25

Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors are heteroscedasticity robust (HC3)

In [ ]:
```
plt.figure(figsize=(7, 7))
heatmap=sns.heatmap(finance_df_updated_final.corr(),vmin=-1,vmax=1,annot=Tru
```



Interpretation of the Regression Results

To accurately compare what coefficients are most significant in our regression analysis, we needed to standardize them. Otherwise if we did not, the coefficients would have different units which means the effect they have on the predictor variable could not be accurately compared since the numbers would mean drastically different things. However, by standardizing the coefficients we also change how the coefficients are interpreted, as they no longer represent the effect one unit change on the predictor variable has on the response variable. Instead, the coefficients now represent a one standard deviation increase in an independent variable leads to an increase or decrease

in standard deviation on the dependent variable.

Avg_EPU_Index (Average Economic Policy Uncertainty Index): A one standard deviation increase in Avg_EPU_Index is associated with a 0.9216 standard deviaton decrease in M2V, holding other variables constant. This suggests that a rise in economic policy uncertainty is linked to a decrease in money velocity. We can consider this variable statistically significant as the p-value for the coefficient is extremely small where it registers as 0, which is less than the significance level .05.

Change in S&P (Quarterly Change in S&P500): A one standard deviation increase in the quarterly change in S&P500 is associated with a 0.2744 standard deviation decrease in M2V. This implies that a negative change in the S&P500 index is linked to an slight increase in money velocity. We can consider this variable statistically significant as the p-value for the coefficient is extremely small where it registers as 0, which is less than the significance level .05.

Int_change (Change in Interest Rate): A one standard deviation increase in the change in interest rate is associated with a standard deviation decrease of 0.2051 in M2V. This indicates a negative relationship, suggesting that a decline in interest rates is associated with a reduction in money velocity. We can consider this variable statistically significant as the p-value for the coefficient is .012, which is less than the significance level .05.

CP_change (Change in Corporate Profits After Tax): A one standard deviation increase in the change in corporate profits is associated with a 0.1171 standard deviation increase in M2V. This implies that an increase in corporate profits is linked to a slight rise in money velocity. We cannot consider this variable statistically significant as the p-value for the coefficient is larger than the significance level .05, so this cannot be a contender for what variable most effect money velocity.

US Gini Coefficient: A one standard deviation increase in the Gini Coefficient is associated with a 0.5126 standard deviation decrease in M2V. This suggests that higher wealth inequality is linked to a lower money velocity. We can consider this variable statistically significant as the p-value for the coefficient is extremely small where it registers as 0, which is less than the significance level .05.

Party: The coefficient for the Party variable is 0.0122. Because it is not statistically significant at the 0.05 significance level (p-value > 0.05), the majority party in Congress may not have a significant impact on money velocity in this model. We cannot consider this variable statistically significant as the p-value for the coefficient is much larger than the significance level .05, so this cannot be a contender for what variable most effect money velocity.

The **R-squared** value for the regression model is 69.9%, indicating that approximately 69.9% of the variability in the dependent variable, M2V (Money Velocity), can be accounted for by the changes in the independent variables. This relatively high R-squared value signifies an good level of fit for the model, suggesting that the selected regressors capture and explain most of the variations observed in money velocity.

Pearson Correlation Coefficient

Using the Pearson Correlation Coefficients that we receive from the table, we can also see that our results align with the what we found in the linear regression model with the relationship between money velocity and Avg_EPU_Index being -.58. We ignore variables that are not included in the regression since we could not determine their statistical significance through linear regression.

Ethics & Privacy

We will consider the transparency and accountability principle specifically by maintaining transparency about data sources and the research process. We will clearly document the sources of financial and economics data and the steps taken in data processing and analysis to ensure accountability and transparency.

We will consider the unbiased data principle by mitigating biases in the data source. For example, when we collect data about political stability, we will cross check data from multiple sources to ensure that the data well represents the reality instead of having political biases.

We will implement the continuous monitoring principle by regularly monitoring for potential unintended consequences. If the research reveals unexpected factors that influence money velocity, we will prospect and monitor for any unintended societal or economic impacts of the study result. For example, if the study reveals that big corporations or economic activities of a particular racial group have a huge weight in regulating money velocity, we will consider whether revealing this information might impose societal conflicts.

We will consider the public awareness principle by making the public aware of the project's purpose and potential impacts. Specifically, we will provide information to the public through clear project descriptions and public announcements, allowing individuals to make informed decisions about what this study is about and what the implication of

the study be.

We will consider the harm minimization principle specifically by taking steps to prevent harm arising from the research. If findings indicate potential economic vulnerabilities, we will do our best to make suggestions and even collaborate with policymakers to mitigate negative economic consequences and prevent harm to society.

Discusison and Conclusion

In our project we set out to identify which macroeconomic indicators are most closely related to money velocity. Interestingly, we discovered that the **Economic Policy Uncertainty Index** is the most statistically significant variable in modelling money velocity. The reasoning behind this could be that when people are more uncertain about economic policy, and hence more uncertain about future economic conditions, they save their money which results in lower money velocity. This is important because it means policymakers can potentially influence consumer saving habits through the policies they implement, in turn helping the US economy recover from a recession or cool down after a boom. It could also mean that large media and publishing companies have a significant influence on money velocity. This is because the index was calculated in part using the language in articles about economic policies, so if these companies create an uncertain economic outlook they may influence money velocoity negatively, regardless of the true intentions of policymakers.

We used linear regression to determine if there is a statistically significant relationship between our independent variables and money velocity. In our exploratory analysis, we found that most of our independent variables had autocorrelation. This presented a significant problem because one of the assumptions of linear regression is that the independent variables should not have autocorrelation. We addressed this through block bootstrapping, a technique that allows us to mitigate autocorrelation while retaining the temporal structure and patterns in the data over time.

In addition to the Economic Policy Uncertainty Index, our analysis also reveals statistically significant correlations between **Change in S&P 500, Change in Interest Rate**, and **US Gini Coefficient** with **Money Velocity** in the United States. The identified relationships suggest that changes in these variables are significant indicators of the speed at which money circulates throughout the U.S. economy. Specifically, we observe the following directions of the relationships:

1. **Economic Policy Uncertainty:** As economic policy uncertainty increases, money

velocity tends to decrease.
2. **Change in S&P 500:** A positive change in the S&P 500 index is associated with a decrease in money velocity.
3. **Change in Interest Rate:** A positive change in the federal funds rate is associated with a decrease in money velocity.
4. **US Gini Coefficient:** Higher levels of wealth inequality, indicated by a Gini coefficient closer to 1, are associated with a decrease in money velocity.

Unfortunately, because we were using linear regression we had to eliminate unemployment rate from our analysis due to its non-linear relationship with money velocity. Determining the true relationship between unemployment rate and money velocity will require additional research and analysis that is outside the scope of our project.

Regarding the prior works covered in ouir background section, our analysis provides additional empirical evidence on what factors influence the velocity of money. While Hetzel found in 1993 that money velocity became more sensitive to changes in the interest rate during the 1980s, our findings show that the sensitivity of money velocity to the interest rate may have decreased since then. Moreover, our findings suggest that policymakers should not only consider the direct economic effects of their policies, but also the psychological interpretations of whether their policies will be effective or cause chaos. The uncertainty created by bad policies or fear mongering by politicians appears to have a significant effect on money velocity, as indicated by the inverse relationship between the EPU Index and money velocity.

In conclusion, our project reinforces the importance of considering a multitude of different macroeconomic indicators in analyzing money velocity while also emphasizing the interconnectedness of economic policies, public perception, and media narratives in shaping the economic landscape.

Team Contributions

- Jingwei Guo - Wrote variable descriptions, did pairplot visualizations, run block boostrapping and multivariate regressions
- Sinclair Lim - Imported, wrangled code, worked and researched on EDA for outliers
- Alejandro Vazquez - Imported data, created lag plots, helped with heatmap visualizations, wrote 'background and prior work' and 'discussion and conclusion' sections
- Alisa Vu - Imported & wrangled data; EDA line charts; abstract

- David Yonemura - Debugged, adjusted and cleaned code; introduced statistic methods relevant to project; narrowed project focus