

academic-paper-long-term-investing-in-triple-leveraged-exchange-traded-funds

October 24, 2026

0.1 Introduction

In late 2025, I ran across a paper by Lewis Glenn entitled “Long-Term Investing in Triple Leveraged Exchange Traded Funds.” The paper explores the performance and risks associated with holding triple leveraged ETFs over extended periods, and proposes an investment strategy involving rebalancing two triple leveraged ETFs (abbreviated “TLETFs”) on a bimonthly schedule. From the abstract:

“The phrase long-term investing in triple leveraged ETFs is somewhat of an anathema for investment academicians. As a result of daily rebalancing and so-called beta slippage or “the constant leverage trap” it is highly likely over the long term to significantly deviate from the targeted leverage and, in so doing, generate wipeout losses. In fact, the data show that many 3X leveraged ETFs have performed poorly over the long term in the period of their existence. However, there are some significant exceptions. The 3X leveraged Nasdaq ETF, TQQQ, for example has delivered a total return exceeding 10,000% (CAGR of 54.4%) over the 10+ years of its existence, more than 13 times the total return of its underlying ETF (QQQ). Similar, but less impressive numbers have been achieved with triple leveraged ETFs that target the S&P 500 and the Dow Industrials.”

“These results, however, are accompanied by excessive volatility and very large maximum end-of-month (EOM) draw downs, in excess of 49% for TQQQ, and with maximum daily drawdown of almost 70%. Many retail investors would be unwilling to sustain such (paper) losses. Here we propose a strategy to mitigate these large draw downs while still retaining much of the upside performance of triple leveraging. We show that by simply establishing a portfolio consisting of an equal dollar amount of TQQQ and TMF (the triple leveraged ETF linked to the 20+ year treasury bond), and performing bi-monthly rebalancing, a total return, after transaction costs, in excess of 5,800% (CAGR of 44.9%) would have been achieved over the 10+ years of TQQQ existence, with the maximum EOM draw down less than 25%.”

The paper was published (or dated, at least) November 7, 2020. In this piece, we will attempt to reproduce the results of the investment strategy presented in the paper section by section, and examine whether the impressive performance in sample has continued.

A note on the numbering of figures and tables: figures that reproduce a figure from the paper use the same number as the paper (for example, Figure 1), with a letter added when we split one of the paper’s figures into several plots (for example, Figures 9a, 9b, and 9c). Figure 10 from the paper is reproduced as a table. Figure 6 is omitted due to data limitations, and figure 11 (a pseudo-code

diagram) is discussed but not reproduced. All figures and tables that do not appear in the paper are numbered with an “E” prefix (for example, Figure E1 and Table E1).

0.2 Python Imports

```
[1]: # Standard Library
import os
import sys
import warnings

from pathlib import Path

# Data Handling
import pandas as pd

# Suppress warnings
warnings.filterwarnings("ignore")

[2]: # Add the source subdirectory to the system path to allow import config from
↳ settings.py
current_directory = Path(os.getcwd())
website_base_directory = current_directory.parent.parent.parent
src_directory = website_base_directory / "src"
sys.path.append(str(src_directory)) if str(src_directory) not in sys.path else
↳ None

# Import settings.py
from settings import config

# Add configured directories from config to path
SOURCE_DIR = config("SOURCE_DIR")
(
    sys.path.append(str(Path(SOURCE_DIR)))
    if str(Path(SOURCE_DIR)) not in sys.path
    else None
)

# Add other configured directories
BASE_DIR = config("BASE_DIR")
CONTENT_DIR = config("CONTENT_DIR")
POSTS_DIR = config("POSTS_DIR")
PAGES_DIR = config("PAGES_DIR")
PUBLIC_DIR = config("PUBLIC_DIR")
SOURCE_DIR = config("SOURCE_DIR")
DATA_DIR = config("DATA_DIR")
DATA_MANUAL_DIR = config("DATA_MANUAL_DIR")
```

0.3 Python Functions

Here are the functions needed for this project:

- `load_data`: Load data from a CSV, Excel, or Pickle file into a pandas DataFrame.
- `pandas_set_decimal_places`: Set the number of decimal places displayed for floating-point numbers in pandas.
- `plot_time_series`: Plot the timeseries data from a DataFrame for a specified date range and columns.
- `run_regression`: Run a linear regression using statsmodels OLS and return the results.
- `summary_stats`: Generate summary statistics for a series of returns.
- `yf_pull_data`: Download daily price data from Yahoo Finance and export it.

```
[3]: from load_data import load_data
from pandas_set_decimal_places import pandas_set_decimal_places
from plot_time_series import plot_time_series
from run_regression import run_regression
from summary_stats import summary_stats
from yf_pull_data import yf_pull_data
```

0.4 Data Overview

For this exercise, we will use the following ETFs:

- TQQQ - ProShares UltraPro QQQ
- QQQ - Invesco QQQ Trust
- UPRO - ProShares UltraPro S&P 500
- SPY - SPDR S&P 500 ETF Trust
- UDOW - ProShares UltraPro Dow30
- DIA - SPDR Dow Jones Industrial Average ETF Trust
- TNA - Direxion Daily Small Cap Bull 3X Shares
- IWM - iShares Russell 2000 ETF
- TMF - Direxion Daily 20+ Year Treasury Bull 3X Shares
- TLT - iShares 20+ Year Treasury Bond ETF
- EDC - Direxion Daily MSCI Emerging Markets Bull 3X Shares
- EEM - iShares MSCI Emerging Markets ETF
- IEF - iShares 7-10 Year Treasury Bond ETF (used for the “black swan” filter in section 3.2)
- AGG - iShares Core U.S. Aggregate Bond ETF (used in extension 3)

We’ll use the adjusted close prices for each of these ETFs.

The paper also used the following funds:

- DZK - Direxion Daily MSCI Developed Markets Bull 3X Shares
- EFA - iShares MSCI EAFE ETF

But we will omit these due to DZK being delisted.

0.5 Acquire Data

First, let’s get the data for these ETFs. If we already have the desired data, we can load it from a local pickle file. Otherwise, we can download it from Yahoo Finance using the `yf_pull_data`

function.

```
[4]: pandas_set_decimal_places(2)

# Create a list of the ETF tickers
fund_list = [
    "TQQQ",
    "QQQ",
    "UPRO",
    "SPY",
    "UDOW",
    "DIA",
    "TNA",
    "IWM",
    "TMF",
    "TLT",
    "EDC",
    "EEM",
]

[5]: # Pull data for each ETF and cache it locally
for fund in fund_list:
    yf_pull_data(
        base_directory=DATA_DIR,
        ticker=fund,
        adjusted=False,
        source="Yahoo_Finance",
        asset_class="Exchange_Traded_Funds",
        excel_export=True,
        pickle_export=True,
        output_confirmation=False,
    )
```

0.6 Create Data Dictionary

Then, create a dictionary to hold the data for each ETF.

```
[6]: # Create an empty dictionary to hold the data for each ETF
fund_data = {}

# Load the data for each ETF, rename the columns to include the ticker, store_
↳ it in the fund_data dictionary
for fund in fund_list:
    data = load_data(
        base_directory=DATA_DIR,
        ticker=fund,
        source="Yahoo_Finance",
        asset_class="Exchange_Traded_Funds",
```

```

        timeframe="Daily",
        file_format="pickle",
    )

    data = data.rename(
        columns={
            "Adj Close": f"{fund}_Adj_Close",
            "Close": f"{fund}_Close",
            "High": f"{fund}_High",
            "Low": f"{fund}_Low",
            "Open": f"{fund}_Open",
            "Volume": f"{fund}_Volume",
        }
    )

    fund_data[fund] = data

```

0.7 Section 2.0: Long-term Empirical behavior of Triple Leveraged Exchange Traded Funds

Section 2.0 of the paper establishes some baseline definitions, specifically CAGR (referenced in the paper as cumulative annualized growth rate, but more commonly as compound annual growth rate), DDD (daily drawdown, daily and maximum), and MDD (monthly drawdown, monthly and maximum).

The following two helper functions will combine and normalize the fund prices, and calculate the summary statistics.

```

[7]: def combine_normalize(
    fund_data: dict,
    lev_etf: str,
    etf: str,
    start_date=None,
    end_date=None,
) -> pd.DataFrame:
    """
    Combine and normalize the data for a leveraged ETF and its corresponding
    ↪non-leveraged ETF.

    Parameters
    -----
    fund_data : dict
        Dictionary containing DataFrames for each fund.
    lev_etf : str
        The ticker symbol for the leveraged ETF.
    etf : str
        The ticker symbol for the non-leveraged ETF.
    start_date : optional,

```

```

        Start date for the data. If None, uses the earliest available date,
        ↪(default is None).
        end_date : optional
            End date for the data. If None, uses the latest available date (default,
            ↪is None).

    Returns
    -----
    pd.DataFrame
        DataFrame with normalized values for both ETFs.
    """

    # Combine into a single DataFrame
    df = fund_data[lev_etf].merge(fund_data[etf], left_index=True,
    ↪right_index=True)

    # Normalize values to start at 1
    df = df / df.iloc[0]

    # Set start and end dates
    if start_date is None:
        start_date = df.index.min()
    if end_date is None:
        end_date = df.index.max()

    df = df.loc[start_date:end_date]

    return df

```

```

[8]: def calculate_returns_stats(
    df: pd.DataFrame,
    lev_etf: str,
    etf: str,
    max_per_ratio_date=None,
) -> pd.DataFrame:
    """
        Calculate and display summary statistics for the returns of a leveraged ETF,
        ↪and its corresponding non-leveraged ETF.

    Parameters
    -----
    df : pd.DataFrame
        DataFrame containing the adjusted close prices of the ETFs.
    lev_etf : str
        Ticker symbol of the leveraged ETF.
    etf : str
        Ticker symbol of the non-leveraged ETF.
    """

```

max_per_ratio_date : optional
Date of the maximum performance ratio. If provided, the CAGR is
 ↪ *calculated through this date, while*
all other statistics use the full date range. If None, the CAGR uses
 ↪ *the full date range (default is None).*

Returns

pd.DataFrame

DataFrame containing the summary statistics for both ETFs, including
 ↪ *the maximum monthly drawdown*
calculated from end-of-month prices only.

"""

```
df[f"{lev_etf}_Return"] = df[f"{lev_etf}_Adj_Close"].pct_change()
df[f"{etf}_Return"] = df[f"{etf}_Adj_Close"].pct_change()
```

```
# Create empty DataFrame for summary statistics
sum_stats = pd.DataFrame()
```

```
df_lev_stats = summary_stats(
    fund_list=[f"{lev_etf}"],
    df=df[[f"{lev_etf}_Return"]].dropna(),
    period="Daily",
    use_calendar_days=False,
    excel_export=False,
    pickle_export=False,
    output_confirmation=False,
)
```

```
sum_stats = pd.concat([sum_stats, df_lev_stats])
```

```
df_etf_stats = summary_stats(
    fund_list=[f"{etf}"],
    df=df[[f"{etf}_Return"]].dropna(),
    period="Daily",
    use_calendar_days=False,
    excel_export=False,
    pickle_export=False,
    output_confirmation=False,
)
```

```
sum_stats = pd.concat([sum_stats, df_etf_stats])
```

Replace the CAGR with the CAGR through the maximum performance ratio
 ↪ *date, if provided*

```
if max_per_ratio_date is not None:
```

```

    for fund in [lev_etf, etf]:
        df_cagr_stats = summary_stats(
            fund_list=[f"{fund}"],
            df=df.loc[:max_per_ratio_date, [f"{fund}_Return"]].dropna(),
            period="Daily",
            use_calendar_days=False,
            excel_export=False,
            pickle_export=False,
            output_confirmation=False,
        )

        sum_stats.loc[f"{fund}_Return", "CAGR (Geometric)"] = df_cagr_stats.
↪loc[
            f"{fund}_Return", "CAGR (Geometric)"
        ]

# Record the end date used for the CAGR
sum_stats["CAGR End Date"] = (
    max_per_ratio_date
    if max_per_ratio_date is not None
    else df.index.max().strftime("%Y-%m-%d")
)

# Calculate the maximum monthly drawdown using end-of-month prices only
for fund in [lev_etf, etf]:
    month_end_prices = df[f"{fund}_Adj_Close"].resample("ME").last()
    sum_stats.loc[f"{fund}_Return", "Max Monthly Drawdown"] = (
        month_end_prices / month_end_prices.cummax() - 1
    ).min()

return sum_stats

```

0.7.1 Figure 1: Comparing the normalized value of TQQQ with its underlying index

First, we will plot TQQQ and QQQ to match figure 1. Note that the data from the paper is normalized to begin at 1, so we will follow that approach as well.

```

[9]: lev_etf = "TQQQ"
    etf = "QQQ"
    end_date = "2020-10-04"

    df = combine_normalize(
        fund_data=fund_data,
        lev_etf=lev_etf,
        etf=etf,
        start_date=None,
        end_date=end_date,
    )

```

```
)

print(f"Table E1: {lev_etf} & {etf} Normalized Data")
display(df)
```

Table E1: TQQQ & QQQ Normalized Data

	TQQQ_Adj_Close	TQQQ_Close	TQQQ_High	TQQQ_Low	TQQQ_Open \
Date					
2010-02-11	1.00	1.00	1.00	1.00	1.00
2010-02-12	1.00	1.00	1.01	1.03	1.03
2010-02-16	1.04	1.04	1.04	1.08	1.09
2010-02-17	1.06	1.06	1.06	1.11	1.12
2010-02-18	1.08	1.08	1.08	1.12	1.13
...	...	...	...	...	...
2020-09-28	75.16	74.97	74.73	77.40	79.44
2020-09-29	74.37	74.19	75.21	78.61	79.56
2020-09-30	75.85	75.66	77.90	79.06	79.01
2020-10-01	79.40	79.20	79.37	82.46	83.91
2020-10-02	72.85	72.67	76.82	76.64	78.72

	TQQQ_Volume	QQQ_Adj_Close	QQQ_Close	QQQ_High	QQQ_Low \
Date					
2010-02-11	1.00	1.00	1.00	1.00	1.00
2010-02-12	2.49	1.00	1.00	1.00	1.01
2010-02-16	2.78	1.01	1.01	1.01	1.03
2010-02-17	5.55	1.02	1.02	1.02	1.04
2010-02-18	11.24	1.03	1.03	1.03	1.04
...	...	...	...	...	...
2020-09-28	39.54	7.08	6.35	6.33	6.35
2020-09-29	32.09	7.04	6.32	6.34	6.44
2020-09-30	49.01	7.09	6.36	6.42	6.45
2020-10-01	40.32	7.20	6.46	6.46	6.54
2020-10-02	57.81	7.00	6.28	6.45	6.39

	QQQ_Open	QQQ_Volume
Date		
2010-02-11	1.00	1.00
2010-02-12	1.01	1.01
2010-02-16	1.02	0.90
2010-02-17	1.03	0.89
2010-02-18	1.04	0.71
...	...	...
2020-09-28	6.43	0.63
2020-09-29	6.43	0.29
2020-09-30	6.42	0.53
2020-10-01	6.55	0.54
2020-10-02	6.41	0.81

[2680 rows x 12 columns]

```
[10]: # Create one empty dataframe for all summary statistics
all_sum_stats = pd.DataFrame()
```

```
[11]: max_per_ratio_date = "2020-09-02"

sum_stats = calculate_returns_stats(
    df=df,
    lev_etf=lev_etf,
    etf=etf,
    max_per_ratio_date=max_per_ratio_date,
)

all_sum_stats = pd.concat([all_sum_stats, sum_stats])

print("Table E2: Leveraged and Non-Leveraged ETF Summary Statistics")
display(all_sum_stats)
```

Table E2: Leveraged and Non-Leveraged ETF Summary Statistics

	Number of Observations	Data Start Date	Data End Date	\
TQQQ_Return	2679	2010-02-12	2020-10-02	
QQQ_Return	2679	2010-02-12	2020-10-02	
	Annual Mean Return (Arithmetic)	Annualized Volatility		
TQQQ_Return	0.57	0.58		
QQQ_Return	0.20	0.20		
	Annualized Sharpe Ratio	CAGR (Geometric)	Daily Max Return	
TQQQ_Return	0.99	0.55	0.27	
QQQ_Return	1.03	0.21	0.08	
	Daily Max Return (Date)	Daily Min Return	Daily Min Return (Date)	\
TQQQ_Return	2020-03-13	-0.34	2020-03-16	
QQQ_Return	2020-03-13	-0.12	2020-03-16	
	Max Drawdown	Peak	Trough	Recovery Date
TQQQ_Return	-0.70	2020-02-19	2020-03-20	2020-07-10
QQQ_Return	-0.29	2020-02-19	2020-03-16	2020-06-03
	Calendar Days to Recovery	MAR Ratio	CAGR	End Date
TQQQ_Return	112	0.71	2020-09-02	
QQQ_Return	79	0.70	2020-09-02	
	Max Monthly Drawdown			
TQQQ_Return	-0.49			
QQQ_Return	-0.17			

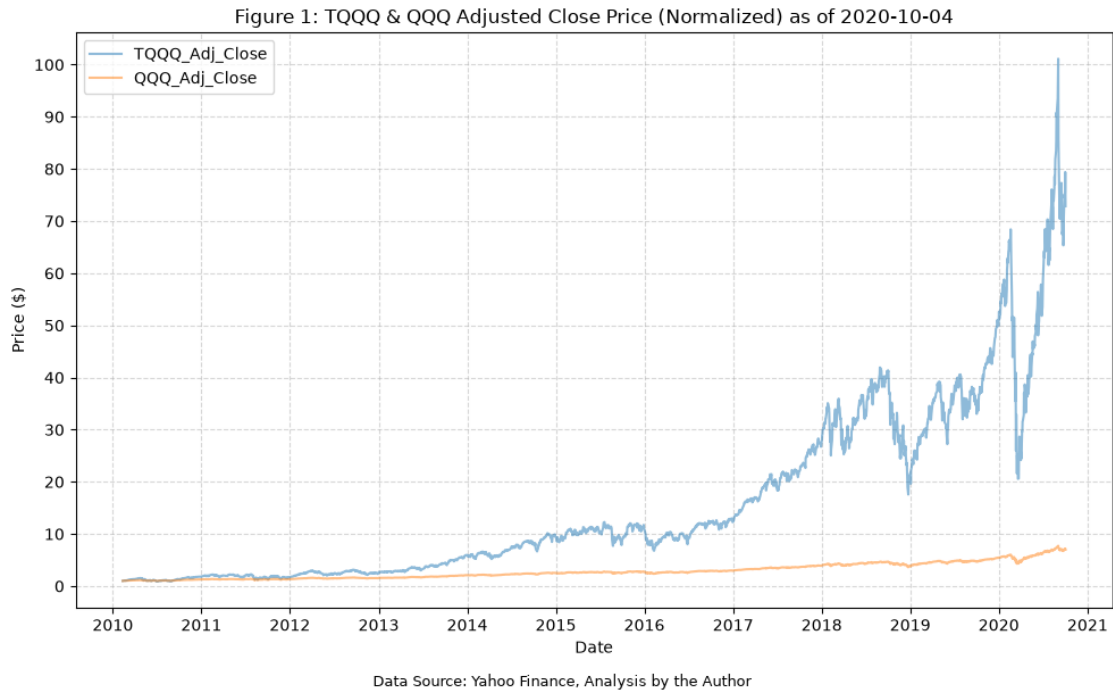
The paper lists a CAGR of 54.4% for TQQQ and 21.2% for QQQ. We find nearly the same values in our calculations, as can be seen above at 55% and 21%.

For the DDD, the paper finds a maximum DDD of 69.2%, and we find 70%. For the MDD, the paper finds a maximum MDD of 49.1%, and we find 49%.

Note that the paper calculates the CAGR and the drawdowns over different date ranges. The CAGR is calculated through the date of the maximum performance ratio, which is the normalized value of the leveraged ETF divided by the normalized value of its underlying ETF. For TQQQ and QQQ, the paper finds a maximum performance ratio of 13.1 (101.12/7.73) on 9/2/2020. The DDD and MDD, on the other hand, are calculated over the full period shown in the plots, which ends on 10/4/2020. We follow the same approach, calculating the CAGR through the maximum performance ratio date and all other statistics through 10/4/2020.

Any slight differences can likely be attributed to our use of 252 trading days per year, vs the paper's use of 250 trading days per year.

```
[12]: plot_time_series(  
    df=df,  
    plot_start_date=None,  
    plot_end_date=None,  
    plot_columns=[f"{lev_etf}_Adj_Close", f"{etf}_Adj_Close"],  
    title=f"Figure 1: {lev_etf} & {etf} Adjusted Close Price (Normalized) as of_  
↪{end_date}",  
    x_label="Date",  
    x_format="Year",  
    x_tick_spacing=1,  
    x_tick_start=None,  
    x_tick_rotation=0,  
    y_label="Price ($)",  
    y_format="Decimal",  
    y_format_decimal_places=0,  
    y_tick_spacing="Auto",  
    y_tick_rotation=0,  
    caption="Data Source: Yahoo Finance, Analysis by the Author",  
)
```



0.7.2 Figure 2: Comparing the normalized value of UPRO with its underlying index
Next, UPRO and SPY.

```
[13]: lev_etf = "UPRO"
      etf = "SPY"
      end_date = "2020-10-04"

      df = combine_normalize(
          fund_data=fund_data,
          lev_etf=lev_etf,
          etf=etf,
          start_date=None,
          end_date=end_date,
      )

      print(f"Table E3: {lev_etf} & {etf} Normalized Data")
      display(df)
```

Table E3: UPRO & SPY Normalized Data

	UPRO_Adj_Close	UPRO_Close	UPRO_High	UPRO_Low	UPRO_Open \
Date					
2009-06-25	1.00	1.00	1.00	1.00	1.00
2009-06-26	0.99	0.99	1.00	1.05	1.06
2009-06-29	1.02	1.02	1.02	1.06	1.07

2009-06-30	1.00	1.00	1.03	1.04	1.10
2009-07-01	1.01	1.01	1.04	1.08	1.08
...	...	...	...	...	...
2020-09-28	23.67	23.06	23.10	24.24	24.46
2020-09-29	23.28	22.68	23.07	24.11	24.63
2020-09-30	23.80	23.19	23.79	24.39	24.44
2020-10-01	24.24	23.62	23.88	24.84	25.44
2020-10-02	23.57	22.97	23.30	23.98	24.10

	UPRO_Volume	SPY_Adj_Close	SPY_Close	SPY_High	SPY_Low	\
Date						
2009-06-25	1.00	1.00	1.00	1.00	1.00	
2009-06-26	5.08	1.00	1.00	1.00	1.02	
2009-06-29	3.37	1.01	1.01	1.01	1.02	
2009-06-30	6.65	1.00	1.00	1.01	1.02	
2009-07-01	4.67	1.00	1.00	1.01	1.03	
...	...	...	...	...	...	
2020-09-28	3.34	4.54	3.63	3.63	3.71	
2020-09-29	2.74	4.52	3.61	3.63	3.70	
2020-09-30	5.30	4.55	3.64	3.67	3.72	
2020-10-01	3.11	4.58	3.66	3.68	3.74	
2020-10-02	4.32	4.54	3.63	3.66	3.70	

	SPY_Open	SPY_Volume
Date		
2009-06-25	1.00	1.00
2009-06-26	1.02	0.60
2009-06-29	1.03	0.60
2009-06-30	1.03	0.82
2009-07-01	1.03	0.62
...	...	...
2020-09-28	3.72	0.23
2020-09-29	3.72	0.18
2020-09-30	3.71	0.37
2020-10-01	3.77	0.32
2020-10-02	3.70	0.32

[2839 rows x 12 columns]

```
[14]: max_per_ratio_date = "2020-02-19"

sum_stats = calculate_returns_stats(
    df=df,
    lev_etf=lev_etf,
    etf=etf,
    max_per_ratio_date=max_per_ratio_date,
)
```

```
all_sum_stats = pd.concat([all_sum_stats, sum_stats])

print("Table E4: Leveraged and Non-Leveraged ETF Summary Statistics")
display(all_sum_stats)
```

Table E4: Leveraged and Non-Leveraged ETF Summary Statistics

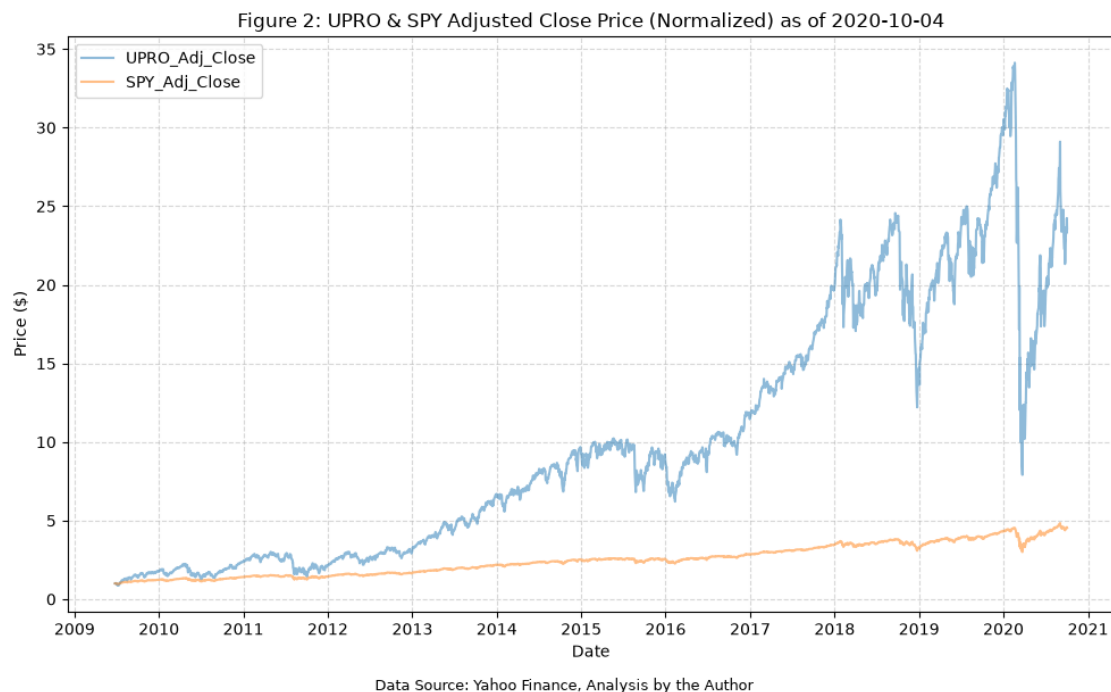
	Number of Observations	Data Start Date	Data End Date	\	
TQQQ_Return	2679	2010-02-12	2020-10-02		
QQQ_Return	2679	2010-02-12	2020-10-02		
UPRO_Return	2838	2009-06-26	2020-10-02		
SPY_Return	2838	2009-06-26	2020-10-02		
	Annual Mean Return (Arithmetic)	Annualized Volatility	\		
TQQQ_Return	0.57	0.58			
QQQ_Return	0.20	0.20			
UPRO_Return	0.42	0.52			
SPY_Return	0.15	0.17			
	Annualized Sharpe Ratio	CAGR (Geometric)	Daily Max Return	\	
TQQQ_Return	0.99	0.55	0.27		
QQQ_Return	1.03	0.21	0.08		
UPRO_Return	0.81	0.39	0.28		
SPY_Return	0.87	0.15	0.09		
	Daily Max Return (Date)	Daily Min Return	Daily Min Return (Date)	\	
TQQQ_Return	2020-03-13	-0.34	2020-03-16		
QQQ_Return	2020-03-13	-0.12	2020-03-16		
UPRO_Return	2020-03-24	-0.35	2020-03-16		
SPY_Return	2020-03-24	-0.11	2020-03-16		
	Max Drawdown	Peak	Trough	Recovery Date	\
TQQQ_Return	-0.70	2020-02-19	2020-03-20	2020-07-10	
QQQ_Return	-0.29	2020-02-19	2020-03-16	2020-06-03	
UPRO_Return	-0.77	2020-02-19	2020-03-23	NaT	
SPY_Return	-0.34	2020-02-19	2020-03-23	2020-08-10	
	Calendar Days to Recovery	MAR Ratio	CAGR	End Date	\
TQQQ_Return	112.00	0.71		2020-09-02	
QQQ_Return	79.00	0.70		2020-09-02	
UPRO_Return	NaN	0.42		2020-02-19	
SPY_Return	140.00	0.43		2020-02-19	
	Max Monthly Drawdown				
TQQQ_Return	-0.49				
QQQ_Return	-0.17				
UPRO_Return	-0.60				

SPY_Return

-0.19

For UPRO and SPY, the paper finds a maximum performance ratio of 7.5 (34.13/4.53) on 2/19/2020, and lists a CAGR through that date of 39.0% for UPRO and 15.1% for SPY. We find 39% for UPRO and 15% for SPY. For the DDD, the paper finds a maximum DDD of 78.8% for UPRO, and we find 77%, so very close there. The paper's maximum MDD for UPRO is 60.4%, which matches our 60%.

```
[15]: plot_time_series(
    df=df,
    plot_start_date=None,
    plot_end_date=None,
    plot_columns=[f"{lev_etf}_Adj_Close", f"{etf}_Adj_Close"],
    title=f"Figure 2: {lev_etf} & {etf} Adjusted Close Price (Normalized) as of_{end_date}",
    x_label="Date",
    x_format="Year",
    x_tick_spacing=1,
    x_tick_start=None,
    x_tick_rotation=0,
    y_label="Price ($)",
    y_format="Decimal",
    y_format_decimal_places=0,
    y_tick_spacing="Auto",
    y_tick_rotation=0,
    caption="Data Source: Yahoo Finance, Analysis by the Author",
)
```



0.7.3 Figure 3: Comparing the normalized value of UDOW with its underlying index

Next, figure #3 plots UDOW and DIA.

```
[16]: lev_etf = "UDOW"
      etf = "DIA"
      end_date = "2020-10-04"

      df = combine_normalize(
          fund_data=fund_data,
          lev_etf=lev_etf,
          etf=etf,
          start_date=None,
          end_date=end_date,
      )

      print(f"Table E5: {lev_etf} & {etf} Normalized Data")
      display(df)
```

Table E5: UDOW & DIA Normalized Data

	UDOW_Adj_Close	UDOW_Close	UDOW_High	UDOW_Low	UDOW_Open \
Date					
2010-02-11	1.00	1.00	1.00	1.00	1.00
2010-02-12	1.00	1.00	1.00	0.99	1.00
2010-02-16	1.06	1.06	1.06	1.04	1.05
2010-02-17	1.09	1.09	1.09	1.08	1.08
2010-02-18	1.11	1.11	1.12	1.09	1.09
...	...	...	...	...	...
2020-09-28	12.03	11.55	11.70	11.41	11.46
2020-09-29	11.82	11.35	11.55	11.22	11.51
2020-09-30	12.24	11.76	12.08	11.43	11.43
2020-10-01	12.32	11.83	12.11	11.63	11.98
2020-10-02	12.13	11.66	11.87	11.25	11.31

	UDOW_Volume	DIA_Adj_Close	DIA_Close	DIA_High	DIA_Low \
Date					
2010-02-11	1.00	1.00	1.00	1.00	1.00
2010-02-12	2.15	1.00	1.00	1.00	1.00
2010-02-16	1.56	1.01	1.01	1.01	1.02
2010-02-17	6.41	1.02	1.02	1.02	1.03
2010-02-18	5.00	1.03	1.03	1.03	1.03
...	...	...	...	...	...
2020-09-28	49.80	3.49	2.72	2.72	2.75
2020-09-29	69.70	3.47	2.70	2.71	2.74
2020-09-30	86.74	3.51	2.73	2.75	2.75

2020-10-01	69.12	3.52	2.74	2.76	2.77
2020-10-02	71.53	3.50	2.72	2.74	2.74

Date	DIA_Open	DIA_Volume
2010-02-11	1.00	1.00
2010-02-12	1.00	1.37
2010-02-16	1.02	0.90
2010-02-17	1.03	0.88
2010-02-18	1.03	0.87
...	...	...
2020-09-28	2.74	0.28
2020-09-29	2.74	0.20
2020-09-30	2.74	0.41
2020-10-01	2.78	0.30
2020-10-02	2.73	0.48

[2680 rows x 12 columns]

```
[17]: max_per_ratio_date = "2020-02-12"

sum_stats = calculate_returns_stats(
    df=df,
    lev_etf=lev_etf,
    etf=etf,
    max_per_ratio_date=max_per_ratio_date,
)

all_sum_stats = pd.concat([all_sum_stats, sum_stats])

print("Table E6: Leveraged and Non-Leveraged ETF Summary Statistics")
display(all_sum_stats)
```

Table E6: Leveraged and Non-Leveraged ETF Summary Statistics

	Number of Observations	Data Start Date	Data End Date \
TQQQ_Return	2679	2010-02-12	2020-10-02
QQQ_Return	2679	2010-02-12	2020-10-02
UPRO_Return	2838	2009-06-26	2020-10-02
SPY_Return	2838	2009-06-26	2020-10-02
UDOW_Return	2679	2010-02-12	2020-10-02
DIA_Return	2679	2010-02-12	2020-10-02

	Annual Mean Return (Arithmetic)	Annualized Volatility \
TQQQ_Return	0.57	0.58
QQQ_Return	0.20	0.20
UPRO_Return	0.42	0.52
SPY_Return	0.15	0.17
UDOW_Return	0.37	0.51

DIA_Return	0.13	0.17
------------	------	------

	Annualized Sharpe Ratio	CAGR (Geometric)	Daily Max Return \
TQQQ_Return	0.99	0.55	0.27
QQQ_Return	1.03	0.21	0.08
UPRO_Return	0.81	0.39	0.28
SPY_Return	0.87	0.15	0.09
UDOW_Return	0.72	0.36	0.33
DIA_Return	0.77	0.14	0.11

	Daily Max Return (Date)	Daily Min Return	Daily Min Return (Date) \
TQQQ_Return	2020-03-13	-0.34	2020-03-16
QQQ_Return	2020-03-13	-0.12	2020-03-16
UPRO_Return	2020-03-24	-0.35	2020-03-16
SPY_Return	2020-03-24	-0.11	2020-03-16
UDOW_Return	2020-03-24	-0.35	2020-03-16
DIA_Return	2020-03-24	-0.13	2020-03-16

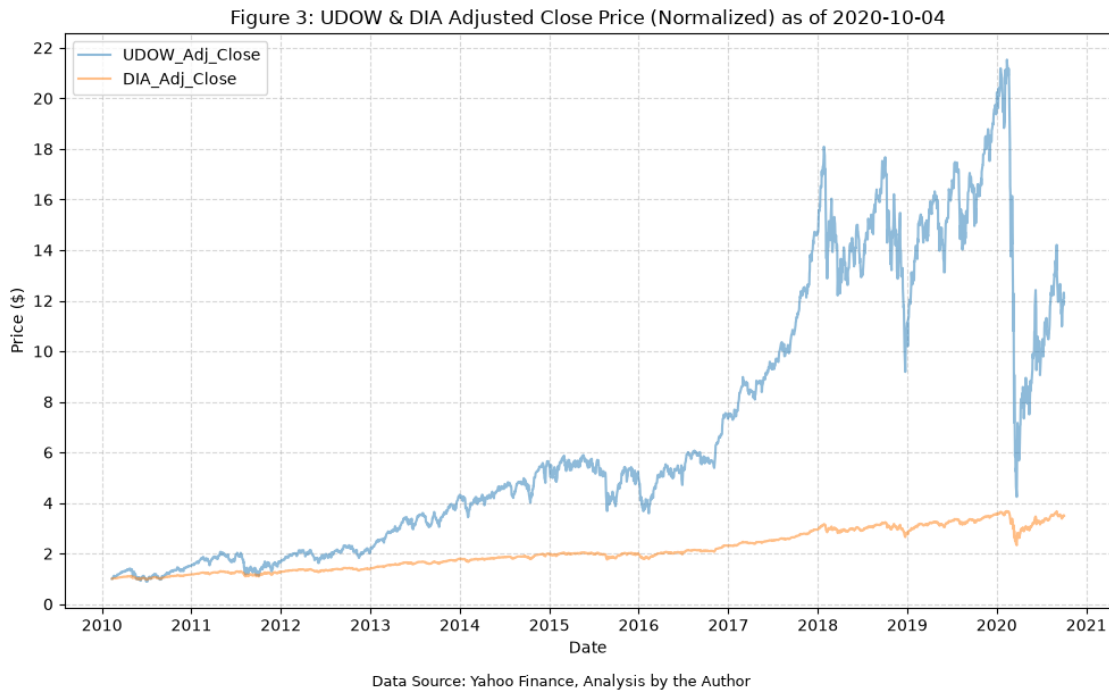
	Max Drawdown	Peak	Trough	Recovery Date \
TQQQ_Return	-0.70	2020-02-19	2020-03-20	2020-07-10
QQQ_Return	-0.29	2020-02-19	2020-03-16	2020-06-03
UPRO_Return	-0.77	2020-02-19	2020-03-23	NaT
SPY_Return	-0.34	2020-02-19	2020-03-23	2020-08-10
UDOW_Return	-0.80	2020-02-12	2020-03-23	NaT
DIA_Return	-0.37	2020-02-12	2020-03-23	NaT

	Calendar Days to Recovery	MAR Ratio	CAGR End Date \
TQQQ_Return	112.00	0.71	2020-09-02
QQQ_Return	79.00	0.70	2020-09-02
UPRO_Return	NaN	0.42	2020-02-19
SPY_Return	140.00	0.43	2020-02-19
UDOW_Return	NaN	0.33	2020-02-12
DIA_Return	NaN	0.34	2020-02-12

	Max Monthly Drawdown
TQQQ_Return	-0.49
QQQ_Return	-0.17
UPRO_Return	-0.60
SPY_Return	-0.19
UDOW_Return	-0.66
DIA_Return	-0.23

For UDOW and DIA, the paper finds a maximum performance ratio of 5.9 (21.62/3.68) on 2/12/2020, and lists a CAGR through that date of 35.7% for UDOW and 13.8% for DIA. Our values are 36% and 14% respectively. The DDD values from the paper at 80.3% and 33.7% are reasonably close to ours at 80% and 37%. For the MDD, the paper lists 66.3% for UDOW, which matches our 66%, and 19.4% for DIA, compared with our 23%. Interestingly, 19.4% is exactly the maximum MDD we find for SPY, so this may be a transcription error in the paper.

```
[18]: plot_time_series(
    df=df,
    plot_start_date=None,
    plot_end_date=None,
    plot_columns=[f"{lev_etf}_Adj_Close", f"{etf}_Adj_Close"],
    title=f"Figure 3: {lev_etf} & {etf} Adjusted Close Price (Normalized) as of_{end_date}",
    x_label="Date",
    x_format="Year",
    x_tick_spacing=1,
    x_tick_start=None,
    x_tick_rotation=0,
    y_label="Price ($)",
    y_format="Decimal",
    y_format_decimal_places=0,
    y_tick_spacing="Auto",
    y_tick_rotation=0,
    caption="Data Source: Yahoo Finance, Analysis by the Author",
)
```



0.7.4 Figure 4: Comparing the normalized value of TNA with its underlying index
 Figure #4 plots TNA and IWM.

```
[19]: lev_etf = "TNA"
      etf = "IWM"
      end_date = "2020-10-04"

      df = combine_normalize(
          fund_data=fund_data,
          lev_etf=lev_etf,
          etf=etf,
          start_date=None,
          end_date=end_date,
      )

      print(f"Table E7: {lev_etf} & {etf} Normalized Data")
      display(df)
```

Table E7: TNA & IWM Normalized Data

	TNA_Adj_Close	TNA_Close	TNA_High	TNA_Low	TNA_Open	TNA_Volume	\
Date							
2008-11-19	1.00	1.00	1.00	1.00	1.00	1.00	
2008-11-20	0.79	0.79	0.78	0.79	0.75	1.82	
2008-11-21	0.91	0.91	0.70	0.71	0.71	1.74	
2008-11-24	1.10	1.10	0.87	0.93	0.76	1.28	
2008-11-25	1.16	1.16	0.89	1.05	0.90	1.04	
...	...	...	...	...	...	...	
2020-09-28	5.99	5.08	3.87	4.94	3.83	1.13	
2020-09-29	5.90	5.00	3.86	4.91	3.94	1.34	
2020-09-30	5.96	5.05	3.99	4.96	3.94	1.39	
2020-10-01	6.23	5.28	4.01	5.06	4.01	1.15	
2020-10-02	6.32	5.36	4.11	5.00	3.90	1.52	
	IWM_Adj_Close	IWM_Close	IWM_High	IWM_Low	IWM_Open	IWM_Volume	
Date							
2008-11-19	1.00	1.00	1.00	1.00	1.00	1.00	
2008-11-20	0.93	0.93	0.93	0.93	0.92	1.46	
2008-11-21	0.98	0.98	0.90	0.90	0.88	1.67	
2008-11-24	1.05	1.05	0.98	0.99	0.92	1.01	
2008-11-25	1.07	1.07	0.98	1.03	0.99	0.83	
...	...	...	...	...	...	...	
2020-09-28	4.32	3.63	3.33	3.55	3.31	0.18	
2020-09-29	4.30	3.61	3.33	3.59	3.35	0.19	
2020-09-30	4.31	3.62	3.37	3.60	3.35	0.30	
2020-10-01	4.38	3.68	3.37	3.62	3.37	0.27	
2020-10-02	4.40	3.70	3.40	3.61	3.34	0.31	

[2988 rows x 12 columns]

```
[20]: max_per_ratio_date = "2018-08-29"

sum_stats = calculate_returns_stats(
    df=df,
    lev_etf=lev_etf,
    etf=etf,
    max_per_ratio_date=max_per_ratio_date,
)

all_sum_stats = pd.concat([all_sum_stats, sum_stats])

print("Table E8: Leveraged and Non-Leveraged ETF Summary Statistics")
display(all_sum_stats)
```

Table E8: Leveraged and Non-Leveraged ETF Summary Statistics

	Number of Observations	Data Start Date	Data End Date	\
TQQQ_Return	2679	2010-02-12	2020-10-02	
QQQ_Return	2679	2010-02-12	2020-10-02	
UPRO_Return	2838	2009-06-26	2020-10-02	
SPY_Return	2838	2009-06-26	2020-10-02	
UDOW_Return	2679	2010-02-12	2020-10-02	
DIA_Return	2679	2010-02-12	2020-10-02	
TNA_Return	2987	2008-11-20	2020-10-02	
IWM_Return	2987	2008-11-20	2020-10-02	

	Annual Mean Return (Arithmetic)	Annualized Volatility	\
TQQQ_Return	0.57	0.58	
QQQ_Return	0.20	0.20	
UPRO_Return	0.42	0.52	
SPY_Return	0.15	0.17	
UDOW_Return	0.37	0.51	
DIA_Return	0.13	0.17	
TNA_Return	0.42	0.72	
IWM_Return	0.15	0.24	

	Annualized Sharpe Ratio	CAGR (Geometric)	Daily Max Return	\
TQQQ_Return	0.99	0.55	0.27	
QQQ_Return	1.03	0.21	0.08	
UPRO_Return	0.81	0.39	0.28	
SPY_Return	0.87	0.15	0.09	
UDOW_Return	0.72	0.36	0.33	
DIA_Return	0.77	0.14	0.11	
TNA_Return	0.58	0.35	0.27	
IWM_Return	0.64	0.17	0.09	

	Daily Max Return (Date)	Daily Min Return	Daily Min Return (Date)	\
TQQQ_Return	2020-03-13	-0.34	2020-03-16	

QQQ_Return	2020-03-13	-0.12	2020-03-16
UPRO_Return	2020-03-24	-0.35	2020-03-16
SPY_Return	2020-03-24	-0.11	2020-03-16
UDOW_Return	2020-03-24	-0.35	2020-03-16
DIA_Return	2020-03-24	-0.13	2020-03-16
TNA_Return	2020-03-24	-0.37	2020-03-16
IWM_Return	2020-03-24	-0.13	2020-03-16

	Max Drawdown	Peak	Trough	Recovery Date \
TQQQ_Return	-0.70	2020-02-19	2020-03-20	2020-07-10
QQQ_Return	-0.29	2020-02-19	2020-03-16	2020-06-03
UPRO_Return	-0.77	2020-02-19	2020-03-23	NaT
SPY_Return	-0.34	2020-02-19	2020-03-23	2020-08-10
UDOW_Return	-0.80	2020-02-12	2020-03-23	NaT
DIA_Return	-0.37	2020-02-12	2020-03-23	NaT
TNA_Return	-0.88	2018-08-31	2020-03-23	NaT
IWM_Return	-0.41	2018-08-31	2020-03-23	NaT

	Calendar Days to Recovery	MAR Ratio	CAGR	End Date \
TQQQ_Return	112.00	0.71		2020-09-02
QQQ_Return	79.00	0.70		2020-09-02
UPRO_Return	NaN	0.42		2020-02-19
SPY_Return	140.00	0.43		2020-02-19
UDOW_Return	NaN	0.33		2020-02-12
DIA_Return	NaN	0.34		2020-02-12
TNA_Return	NaN	0.19		2018-08-29
IWM_Return	NaN	0.32		2018-08-29

	Max Monthly Drawdown
TQQQ_Return	-0.49
QQQ_Return	-0.17
UPRO_Return	-0.60
SPY_Return	-0.19
UDOW_Return	-0.66
DIA_Return	-0.23
TNA_Return	-0.83
IWM_Return	-0.32

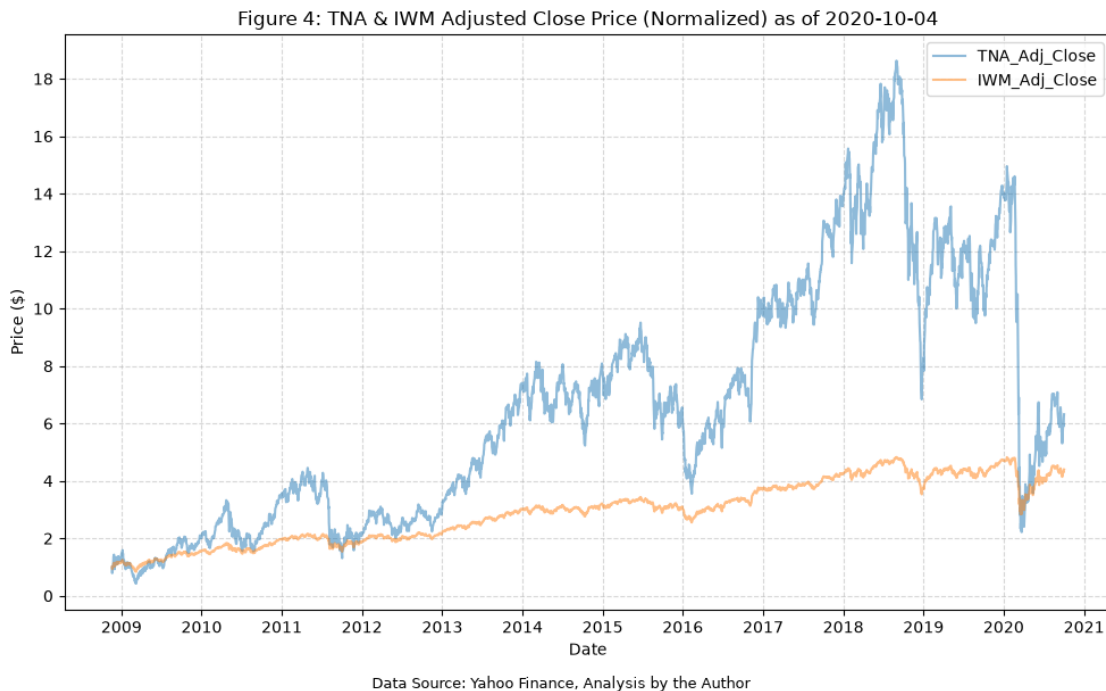
For TNA and IWM, the paper finds a maximum performance ratio of 3.8 (18.48/4.81) on 8/29/2018, well before the end of the period, and lists a CAGR through that date of 34.5% for TNA and 17.3% for IWM. Our values are 35% and 17% respectively. The DDD values from the paper at 88.1% and 41.1% are very close to ours at 88% and 41%. For the MDD, the paper lists 68.0% for TNA and 32.3% for IWM. Our IWM value of 32% matches, but our TNA value is 83%, measured from the end-of-month peak in August 2018 to the end-of-month trough in March 2020. We were not able to reproduce the paper's value for TNA.

```
[21]: plot_time_series(
      df=df,
```

```

plot_start_date=None,
plot_end_date=None,
plot_columns=[f"{lev_etf}_Adj_Close", f"{etf}_Adj_Close"],
title=f"Figure 4: {lev_etf} & {etf} Adjusted Close Price (Normalized) as of_{end_date}",
x_label="Date",
x_format="Year",
x_tick_spacing=1,
x_tick_start=None,
x_tick_rotation=0,
y_label="Price ($)",
y_format="Decimal",
y_format_decimal_places=0,
y_tick_spacing="Auto",
y_tick_rotation=0,
caption="Data Source: Yahoo Finance, Analysis by the Author",
)

```



An interesting observation from the paper, from page 6:

“Note that, in this case, the normalized value of TNA at the end of the period actually briefly diminishes below that of IWM as a result of the financial crisis triggered by the COVID-19 pandemic, another illustration of the extreme volatility of the TLETFs compared with their underlying.”

0.7.5 Figure 5: Comparing the normalized value of TMF with its underlying index

Next, figure #5 plots TMF and TLT.

```
[22]: lev_etf = "TMF"
      etf = "TLT"
      end_date = "2020-10-04"

      df = combine_normalize(
        fund_data=fund_data,
        lev_etf=lev_etf,
        etf=etf,
        start_date=None,
        end_date=end_date,
      )

      print(f"Table E9: {lev_etf} & {etf} Normalized Data")
      display(df)
```

Table E9: TMF & TLT Normalized Data

	TMF_Adj_Close	TMF_Close	TMF_High	TMF_Low	TMF_Open	TMF_Volume	\
Date							
2009-04-16	1.00	1.00	1.00	1.00	1.00	1.00	
2009-04-17	0.96	0.96	0.97	0.95	0.98	2.64	
2009-04-20	1.01	1.01	1.00	0.99	0.98	0.60	
2009-04-21	0.98	0.98	1.03	0.99	1.03	1.37	
2009-04-22	0.95	0.95	0.97	0.94	0.98	2.22	
...	...	...	...	...	...	...	
2020-09-28	3.51	2.82	2.78	2.82	2.82	9.83	
2020-09-29	3.52	2.83	2.80	2.82	2.81	15.62	
2020-09-30	3.43	2.75	2.74	2.72	2.77	22.09	
2020-10-01	3.45	2.77	2.73	2.71	2.70	22.16	
2020-10-02	3.41	2.73	2.73	2.71	2.76	18.32	
	TLT_Adj_Close	TLT_Close	TLT_High	TLT_Low	TLT_Open	TLT_Volume	
Date							
2009-04-16	1.00	1.00	1.00	1.00	1.00	1.00	
2009-04-17	0.99	0.99	0.99	0.99	1.00	2.45	
2009-04-20	1.00	1.00	1.00	1.00	1.00	1.62	
2009-04-21	0.99	0.99	1.01	0.99	1.01	0.86	
2009-04-22	0.99	0.99	0.99	0.98	1.00	2.16	
...	...	...	...	...	...	...	
2020-09-28	2.22	1.60	1.59	1.60	1.60	4.23	
2020-09-29	2.22	1.60	1.60	1.60	1.60	9.76	
2020-09-30	2.20	1.59	1.59	1.58	1.59	12.29	
2020-10-01	2.21	1.59	1.58	1.58	1.58	10.92	
2020-10-02	2.20	1.58	1.58	1.58	1.59	8.36	

[2888 rows x 12 columns]

```
[23]: max_per_ratio_date = "2020-03-09"

sum_stats = calculate_returns_stats(
    df=df,
    lev_etf=lev_etf,
    etf=etf,
    max_per_ratio_date=max_per_ratio_date,
)

all_sum_stats = pd.concat([all_sum_stats, sum_stats])

print("Table E10: Leveraged and Non-Leveraged ETF Summary Statistics")
display(all_sum_stats)
```

Table E10: Leveraged and Non-Leveraged ETF Summary Statistics

	Number of Observations	Data Start Date	Data End Date	\
TQQQ_Return	2679	2010-02-12	2020-10-02	
QQQ_Return	2679	2010-02-12	2020-10-02	
UPRO_Return	2838	2009-06-26	2020-10-02	
SPY_Return	2838	2009-06-26	2020-10-02	
UDOW_Return	2679	2010-02-12	2020-10-02	
DIA_Return	2679	2010-02-12	2020-10-02	
TNA_Return	2987	2008-11-20	2020-10-02	
IWM_Return	2987	2008-11-20	2020-10-02	
TMF_Return	2887	2009-04-17	2020-10-02	
TLT_Return	2887	2009-04-17	2020-10-02	
	Annual Mean Return (Arithmetic)	Annualized Volatility		\
TQQQ_Return	0.57	0.58		
QQQ_Return	0.20	0.20		
UPRO_Return	0.42	0.52		
SPY_Return	0.15	0.17		
UDOW_Return	0.37	0.51		
DIA_Return	0.13	0.17		
TNA_Return	0.42	0.72		
IWM_Return	0.15	0.24		
TMF_Return	0.21	0.44		
TLT_Return	0.08	0.15		
	Annualized Sharpe Ratio	CAGR (Geometric)	Daily Max Return	\
TQQQ_Return	0.99	0.55	0.27	
QQQ_Return	1.03	0.21	0.08	
UPRO_Return	0.81	0.39	0.28	
SPY_Return	0.87	0.15	0.09	
UDOW_Return	0.72	0.36	0.33	
DIA_Return	0.77	0.14	0.11	

TNA_Return	0.58	0.35	0.27
IWM_Return	0.64	0.17	0.09
TMF_Return	0.46	0.15	0.20
TLT_Return	0.54	0.08	0.08

	Daily Max Return (Date)	Daily Min Return	Daily Min Return (Date)	\
TQQQ_Return	2020-03-13	-0.34	2020-03-16	
QQQ_Return	2020-03-13	-0.12	2020-03-16	
UPRO_Return	2020-03-24	-0.35	2020-03-16	
SPY_Return	2020-03-24	-0.11	2020-03-16	
UDOW_Return	2020-03-24	-0.35	2020-03-16	
DIA_Return	2020-03-24	-0.13	2020-03-16	
TNA_Return	2020-03-24	-0.37	2020-03-16	
IWM_Return	2020-03-24	-0.13	2020-03-16	
TMF_Return	2020-03-20	-0.18	2020-03-18	
TLT_Return	2020-03-20	-0.07	2020-03-17	

	Max Drawdown	Peak	Trough	Recovery Date	\
TQQQ_Return	-0.70	2020-02-19	2020-03-20	2020-07-10	
QQQ_Return	-0.29	2020-02-19	2020-03-16	2020-06-03	
UPRO_Return	-0.77	2020-02-19	2020-03-23	NaT	
SPY_Return	-0.34	2020-02-19	2020-03-23	2020-08-10	
UDOW_Return	-0.80	2020-02-12	2020-03-23	NaT	
DIA_Return	-0.37	2020-02-12	2020-03-23	NaT	
TNA_Return	-0.88	2018-08-31	2020-03-23	NaT	
IWM_Return	-0.41	2018-08-31	2020-03-23	NaT	
TMF_Return	-0.53	2012-07-25	2013-08-21	2015-01-05	
TLT_Return	-0.21	2012-07-25	2013-08-21	2014-12-10	

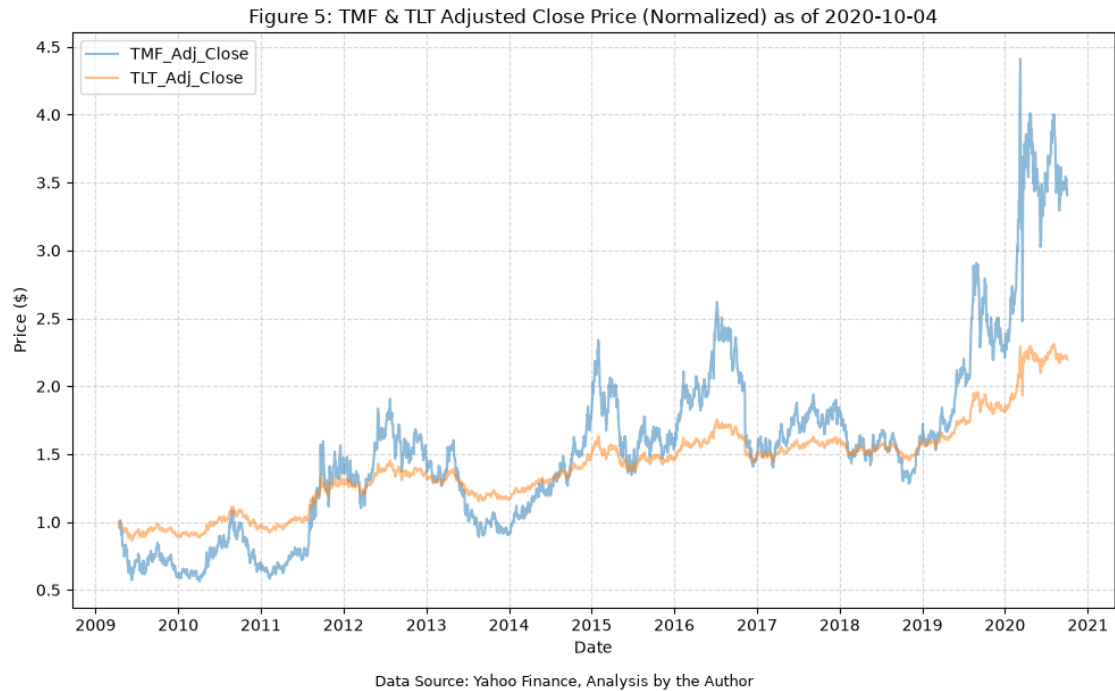
	Calendar Days to Recovery	MAR Ratio	CAGR	End Date	\
TQQQ_Return	112.00	0.71		2020-09-02	
QQQ_Return	79.00	0.70		2020-09-02	
UPRO_Return	NaN	0.42		2020-02-19	
SPY_Return	140.00	0.43		2020-02-19	
UDOW_Return	NaN	0.33		2020-02-12	
DIA_Return	NaN	0.34		2020-02-12	
TNA_Return	NaN	0.19		2018-08-29	
IWM_Return	NaN	0.32		2018-08-29	
TMF_Return	502.00	0.21		2020-03-09	
TLT_Return	476.00	0.34		2020-03-09	

	Max Monthly Drawdown
TQQQ_Return	-0.49
QQQ_Return	-0.17
UPRO_Return	-0.60
SPY_Return	-0.19
UDOW_Return	-0.66
DIA_Return	-0.23

TNA_Return	-0.83
IWM_Return	-0.32
TMF_Return	-0.50
TLT_Return	-0.18

For TMF and TLT, the paper finds a maximum performance ratio of 1.9 (4.412/2.294) on 3/9/2020, and reports a CAGR through that date of 14.5% for TMF and 7.9% for TLT. Our values are 15% and 8% respectively. For the drawdowns, the paper reports 53.3% for TMF and 20.7% for TLT, whereas our values are 53% and 21%. For the MDD, the paper reports 49.7% for TMF and 18.2% for TLT, and we find 50% and 18%. So again, very close for both the CAGR and the drawdown metrics.

```
[24]: plot_time_series(
    df=df,
    plot_start_date=None,
    plot_end_date=None,
    plot_columns=[f"{lev_etf}_Adj_Close", f"{etf}_Adj_Close"],
    title=f"Figure 5: {lev_etf} & {etf} Adjusted Close Price (Normalized) as of_{end_date}",
    x_label="Date",
    x_format="Year",
    x_tick_spacing=1,
    x_tick_start=None,
    x_tick_rotation=0,
    y_label="Price ($)",
    y_format="Decimal",
    y_format_decimal_places=1,
    y_tick_spacing="Auto",
    y_tick_rotation=0,
    caption="Data Source: Yahoo Finance, Analysis by the Author",
)
```



0.7.6 Figure 7: Comparing the normalized value of EDC with its underlying index

As noted above, we will omit DZK and EFA, so we will jump to figure #7 which plots EDC and EEM.

```
[25]: lev_etf = "EDC"
      etf = "EEM"
      end_date = "2020-10-04"

      df = combine_normalize(
          fund_data=fund_data,
          lev_etf=lev_etf,
          etf=etf,
          start_date=None,
          end_date=end_date,
      )

      print(f"Table E11: {lev_etf} & {etf} Normalized Data")
      display(df)
```

Table E11: EDC & EEM Normalized Data

Date	EDC_Adj_Close	EDC_Close	EDC_High	EDC_Low	EDC_Open	EDC_Volume \
2008-12-30	1.00	1.00	1.00	1.00	1.00	1.00
2008-12-31	1.05	1.05	1.06	1.01	1.04	0.56

2009-01-02	1.21	1.21	1.22	1.13	1.13	0.76
2009-01-05	1.24	1.24	1.27	1.23	1.29	1.19
2009-01-06	1.34	1.34	1.36	1.35	1.37	2.62
...	...	...	...	...	...	...
2020-09-28	0.86	0.67	0.68	0.69	0.71	4.07
2020-09-29	0.86	0.67	0.67	0.70	0.70	2.88
2020-09-30	0.89	0.70	0.71	0.72	0.72	7.45
2020-10-01	0.92	0.72	0.72	0.74	0.75	6.49
2020-10-02	0.89	0.70	0.71	0.72	0.72	6.74

	EEM_Adj_Close	EEM_Close	EEM_High	EEM_Low	EEM_Open	EEM_Volume
Date						
2008-12-30	1.00	1.00	1.00	1.00	1.00	1.00
2008-12-31	1.01	1.01	1.01	1.01	1.01	0.89
2009-01-02	1.06	1.06	1.07	1.04	1.04	1.39
2009-01-05	1.08	1.08	1.08	1.07	1.07	1.29
2009-01-06	1.10	1.10	1.10	1.11	1.11	1.36
...	...	...	...	...	...	...
2020-09-28	2.22	1.76	1.76	1.79	1.80	0.91
2020-09-29	2.22	1.76	1.76	1.79	1.79	0.51
2020-09-30	2.25	1.79	1.79	1.81	1.81	0.97
2020-10-01	2.28	1.81	1.80	1.83	1.83	1.03
2020-10-02	2.25	1.79	1.80	1.81	1.81	0.65

[2961 rows x 12 columns]

```
[26]: max_per_ratio_date = None

sum_stats = calculate_returns_stats(
    df=df,
    lev_etf=lev_etf,
    etf=etf,
    max_per_ratio_date=max_per_ratio_date,
)

all_sum_stats = pd.concat([all_sum_stats, sum_stats])

print("Table E12: Leveraged and Non-Leveraged ETF Summary Statistics")
display(all_sum_stats)
```

Table E12: Leveraged and Non-Leveraged ETF Summary Statistics

	Number of Observations	Data Start Date	Data End Date \
TQQQ_Return	2679	2010-02-12	2020-10-02
QQQ_Return	2679	2010-02-12	2020-10-02
UPRO_Return	2838	2009-06-26	2020-10-02
SPY_Return	2838	2009-06-26	2020-10-02
UDOW_Return	2679	2010-02-12	2020-10-02
DIA_Return	2679	2010-02-12	2020-10-02

TNA_Return	2987	2008-11-20	2020-10-02
IWM_Return	2987	2008-11-20	2020-10-02
TMF_Return	2887	2009-04-17	2020-10-02
TLT_Return	2887	2009-04-17	2020-10-02
EDC_Return	2960	2008-12-31	2020-10-02
EEM_Return	2960	2008-12-31	2020-10-02

	Annual Mean Return (Arithmetic)	Annualized Volatility \
TQQQ_Return	0.57	0.58
QQQ_Return	0.20	0.20
UPRO_Return	0.42	0.52
SPY_Return	0.15	0.17
UDOW_Return	0.37	0.51
DIA_Return	0.13	0.17
TNA_Return	0.42	0.72
IWM_Return	0.15	0.24
TMF_Return	0.21	0.44
TLT_Return	0.08	0.15
EDC_Return	0.25	0.72
EEM_Return	0.10	0.25

	Annualized Sharpe Ratio	CAGR (Geometric)	Daily Max Return \
TQQQ_Return	0.99	0.55	0.27
QQQ_Return	1.03	0.21	0.08
UPRO_Return	0.81	0.39	0.28
SPY_Return	0.87	0.15	0.09
UDOW_Return	0.72	0.36	0.33
DIA_Return	0.77	0.14	0.11
TNA_Return	0.58	0.35	0.27
IWM_Return	0.64	0.17	0.09
TMF_Return	0.46	0.15	0.20
TLT_Return	0.54	0.08	0.08
EDC_Return	0.35	-0.01	0.28
EEM_Return	0.40	0.07	0.09

	Daily Max Return (Date)	Daily Min Return	Daily Min Return (Date) \
TQQQ_Return	2020-03-13	-0.34	2020-03-16
QQQ_Return	2020-03-13	-0.12	2020-03-16
UPRO_Return	2020-03-24	-0.35	2020-03-16
SPY_Return	2020-03-24	-0.11	2020-03-16
UDOW_Return	2020-03-24	-0.35	2020-03-16
DIA_Return	2020-03-24	-0.13	2020-03-16
TNA_Return	2020-03-24	-0.37	2020-03-16
IWM_Return	2020-03-24	-0.13	2020-03-16
TMF_Return	2020-03-20	-0.18	2020-03-18
TLT_Return	2020-03-20	-0.07	2020-03-17
EDC_Return	2009-03-23	-0.35	2020-03-16
EEM_Return	2009-03-23	-0.12	2020-03-16

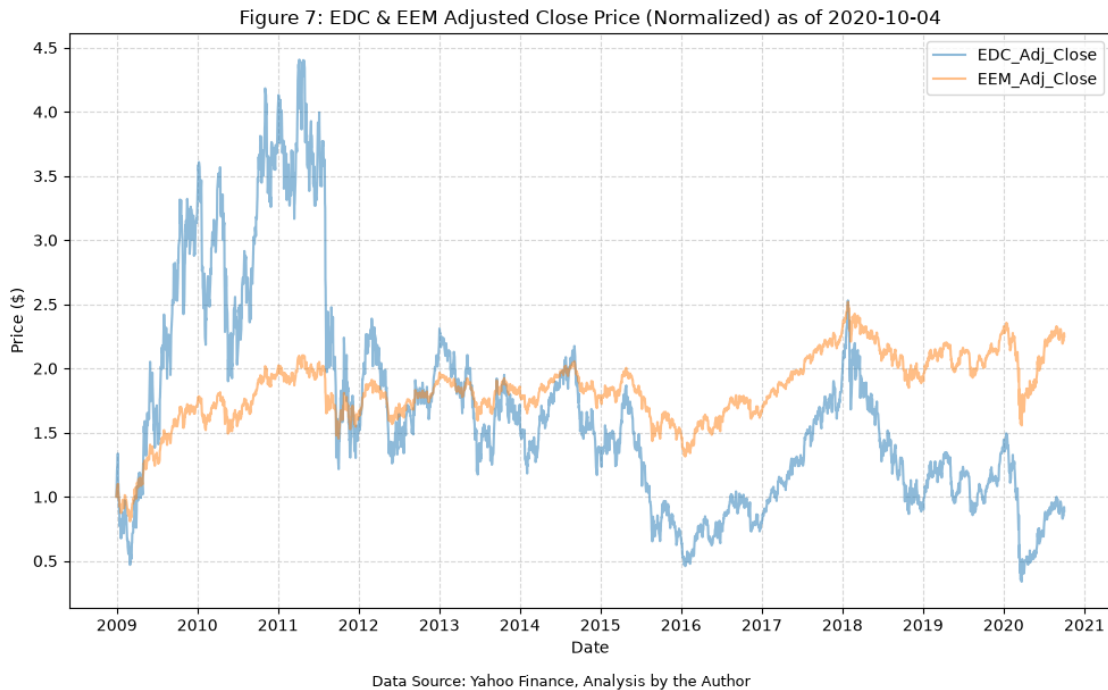
	Max Drawdown	Peak	Trough	Recovery Date	\
TQQQ_Return	-0.70	2020-02-19	2020-03-20	2020-07-10	
QQQ_Return	-0.29	2020-02-19	2020-03-16	2020-06-03	
UPRO_Return	-0.77	2020-02-19	2020-03-23	NaT	
SPY_Return	-0.34	2020-02-19	2020-03-23	2020-08-10	
UDOW_Return	-0.80	2020-02-12	2020-03-23	NaT	
DIA_Return	-0.37	2020-02-12	2020-03-23	NaT	
TNA_Return	-0.88	2018-08-31	2020-03-23	NaT	
IWM_Return	-0.41	2018-08-31	2020-03-23	NaT	
TMF_Return	-0.53	2012-07-25	2013-08-21	2015-01-05	
TLT_Return	-0.21	2012-07-25	2013-08-21	2014-12-10	
EDC_Return	-0.92	2011-04-08	2020-03-23	NaT	
EEM_Return	-0.38	2018-01-26	2020-03-23	NaT	

	Calendar Days to Recovery	MAR Ratio	CAGR	End Date	\
TQQQ_Return	112.00	0.71		2020-09-02	
QQQ_Return	79.00	0.70		2020-09-02	
UPRO_Return	NaN	0.42		2020-02-19	
SPY_Return	140.00	0.43		2020-02-19	
UDOW_Return	NaN	0.33		2020-02-12	
DIA_Return	NaN	0.34		2020-02-12	
TNA_Return	NaN	0.19		2018-08-29	
IWM_Return	NaN	0.32		2018-08-29	
TMF_Return	502.00	0.21		2020-03-09	
TLT_Return	476.00	0.34		2020-03-09	
EDC_Return	NaN	-0.01		2020-10-02	
EEM_Return	NaN	0.19		2020-10-02	

	Max Monthly Drawdown
TQQQ_Return	-0.49
QQQ_Return	-0.17
UPRO_Return	-0.60
SPY_Return	-0.19
UDOW_Return	-0.66
DIA_Return	-0.23
TNA_Return	-0.83
IWM_Return	-0.32
TMF_Return	-0.50
TLT_Return	-0.18
EDC_Return	-0.90
EEM_Return	-0.33

Finally, the paper does not provide a maximum performance ratio or any CAGR values for EDC or EEM, so we calculate the CAGR over the full period, and find -1% for EDC and 7% for EEM. For the drawdowns, we observed 92% for EDC and 38% for EEM, which is in line with the values of 92.3% and 38.2% that were provided in the paper. The MDD values from the paper of 89.6% and 32.7% also match ours at 90% and 33%.

```
[27]: plot_time_series(
    df=df,
    plot_start_date=None,
    plot_end_date=None,
    plot_columns=[f"{lev_etf}_Adj_Close", f"{etf}_Adj_Close"],
    title=f"Figure 7: {lev_etf} & {etf} Adjusted Close Price (Normalized) as of_{end_date}",
    x_label="Date",
    x_format="Year",
    x_tick_spacing=1,
    x_tick_start=None,
    x_tick_rotation=0,
    y_label="Price ($)",
    y_format="Decimal",
    y_format_decimal_places=1,
    y_tick_spacing="Auto",
    y_tick_rotation=0,
    caption="Data Source: Yahoo Finance, Analysis by the Author",
)
```



0.8 Section 3.0: The Risk Mitigation Strategy

In section 3.0, the paper talks about using a portfolio composed of a combination of 2 TLETFs, rebalanced periodically to maintain a specific asset allocation. The section starts with the following claim:

“As mentioned earlier, it is easy to show that QQQ, SPY, and DIA are generally negatively correlated with TLT and the same can be said about their corresponding TLETFs.”

Before we move on to the portfolio and strategy, let’s first look at the correlations between the above mentioned ETFs and the corresponding TLETFs.

```
[28]: corr_data = pd.DataFrame()

for fund in ["QQQ", "SPY", "DIA", "TLT"]:
    df = fund_data[fund]
    df[f"{fund}_Return"] = df[f"{fund}_Adj_Close"].pct_change()
    corr_data = pd.concat([corr_data, df[[f"{fund}_Return"]]], axis=1)

corr_data = corr_data.dropna()
corr_data = corr_data[corr_data.index >= "2010-02-11"]
corr_data = corr_data[corr_data.index <= "2020-10-04"]

print("Table E13: QQQ, SPY, DIA, and TLT Daily Returns")
display(corr_data)

print("Table E14: Correlation Matrix of QQQ, SPY, DIA, and TLT Daily Returns")
display(corr_data.corr())
```

Table E13: QQQ, SPY, DIA, and TLT Daily Returns

	QQQ_Return	SPY_Return	DIA_Return	TLT_Return
Date				
2010-02-11	0.02	0.01	0.01	-0.01
2010-02-12	0.00	-0.00	-0.00	0.00
2010-02-16	0.01	0.02	0.02	0.00
2010-02-17	0.01	0.00	0.01	-0.01
2010-02-18	0.01	0.01	0.01	-0.00
...	...	...	...	...
2020-09-28	0.02	0.02	0.02	-0.00
2020-09-29	-0.00	-0.01	-0.01	0.00
2020-09-30	0.01	0.01	0.01	-0.01
2020-10-01	0.02	0.01	0.00	0.00
2020-10-02	-0.03	-0.01	-0.00	-0.00

[2680 rows x 4 columns]

Table E14: Correlation Matrix of QQQ, SPY, DIA, and TLT Daily Returns

	QQQ_Return	SPY_Return	DIA_Return	TLT_Return
QQQ_Return	1.00	0.93	0.87	-0.42
SPY_Return	0.93	1.00	0.97	-0.48
DIA_Return	0.87	0.97	1.00	-0.47
TLT_Return	-0.42	-0.48	-0.47	1.00

The above table confirms the conjecture from the paper. The correlations amongst the equity and

bond funds are in fact negative.

Next, we will confirm the correlations for the leveraged ETFs.

```
[29]: lev_corr_data = pd.DataFrame()

for fund in ["TQQQ", "UPRO", "UDOW", "TMF"]:
    df = fund_data[fund]
    df[f"{fund}_Return"] = df[f"{fund}_Adj_Close"].pct_change()
    lev_corr_data = pd.concat([lev_corr_data, df[[f"{fund}_Return"]]], axis=1)

lev_corr_data = lev_corr_data.dropna()
lev_corr_data = lev_corr_data[lev_corr_data.index <= "2020-10-04"]

print("Table E15: TQQQ, UPRO, UDOW, and TMF Daily Returns")
display(lev_corr_data)

print("Table E16: Correlation Matrix of TQQQ, UPRO, UDOW, and TMF Daily Returns")
display(lev_corr_data.corr())
```

Table E15: TQQQ, UPRO, UDOW, and TMF Daily Returns

	TQQQ_Return	UPRO_Return	UDOW_Return	TMF_Return
Date				
2010-02-12	0.00	-0.00	-0.00	0.01
2010-02-16	0.04	0.05	0.07	0.01
2010-02-17	0.02	0.01	0.02	-0.03
2010-02-18	0.02	0.02	0.03	-0.02
2010-02-19	0.00	0.01	0.00	0.02
...	...	...	...	...
2020-09-28	0.06	0.05	0.05	-0.01
2020-09-29	-0.01	-0.02	-0.02	0.00
2020-09-30	0.02	0.02	0.04	-0.03
2020-10-01	0.05	0.02	0.01	0.01
2020-10-02	-0.08	-0.03	-0.01	-0.01

[2679 rows x 4 columns]

Table E16: Correlation Matrix of TQQQ, UPRO, UDOW, and TMF Daily Returns

	TQQQ_Return	UPRO_Return	UDOW_Return	TMF_Return
TQQQ_Return	1.00	0.93	0.87	-0.41
UPRO_Return	0.93	1.00	0.97	-0.48
UDOW_Return	0.87	0.97	1.00	-0.46
TMF_Return	-0.41	-0.48	-0.46	1.00

Here we have the same finding. The correlations between the leveraged equity ETFs and the leveraged bond ETF are also negative. They are also nearly identical to the correlations for the underlying non-leveraged ETFs over the same period. This is expected, since the daily return of

a leveraged ETF is approximately 3 times the daily return of its underlying index, and scaling a return series does not change its correlation.

Next, we'll look at the strategy. From the paper:

Strategies that exploit this feature have been shown to exceed the performance of both that of both TLT and the equity ETFs and with reduced risk^{6,7}. Here we focus on the triple leveraged versions of QQQ (TQQQ) and TLT (TMF) and propose simply to purchase a portfolio consisting of equal dollar amounts of both TLETFs and attempt to maintain this percentage by rebalancing on the last day of the k th month, where $k=1,2,3,\dots$

We'll use the following helper function to execute the strategy.

```
[30]: def rebalance_strategy(
    fund_data: dict,
    fund_1: str,
    fund_2: str,
    fund_1_weight: float = 0.5,
    rebalance_months: int = 2,
    transaction_fee: float = 0.0025,
    start_date=None,
    end_date=None,
) -> pd.DataFrame:
    """
    Simulate a two-fund portfolio that is periodically rebalanced back to a
    ↪target allocation.

    Parameters
    -----
    fund_data : dict
        Dictionary containing DataFrames for each fund.
    fund_1 : str
        The ticker symbol for the first fund.
    fund_2 : str
        The ticker symbol for the second fund.
    fund_1_weight : float, optional
        Target allocation to the first fund; the second fund receives the
    ↪remainder (default is 0.5).
    rebalance_months : int, optional
        Rebalance on the last trading day of every  $k$ -th month, counting the
    ↪start month as month 1 (default is 2).
    transaction_fee : float, optional
        Fee charged on the dollar amount of each buy or sell order at a
    ↪rebalance (default is 0.0025).
    start_date : optional
        Start date for the data. If None, uses the earliest date both funds
    ↪have data (default is None).
```

```

    end_date : optional
        End date for the data. If None, uses the latest available date (default_
↳ is None).

Returns
-----
pd.DataFrame
    DataFrame with the normalized fund prices, the value of each holding,
↳ the normalized
        strategy value, the strategy daily return, and the rebalance flag and
↳ fee paid on each day.
    """

    # Combine the adjusted close prices over the common date range
    df = fund_data[fund_1][[f"{fund_1}_Adj_Close"]].merge(
        fund_data[fund_2][[f"{fund_2}_Adj_Close"]], left_index=True,
↳ right_index=True
    )

    # Filter by date range
    df = df.loc[start_date:end_date]

    # Normalize prices to start at 1
    df = df / df.iloc[0]

    # Calculate the return series for each fund
    df[f"{fund_1}_Return"] = df[f"{fund_1}_Adj_Close"].pct_change()
    df[f"{fund_2}_Return"] = df[f"{fund_2}_Adj_Close"].pct_change()

    # Calculate the drawdowns for each fund
    df[f"{fund_1}_Drawdown"] = (
        df[f"{fund_1}_Adj_Close"] / df[f"{fund_1}_Adj_Close"].cummax() - 1
    )
    df[f"{fund_2}_Drawdown"] = (
        df[f"{fund_2}_Adj_Close"] / df[f"{fund_2}_Adj_Close"].cummax() - 1
    )
    df[f"{fund_1}_Drawdown_Daily_Max"] = df[f"{fund_1}_Drawdown"].cummin()
    df[f"{fund_2}_Drawdown_Daily_Max"] = df[f"{fund_2}_Drawdown"].cummin()

    # Flag the last trading day of each month, excluding the final row since
↳ that month may be incomplete
    month = pd.Series(df.index.to_period("M"), index=df.index)
    month_end = month != month.shift(-1)
    month_end.iloc[-1] = False

    # Rebalance on the last trading day of every k-th month
    month_number = (

```

```

        (df.index.year - df.index[0].year) * 12
        + (df.index.month - df.index[0].month)
        + 1
    )
    rebalance = month_end & (month_number % rebalance_months == 0)

    fund_2_weight = 1 - fund_1_weight

    # Initial purchase at the target allocation with a starting portfolio value of 1
    fund_1_shares = fund_1_weight / df[f"{fund_1}_Adj_Close"].iloc[0]
    fund_2_shares = fund_2_weight / df[f"{fund_2}_Adj_Close"].iloc[0]

    fund_1_values, fund_2_values, fees = [], [], []

    for date, row in df.iterrows():
        fund_1_value = fund_1_shares * row[f"{fund_1}_Adj_Close"]
        fund_2_value = fund_2_shares * row[f"{fund_2}_Adj_Close"]
        fee = 0.0

        if rebalance.loc[date]:
            total_value = fund_1_value + fund_2_value

            # Fee on the dollar amount traded in each fund to get back to the target allocation
            fee = transaction_fee * (
                abs(fund_1_weight * total_value - fund_1_value)
                + abs(fund_2_weight * total_value - fund_2_value)
            )
            total_value -= fee

            fund_1_value = fund_1_weight * total_value
            fund_2_value = fund_2_weight * total_value
            fund_1_shares = fund_1_value / row[f"{fund_1}_Adj_Close"]
            fund_2_shares = fund_2_value / row[f"{fund_2}_Adj_Close"]

        fund_1_values.append(fund_1_value)
        fund_2_values.append(fund_2_value)
        fees.append(fee)

    df[f"{fund_1}_Value"] = fund_1_values
    df[f"{fund_2}_Value"] = fund_2_values
    df["Strategy_Adj_Close"] = df[f"{fund_1}_Value"] + df[f"{fund_2}_Value"]
    df["Strategy_Return"] = df["Strategy_Adj_Close"].pct_change()

    # Calculate the drawdown for the strategy
    df["Strategy_Drawdown"] = (

```

```

        df["Strategy_Adj_Close"] / df["Strategy_Adj_Close"].cummax() - 1
    )
    df["Strategy_Drawdown_Daily_Max"] = df["Strategy_Drawdown"].cummin()

    df["Rebalance"] = rebalance
    df["Transaction_Fee"] = fees

    return df

```

Now, we'll execute the strategy for TQQQ and TMF, as the paper does with a start date of 2/11/2010 and an end date of 10/4/2020.

```

[31]: tqqq_tmf = rebalance_strategy(
        fund_data=fund_data,
        fund_1="TQQQ",
        fund_2="TMF",
        fund_1_weight=0.5,
        rebalance_months=2,
        transaction_fee=0.0025, # 0.25% transaction fee
        start_date=None,
        end_date="2020-10-04",
    )

    print("Table E17: TQQQ/TMF Rebalance Strategy Data")
    display(tqqq_tmf)

```

Table E17: TQQQ/TMF Rebalance Strategy Data

	TQQQ_Adj_Close	TMF_Adj_Close	TQQQ_Return	TMF_Return	\
Date					
2010-02-11	1.00	1.00	NaN	NaN	
2010-02-12	1.00	1.01	0.00	0.01	
2010-02-16	1.04	1.02	0.04	0.01	
2010-02-17	1.06	0.99	0.02	-0.03	
2010-02-18	1.08	0.97	0.02	-0.02	
...	...	...	...	...	
2020-09-28	75.16	5.83	0.06	-0.01	
2020-09-29	74.37	5.85	-0.01	0.00	
2020-09-30	75.85	5.69	0.02	-0.03	
2020-10-01	79.40	5.73	0.05	0.01	
2020-10-02	72.85	5.66	-0.08	-0.01	

	TQQQ_Drawdown	TMF_Drawdown	TQQQ_Drawdown_Daily_Max	\
Date				
2010-02-11	0.00	0.00		0.00
2010-02-12	0.00	0.00		0.00
2010-02-16	0.00	0.00		0.00
2010-02-17	0.00	-0.03		0.00
2010-02-18	0.00	-0.05		0.00

...	...	...	...
2020-09-28	-0.26	-0.20	-0.70
2020-09-29	-0.26	-0.20	-0.70
2020-09-30	-0.25	-0.22	-0.70
2020-10-01	-0.21	-0.22	-0.70
2020-10-02	-0.28	-0.23	-0.70

Date	TMF_Drawdown_Daily_Max	TQQQ_Value	TMF_Value	Strategy_Adj_Close \
2010-02-11	0.00	0.50	0.50	1.00
2010-02-12	0.00	0.50	0.51	1.01
2010-02-16	0.00	0.52	0.51	1.03
2010-02-17	-0.03	0.53	0.49	1.02
2010-02-18	-0.05	0.54	0.49	1.03
...	...	...	...	...
2020-09-28	-0.53	23.78	19.49	43.28
2020-09-29	-0.53	23.53	19.56	43.09
2020-09-30	-0.53	21.51	21.51	43.02
2020-10-01	-0.53	22.52	21.63	44.15
2020-10-02	-0.53	20.66	21.37	42.03

Date	Strategy_Return	Strategy_Drawdown	Strategy_Drawdown_Daily_Max \
2010-02-11	NaN	0.00	0.00
2010-02-12	0.01	0.00	0.00
2010-02-16	0.02	0.00	0.00
2010-02-17	-0.01	-0.01	-0.01
2010-02-18	0.00	-0.01	-0.01
...	...	...	...
2020-09-28	0.03	-0.17	-0.42
2020-09-29	-0.00	-0.17	-0.42
2020-09-30	-0.00	-0.17	-0.42
2020-10-01	0.03	-0.15	-0.42
2020-10-02	-0.05	-0.19	-0.42

Date	Rebalance	Transaction_Fee
2010-02-11	False	0.00
2010-02-12	False	0.00
2010-02-16	False	0.00
2010-02-17	False	0.00
2010-02-18	False	0.00
...	...	...
2020-09-28	False	0.00
2020-09-29	False	0.00
2020-09-30	True	0.01
2020-10-01	False	0.00
2020-10-02	False	0.00

[2680 rows x 16 columns]

And then we can check the summary statistics to confirm that our implementation of the strategy matches the results reported in the paper.

```
[32]: # Create one empty dataframe for all summary statistics
all_strat_sum_stats = pd.DataFrame()

tqqq_tmf_sum_stats = summary_stats(
    fund_list=["TQQQ", "TMF"],
    df=tqqq_tmf[["Strategy_Return"]].dropna(),
    period="Daily",
    use_calendar_days=False,
    excel_export=False,
    pickle_export=False,
    output_confirmation=False,
)

# Rename the index to indicate the strategy
tqqq_tmf_sum_stats.index = ["TQQQ/TMF Strategy"]

all_strat_sum_stats = pd.concat([all_strat_sum_stats, tqqq_tmf_sum_stats])

tqqq_sum_stats = summary_stats(
    fund_list=["TQQQ", "TMF"],
    df=tqqq_tmf[["TQQQ_Return"]].dropna(),
    period="Daily",
    use_calendar_days=False,
    excel_export=False,
    pickle_export=False,
    output_confirmation=False,
)

# Rename the index to indicate the strategy
tqqq_sum_stats.index = ["TQQQ"]

all_strat_sum_stats = pd.concat([all_strat_sum_stats, tqqq_sum_stats])

tmf_sum_stats = summary_stats(
    fund_list=["TQQQ", "TMF"],
    df=tqqq_tmf[["TMF_Return"]].dropna(),
    period="Daily",
    use_calendar_days=False,
    excel_export=False,
    pickle_export=False,
    output_confirmation=False,
)
```

```
# Rename the index to indicate the strategy
tmf_sum_stats.index = ["TMF"]

all_strat_sum_stats = pd.concat([all_strat_sum_stats, tmf_sum_stats])

print("Table E18: Strategy Summary Statistics")
display(all_strat_sum_stats)
```

Table E18: Strategy Summary Statistics

	Number of Observations	Data Start Date	Data End Date	\
TQQQ/TMF Strategy	2679	2010-02-12	2020-10-02	
TQQQ	2679	2010-02-12	2020-10-02	
TMF	2679	2010-02-12	2020-10-02	

	Annual Mean Return (Arithmetic)	Annualized Volatility	\
TQQQ/TMF Strategy	0.39	0.28	
TQQQ	0.57	0.58	
TMF	0.26	0.44	

	Annualized Sharpe Ratio	CAGR (Geometric)	\
TQQQ/TMF Strategy	1.40	0.42	
TQQQ	0.99	0.50	
TMF	0.59	0.18	

	Daily Max Return	Daily Max Return (Date)	Daily Min Return	\
TQQQ/TMF Strategy	0.11	2020-03-20	-0.16	
TQQQ	0.27	2020-03-13	-0.34	
TMF	0.20	2020-03-20	-0.18	

	Daily Min Return (Date)	Max Drawdown	Peak	Trough	\
TQQQ/TMF Strategy	2020-03-18	-0.42	2020-03-06	2020-03-18	
TQQQ	2020-03-16	-0.70	2020-02-19	2020-03-20	
TMF	2020-03-18	-0.53	2012-07-25	2013-08-21	

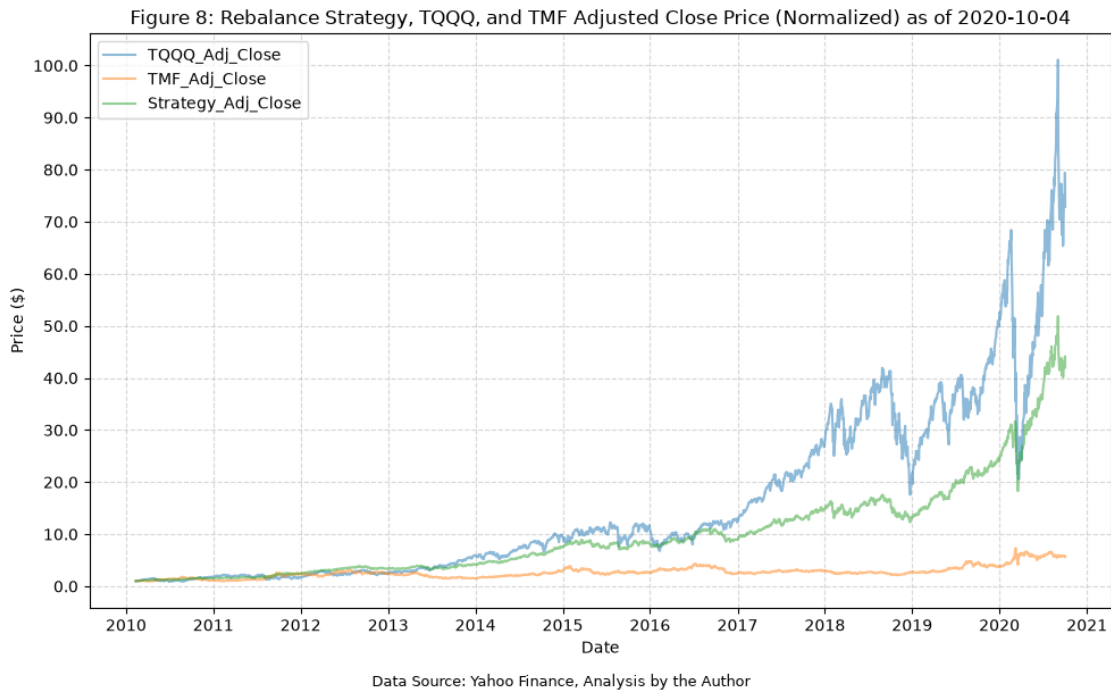
	Recovery Date	Calendar Days to Recovery	MAR Ratio
TQQQ/TMF Strategy	2020-05-07	50	1.00
TQQQ	2020-07-10	112	0.71
TMF	2015-01-05	502	0.33

The paper reports a CAGR of 44.9%, and our implementation finds 42%. The discrepancy here is likely the end date. The paper references 10/4/2020, but it seems the data points it uses are actually from 9/2/2020.

0.8.1 Figure 8: Strategy Performance With Bimonthly Rebalancing

We can reproduce figure 8, which shows the performance of TQQQ, TMF, and the strategy over time.

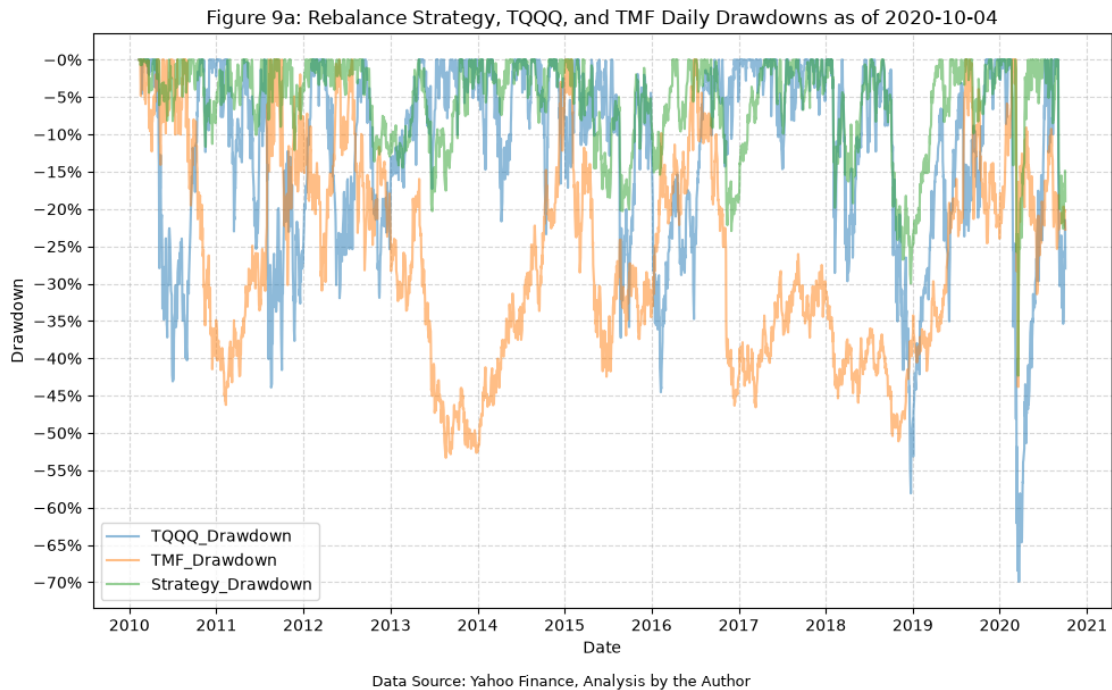
```
[33]: plot_time_series(
    df=tqqq_tmf,
    plot_start_date=None,
    plot_end_date=None,
    plot_columns=["TQQQ_Adj_Close", "TMF_Adj_Close", "Strategy_Adj_Close"],
    title="Figure 8: Rebalance Strategy, TQQQ, and TMF Adjusted Close Price_
↪(Normalized) as of 2020-10-04",
    x_label="Date",
    x_format="Year",
    x_tick_spacing=1,
    x_tick_start=None,
    x_tick_rotation=0,
    y_label="Price ($)",
    y_format="Decimal",
    y_format_decimal_places=1,
    y_tick_spacing="Auto",
    y_tick_rotation=0,
    caption="Data Source: Yahoo Finance, Analysis by the Author",
)
```



0.8.2 Figure 9: Maximum end-of-month draw downs corresponding to fig. 8

Next, we can recreate figure 9, with the maximum DDD and MDD. First, the full daily drawdown plot.

```
[34]: plot_time_series(
    df=tqqq_tmf,
    plot_start_date=None,
    plot_end_date=None,
    plot_columns=["TQQQ_Drawdown", "TMF_Drawdown", "Strategy_Drawdown"],
    title="Figure 9a: Rebalance Strategy, TQQQ, and TMF Daily Drawdowns as of 2020-10-04",
    x_label="Date",
    x_format="Year",
    x_tick_spacing=1,
    x_tick_start=None,
    x_tick_rotation=0,
    y_label="Drawdown",
    y_format="Percentage",
    y_format_decimal_places=0,
    y_tick_spacing="Auto",
    y_tick_rotation=0,
    caption="Data Source: Yahoo Finance, Analysis by the Author",
)
```



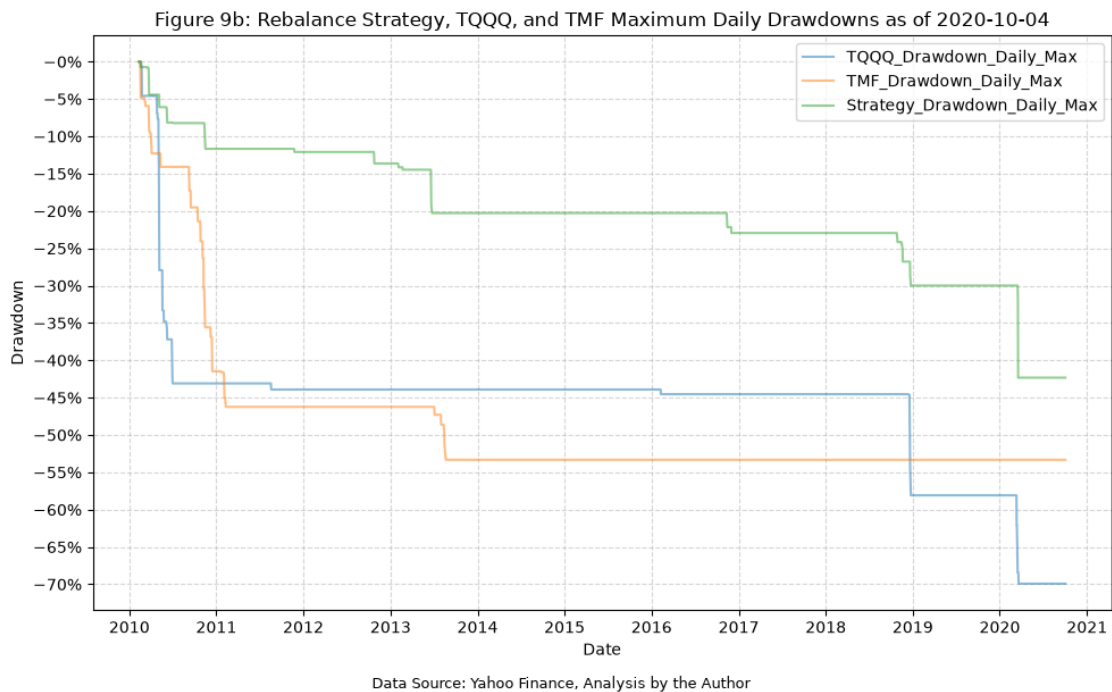
Next, the maximum DDD.

```
[35]: plot_time_series(
    df=tqqq_tmf,
    plot_start_date=None,
```

```

plot_end_date=None,
plot_columns=[
    "TQQQ_Drawdown_Daily_Max",
    "TMF_Drawdown_Daily_Max",
    "Strategy_Drawdown_Daily_Max",
],
title="Figure 9b: Rebalance Strategy, TQQQ, and TMF Maximum Daily Drawdowns_
as of 2020-10-04",
x_label="Date",
x_format="Year",
x_tick_spacing=1,
x_tick_start=None,
x_tick_rotation=0,
y_label="Drawdown",
y_format="Percentage",
y_format_decimal_places=0,
y_tick_spacing="Auto",
y_tick_rotation=0,
caption="Data Source: Yahoo Finance, Analysis by the Author",
)

```



Finally, the maximum MDD.

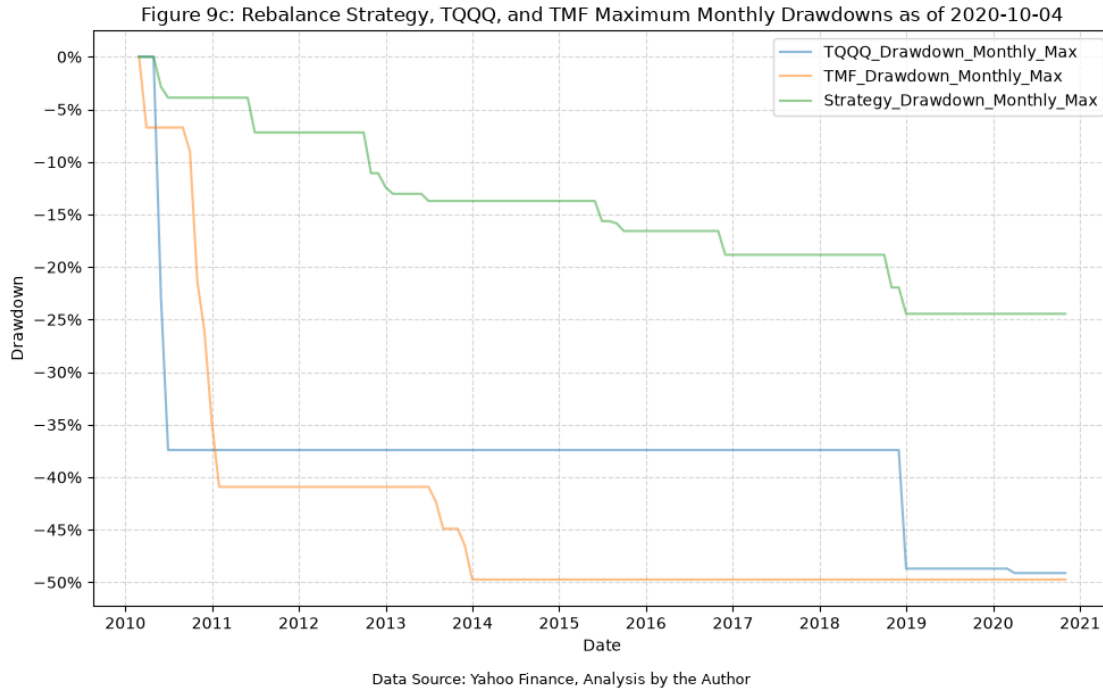
```

[36]: tqqq_tmf_monthly = tqqq_tmf.resample("ME").last()

# Calculate the monthly drawdowns using end-of-month prices only, then the
↳running maximum monthly drawdown
for fund in ["TQQQ", "TMF", "Strategy"]:
    tqqq_tmf_monthly[f"{fund}_Drawdown_Monthly"] = (
        tqqq_tmf_monthly[f"{fund}_Adj_Close"]
        / tqqq_tmf_monthly[f"{fund}_Adj_Close"].cummax()
        - 1
    )
    tqqq_tmf_monthly[f"{fund}_Drawdown_Monthly_Max"] = tqqq_tmf_monthly[
        f"{fund}_Drawdown_Monthly"
    ].cummin()

plot_time_series(
    df=tqqq_tmf_monthly,
    plot_start_date=None,
    plot_end_date=None,
    plot_columns=[
        "TQQQ_Drawdown_Monthly_Max",
        "TMF_Drawdown_Monthly_Max",
        "Strategy_Drawdown_Monthly_Max",
    ],
    title="Figure 9c: Rebalance Strategy, TQQQ, and TMF Maximum Monthly
↳Drawdowns as of 2020-10-04",
    x_label="Date",
    x_format="Year",
    x_tick_spacing=1,
    x_tick_start=None,
    x_tick_rotation=0,
    y_label="Drawdown",
    y_format="Percentage",
    y_format_decimal_places=0,
    y_tick_spacing="Auto",
    y_tick_rotation=0,
    caption="Data Source: Yahoo Finance, Analysis by the Author",
)

```



Our plot above matches figure 9 from the paper. We find a maximum MDD of 49% for TQQQ, 50% for TMF, and 24% for the strategy, compared with 49.1% for TQQQ and 49.7% for TMF from section 2.0 of the paper, and 24.4% for the strategy from section 3.2.

Note that the MDD uses end-of-month prices for both the drawdown and the prior peak, so a peak or trough within a month (such as TQQQ's peak on 2/19/2020) is not captured. This is why the MDD values are lower than the DDD values shown in the previous plot.

0.9 Section 3.1: Optimum Balance Interval for Risk Mitigation Strategy

0.9.1 Figure 10: Strategy performance on 9/2/20 as a function of rebalancing interval

Section 3.1 of the paper examines how the performance of the strategy on 9/2/2020 (the date of the peak normalized value) changes as the rebalancing interval is varied from one to six months, both with and without the 0.25% transaction fee. We can reproduce this by running the strategy for each rebalancing interval.

```
[37]: # Calculate the normalized value of the strategy on 9/2/2020 for rebalancing_
      ↪ intervals of 1 to 6 months, with and without the transaction fee
rebalance_interval_results = pd.DataFrame(
    index=pd.Index(range(1, 7), name="Rebalance Interval (Months)")
)

for k in rebalance_interval_results.index:
    for fee, col in [(0.0, "No Transaction Fee"), (0.0025, "Transaction Fee = 0.
    ↪ 25%")]:
```

```

df_k = rebalance_strategy(
    fund_data=fund_data,
    fund_1="TQQQ",
    fund_2="TMF",
    fund_1_weight=0.5,
    rebalance_months=k,
    transaction_fee=fee,
    start_date=None,
    end_date="2020-09-02",
)
rebalance_interval_results.loc[k, col] = df_k["Strategy_Adj_Close"].
    iloc[-1]

print("Figure 10: Strategy performance on 9/2/20 as a function of rebalancing_
    interval")
display(rebalance_interval_results)

```

Figure 10: Strategy performance on 9/2/20 as a function of rebalancing interval

	No Transaction Fee	Transaction Fee = 0.25%
Rebalance Interval (Months)		
1	50.65	49.33
2	52.82	51.86
3	41.31	40.78
4	44.83	44.34
5	49.15	48.57
6	39.33	38.99

0.10 Section 3.2: Risk Mitigation Strategy Modified to Account for “Black Swan” Events

In section 3.2, the paper proposes the idea of switching the allocation from the rebalance strategy to cash or a cash substitute, after a severe drop in the price of TQQQ. Specifically:

Such large one-day losses might cause even disciplined investors to panic sell. Here we propose a simple filter to overlay the risk mitigation strategy described above. The basic idea is to switch the simple strategy to cash, or a near-cash substitute (here we use the 7-10 year Treasury ETF, IEF) whenever an automated alert indicates that the daily return of TQQQ has dropped below a preset minimum. A return to the standard strategy is then only made when a second alert indicates that the daily adjusted close of TQQQ has returned to and exceeded the value it had when the switch was made.

This is an interesting idea, and we will implement it next, but first we need to pull data for IEF.

```

[38]: yf_pull_data(
    base_directory=DATA_DIR,
    ticker="IEF",
    adjusted=False,
    source="Yahoo_Finance",

```

```

asset_class="Exchange_Traded_Funds",
excel_export=True,
pickle_export=True,
output_confirmation=False,
)

```

```

[38]:
Adj Close  Close  High  Low  Open  Volume
Date
2002-07-30    40.00  81.77  82.12  81.70  81.94    41300
2002-07-31    40.37  82.52  82.58  82.05  82.05    32600
2002-08-01    40.53  82.86  82.90  82.52  82.54    71400
2002-08-02    40.85  83.50  83.70  82.90  83.02   120300
2002-08-05    41.05  83.92  83.92  83.53  83.68   159300
...
2026-10-16    90.42  90.42  90.54  90.12  90.15  10831400
2026-10-19    90.08  90.08  90.38  90.02  90.32  10400200
2026-10-20    90.25  90.25  90.50  90.12  90.29   8956100
2026-10-21    90.23  90.23  90.47  90.00  90.39   8909200
2026-10-22    91.20  91.20  91.40  90.65  90.75  16518600

```

[6098 rows x 6 columns]

Then append it to the existing dictionary.

```

[39]: fund_list.append("IEF")

# Load the data for each ETF, rename the columns to include the ticker, store_
↳ it in the fund_data dictionary
for fund in fund_list:
    data = load_data(
        base_directory=DATA_DIR,
        ticker=fund,
        source="Yahoo_Finance",
        asset_class="Exchange_Traded_Funds",
        timeframe="Daily",
        file_format="pickle",
    )

    data = data.rename(
        columns={
            "Adj Close": f"{fund}_Adj_Close",
            "Close": f"{fund}_Close",
            "High": f"{fund}_High",
            "Low": f"{fund}_Low",
            "Open": f"{fund}_Open",
            "Volume": f"{fund}_Volume",
        }
    )

```

```
fund_data[fund] = data
```

And we will use the following helper function. It implements the paper's filter with a threshold of 20% for the one-day drop in TQQQ (the paper's ∂^*) and IEF as the cash substitute. There are a few implementation details worth noting:

- The paper's pseudo-code in figure 11 sets the switch-back price v^* to the price of TQQQ on the day of the drop, $v(n)$. Implemented that way, TQQQ closes above v^* the very next day (3/10/2020), and the filter switches in and out three times during March 2020. However, the paper states that the return to the standard strategy occurs on 5/18/2020, which is the first date that TQQQ closes above its price on the day *before* the drop, $v(n-1)$. This also matches the paper's description of TQQQ needing to return to "the value it had when the switch was made", so we use $v(n-1)$ for v^* .
- While the strategy is holding IEF, the scheduled bimonthly rebalances are skipped.
- Each switch into or out of IEF pays the 0.25% transaction fee on both the sell and the buy.

```
[40]: def rebalance_strategy_bs(
    fund_data: dict,
    fund_1: str,
    fund_2: str,
    cash_fund: str = "IEF",
    drop_threshold: float = 0.20,
    fund_1_weight: float = 0.5,
    rebalance_months: int = 2,
    transaction_fee: float = 0.0025,
    start_date=None,
    end_date=None,
) -> pd.DataFrame:
    """
    Simulate a two-fund portfolio that is periodically rebalanced back to a
    ↪target allocation, with a
    "black swan" filter that switches the portfolio to a cash substitute after
    ↪a large one-day drop in
    the first fund.

    When the daily return of the first fund falls below -drop_threshold, the
    ↪entire portfolio is sold and
    moved into the cash fund at that day's close, and the first fund's price
    ↪from the prior day's close
    (its value before the drop) is recorded. The portfolio stays in the cash
    ↪fund, skipping any scheduled
    rebalances, until the first fund's price closes above the recorded price,
    ↪at which point the cash fund
    is sold and the two-fund portfolio is bought back at the target allocation.

    Parameters
    -----
```

```

fund_data : dict
    Dictionary containing DataFrames for each fund.
fund_1 : str
    The ticker symbol for the first fund. Its daily return triggers the
↪switch to the cash fund.
fund_2 : str
    The ticker symbol for the second fund.
cash_fund : str, optional
    The ticker symbol for the cash substitute held after the filter is
↪triggered (default is "IEF").
drop_threshold : float, optional
    One-day decline in the first fund that triggers the switch to the cash
↪fund (default is 0.20).
fund_1_weight : float, optional
    Target allocation to the first fund; the second fund receives the
↪remainder (default is 0.5).
rebalance_months : int, optional
    Rebalance on the last trading day of every k-th month, counting the
↪start month as month 1 (default is 2).
transaction_fee : float, optional
    Fee charged on the dollar amount of each buy or sell order at a
↪rebalance or switch (default is 0.0025).
start_date : optional
    Start date for the data. If None, uses the earliest date all three
↪funds have data (default is None).
end_date : optional
    End date for the data. If None, uses the latest available date (default
↪is None).

Returns
-----
pd.DataFrame
    DataFrame with the normalized fund prices, the value of each holding,
↪the normalized
    strategy value, the strategy daily return, the rebalance and switch
↪flags, whether the
    portfolio is held in the cash fund, and the fee paid on each day.
    """

# Combine the adjusted close prices over the common date range
df = (
    fund_data[fund_1][[f"{fund_1}_Adj_Close"]]
    .merge(
        fund_data[fund_2][[f"{fund_2}_Adj_Close"]],
        left_index=True,
        right_index=True,

```

```

    )
    .merge(
        fund_data[cash_fund][[f"{cash_fund}_Adj_Close"]],
        left_index=True,
        right_index=True,
    )
)

# Filter by date range
df = df.loc[start_date:end_date]

# Normalize prices to start at 1
df = df / df.iloc[0]

# Calculate the return series for each fund
df[f"{fund_1}_Return"] = df[f"{fund_1}_Adj_Close"].pct_change()
df[f"{fund_2}_Return"] = df[f"{fund_2}_Adj_Close"].pct_change()

# Calculate the drawdowns for each fund
df[f"{fund_1}_Drawdown"] = (
    df[f"{fund_1}_Adj_Close"] / df[f"{fund_1}_Adj_Close"].cummax() - 1
)
df[f"{fund_2}_Drawdown"] = (
    df[f"{fund_2}_Adj_Close"] / df[f"{fund_2}_Adj_Close"].cummax() - 1
)
df[f"{fund_1}_Drawdown_Daily_Max"] = df[f"{fund_1}_Drawdown"].cummin()
df[f"{fund_2}_Drawdown_Daily_Max"] = df[f"{fund_2}_Drawdown"].cummin()

# Flag the last trading day of each month, excluding the final row since
↳ that month may be incomplete
month = pd.Series(df.index.to_period("M"), index=df.index)
month_end = month != month.shift(-1)
month_end.iloc[-1] = False

# Rebalance on the last trading day of every k-th month
month_number = (
    (df.index.year - df.index[0].year) * 12
    + (df.index.month - df.index[0].month)
    + 1
)
rebalance = month_end & (month_number % rebalance_months == 0)

fund_2_weight = 1 - fund_1_weight

# Initial purchase at the target allocation with a starting portfolio value
↳ of 1
fund_1_shares = fund_1_weight / df[f"{fund_1}_Adj_Close"].iloc[0]

```

```

fund_2_shares = fund_2_weight / df[f"{fund_2}_Adj_Close"].iloc[0]
cash_fund_shares = 0.0

# Price of the first fund before the drop that triggered the switch to the
↪ cash fund (v* in the paper); 0 when
# not in the cash fund. The paper's flowchart sets v* = v(n), but its
↪ switch back on 5/18/20 matches v* = v(n-1)
switch_price = 0.0
prior_close = df[f"{fund_1}_Adj_Close"].shift(1)

fund_1_values, fund_2_values, cash_fund_values = [], [], []
rebalanced, switched, in_cash_fund, fees = [], [], [], []

for date, row in df.iterrows():
    fund_1_value = fund_1_shares * row[f"{fund_1}_Adj_Close"]
    fund_2_value = fund_2_shares * row[f"{fund_2}_Adj_Close"]
    cash_fund_value = cash_fund_shares * row[f"{cash_fund}_Adj_Close"]
    fee = 0.0
    rebalance_today = False
    switch_today = False

    if switch_price > 0:
        # Currently in the cash fund; switch back once the first fund
        ↪ closes above the switch price
        if row[f"{fund_1}_Adj_Close"] > switch_price:
            total_value = cash_fund_value

            # Fee on selling the cash fund and buying the two funds
            fee = transaction_fee * 2 * total_value
            total_value -= fee

            fund_1_value = fund_1_weight * total_value
            fund_2_value = fund_2_weight * total_value
            cash_fund_value = 0.0
            fund_1_shares = fund_1_value / row[f"{fund_1}_Adj_Close"]
            fund_2_shares = fund_2_value / row[f"{fund_2}_Adj_Close"]
            cash_fund_shares = 0.0

            switch_price = 0.0
            switch_today = True

    elif row[f"{fund_1}_Return"] < -drop_threshold:
        # One-day drop in the first fund exceeds the threshold; sell both
        ↪ funds and buy the cash fund
        total_value = fund_1_value + fund_2_value

        # Fee on selling the two funds and buying the cash fund

```

```

    fee = transaction_fee * 2 * total_value
    total_value -= fee

    cash_fund_value = total_value
    fund_1_value = 0.0
    fund_2_value = 0.0
    cash_fund_shares = cash_fund_value / row[f"{cash_fund}_Adj_Close"]
    fund_1_shares = 0.0
    fund_2_shares = 0.0

    switch_price = prior_close.loc[date]
    switch_today = True

elif rebalance.loc[date]:
    total_value = fund_1_value + fund_2_value

    # Fee on the dollar amount traded in each fund to get back to the
    ↪target allocation
    fee = transaction_fee * (
        abs(fund_1_weight * total_value - fund_1_value)
        + abs(fund_2_weight * total_value - fund_2_value)
    )
    total_value -= fee

    fund_1_value = fund_1_weight * total_value
    fund_2_value = fund_2_weight * total_value
    fund_1_shares = fund_1_value / row[f"{fund_1}_Adj_Close"]
    fund_2_shares = fund_2_value / row[f"{fund_2}_Adj_Close"]
    rebalance_today = True

    fund_1_values.append(fund_1_value)
    fund_2_values.append(fund_2_value)
    cash_fund_values.append(cash_fund_value)
    rebalanced.append(rebalance_today)
    switched.append(switch_today)
    in_cash_fund.append(switch_price > 0)
    fees.append(fee)

df[f"{fund_1}_Value"] = fund_1_values
df[f"{fund_2}_Value"] = fund_2_values
df[f"{cash_fund}_Value"] = cash_fund_values
df["Strategy_Adj_Close"] = (
    df[f"{fund_1}_Value"] + df[f"{fund_2}_Value"] + df[f"{cash_fund}_Value"]
)
df["Strategy_Return"] = df["Strategy_Adj_Close"].pct_change()

# Calculate the drawdown for the strategy

```

```

df["Strategy_Drawdown"] = (
    df["Strategy_Adj_Close"] / df["Strategy_Adj_Close"].cummax() - 1
)
df["Strategy_Drawdown_Daily_Max"] = df["Strategy_Drawdown"].cummin()

df["Rebalance"] = rebalanced
df["Switch"] = switched
df[f"In_{cash_fund}"] = in_cash_fund
df["Transaction_Fee"] = fees

return df

```

Now, we'll execute the "black swan" strategy for TQQQ and TMF, as the paper does with a start date of 2/11/2010 and an end date of 10/4/2020.

```

[41]: tqqq_tmf_bs = rebalance_strategy_bs(
    fund_data=fund_data,
    fund_1="TQQQ",
    fund_2="TMF",
    cash_fund="IEF",
    drop_threshold=0.2,
    fund_1_weight=0.5,
    rebalance_months=2,
    transaction_fee=0.0025, # 0.25% transaction fee
    start_date=None,
    end_date="2020-10-04",
)

print("Table E19: TQQQ/TMF Black Swan Strategy Data")
display(tqqq_tmf_bs)

```

Table E19: TQQQ/TMF Black Swan Strategy Data

	TQQQ_Adj_Close	TMF_Adj_Close	IEF_Adj_Close	TQQQ_Return \	
Date					
2010-02-11	1.00	1.00	1.00	NaN	
2010-02-12	1.00	1.01	1.00	0.00	
2010-02-16	1.04	1.02	1.00	0.04	
2010-02-17	1.06	0.99	1.00	0.02	
2010-02-18	1.08	0.97	1.00	0.02	
...	...	...	...	...	
2020-09-28	75.16	5.83	1.69	0.06	
2020-09-29	74.37	5.85	1.69	-0.01	
2020-09-30	75.85	5.69	1.69	0.02	
2020-10-01	79.40	5.73	1.69	0.05	
2020-10-02	72.85	5.66	1.69	-0.08	
	TMF_Return	TQQQ_Drawdown	TMF_Drawdown	TQQQ_Drawdown_Daily_Max	\
Date					

2010-02-11	NaN	0.00	0.00	0.00
2010-02-12	0.01	0.00	0.00	0.00
2010-02-16	0.01	0.00	0.00	0.00
2010-02-17	-0.03	0.00	-0.03	0.00
2010-02-18	-0.02	0.00	-0.05	0.00
...	...	...	...	...
2020-09-28	-0.01	-0.26	-0.20	-0.70
2020-09-29	0.00	-0.26	-0.20	-0.70
2020-09-30	-0.03	-0.25	-0.22	-0.70
2020-10-01	0.01	-0.21	-0.22	-0.70
2020-10-02	-0.01	-0.28	-0.23	-0.70

Date	TMF_Drawdown_Daily_Max	TQQQ_Value	TMF_Value	IEF_Value	\
2010-02-11	0.00	0.50	0.50	0.00	
2010-02-12	0.00	0.50	0.51	0.00	
2010-02-16	0.00	0.52	0.51	0.00	
2010-02-17	-0.03	0.53	0.49	0.00	
2010-02-18	-0.05	0.54	0.49	0.00	
...	...	...	...	...	
2020-09-28	-0.53	22.53	18.47	0.00	
2020-09-29	-0.53	22.29	18.52	0.00	
2020-09-30	-0.53	20.37	20.37	0.00	
2020-10-01	-0.53	21.33	20.49	0.00	
2020-10-02	-0.53	19.57	20.25	0.00	

Date	Strategy_Adj_Close	Strategy_Return	Strategy_Drawdown	\
2010-02-11	1.00	NaN	0.00	
2010-02-12	1.01	0.01	0.00	
2010-02-16	1.03	0.02	0.00	
2010-02-17	1.02	-0.01	-0.01	
2010-02-18	1.03	0.00	-0.01	
...	...	...	...	
2020-09-28	40.99	0.03	-0.17	
2020-09-29	40.82	-0.00	-0.17	
2020-09-30	40.75	-0.00	-0.17	
2020-10-01	41.82	0.03	-0.15	
2020-10-02	39.81	-0.05	-0.19	

Date	Strategy_Drawdown_Daily_Max	Rebalance	Switch	In_IEF	\
2010-02-11	0.00	False	False	False	
2010-02-12	0.00	False	False	False	
2010-02-16	0.00	False	False	False	
2010-02-17	-0.01	False	False	False	
2010-02-18	-0.01	False	False	False	
...	...	...	...	...	

2020-09-28	-0.30	False	False	False
2020-09-29	-0.30	False	False	False
2020-09-30	-0.30	True	False	False
2020-10-01	-0.30	False	False	False
2020-10-02	-0.30	False	False	False

Date	Transaction_Fee
2010-02-11	0.00
2010-02-12	0.00
2010-02-16	0.00
2010-02-17	0.00
2010-02-18	0.00
...	...
2020-09-28	0.00
2020-09-29	0.00
2020-09-30	0.01
2020-10-01	0.00
2020-10-02	0.00

[2680 rows x 20 columns]

And then we can check the summary statistics to confirm that our implementation of the strategy matches the results reported in the paper.

```
[42]: tqqq_tmf_bs_sum_stats = summary_stats(
    fund_list=["TQQQ", "TMF"],
    df=tqqq_tmf_bs[["Strategy_Return"]].dropna(),
    period="Daily",
    use_calendar_days=False,
    excel_export=False,
    pickle_export=False,
    output_confirmation=False,
)

# Rename the index to indicate the strategy
tqqq_tmf_bs_sum_stats.index = ["TQQQ/TMF BS"]

all_strat_sum_stats = pd.concat([all_strat_sum_stats, tqqq_tmf_bs_sum_stats])

print("Table E20: Strategy Summary Statistics")
display(all_strat_sum_stats)
```

Table E20: Strategy Summary Statistics

	Number of Observations	Data Start Date	Data End Date	\
TQQQ/TMF Strategy	2679	2010-02-12	2020-10-02	
TQQQ	2679	2010-02-12	2020-10-02	
TMF	2679	2010-02-12	2020-10-02	

TQQQ/TMF BS	2679	2010-02-12	2020-10-02	
	Annual Mean Return (Arithmetic)	Annualized Volatility	\	
TQQQ/TMF Strategy	0.39	0.28		
TQQQ	0.57	0.58		
TMF	0.26	0.44		
TQQQ/TMF BS	0.38	0.26		
	Annualized Sharpe Ratio	CAGR (Geometric)	\	
TQQQ/TMF Strategy	1.40	0.42		
TQQQ	0.99	0.50		
TMF	0.59	0.18		
TQQQ/TMF BS	1.48	0.41		
	Daily Max Return	Daily Max Return (Date)	Daily Min Return	\
TQQQ/TMF Strategy	0.11	2020-03-20	-0.16	
TQQQ	0.27	2020-03-13	-0.34	
TMF	0.20	2020-03-20	-0.18	
TQQQ/TMF BS	0.07	2020-03-06	-0.09	
	Daily Min Return (Date)	Max Drawdown	Peak	Trough \
TQQQ/TMF Strategy	2020-03-18	-0.42	2020-03-06	2020-03-18
TQQQ	2020-03-16	-0.70	2020-02-19	2020-03-20
TMF	2020-03-18	-0.53	2012-07-25	2013-08-21
TQQQ/TMF BS	2020-09-03	-0.30	2018-08-29	2018-12-24
	Recovery Date	Calendar Days to Recovery	MAR Ratio	
TQQQ/TMF Strategy	2020-05-07	50	1.00	
TQQQ	2020-07-10	112	0.71	
TMF	2015-01-05	502	0.33	
TQQQ/TMF BS	2019-04-23	120	1.38	

0.10.1 Figure 12: Strategy performance with and without IEF filter

We can then plot figure 12, which compares the performance of the strategy with and without the black swan filter.

```
[43]: tqqq_tmf_both = tqqq_tmf.merge(
    tqqq_tmf_bs, on="Date", how="outer", suffixes=("", "_BS")
)

plot_time_series(
    df=tqqq_tmf_both,
    plot_start_date=None,
    plot_end_date=None,
    plot_columns=[
        "TQQQ_Adj_Close",
        "TMF_Adj_Close",
    ],
)
```

```

        "Strategy_Adj_Close",
        "Strategy_Adj_Close_BS",
    ],
    title="Figure 12a: Rebalance Strategy, Black Swan Strategy, TQQQ, and TMF_
Adjusted Close Price (Normalized) as of 2020-10-04",
    x_label="Date",
    x_format="Year",
    x_tick_spacing=1,
    x_tick_start=None,
    x_tick_rotation=0,
    y_label="Price ($)",
    y_format="Decimal",
    y_format_decimal_places=1,
    y_tick_spacing="Auto",
    y_tick_rotation=0,
    caption="Data Source: Yahoo Finance, Analysis by the Author",
)

```

Figure 12a: Rebalance Strategy, Black Swan Strategy, TQQQ, and TMF Adjusted Close Price (Normalized) as of 2020-10-04



Data Source: Yahoo Finance, Analysis by the Author

Zooming in on the data for 2020:

```

[44]: plot_time_series(
    df=tqqq_tmf_both[tqqq_tmf_both.index >= "2020-01-01"],
    plot_start_date=None,
    plot_end_date=None,
    plot_columns=["Strategy_Adj_Close", "Strategy_Adj_Close_BS"],
)

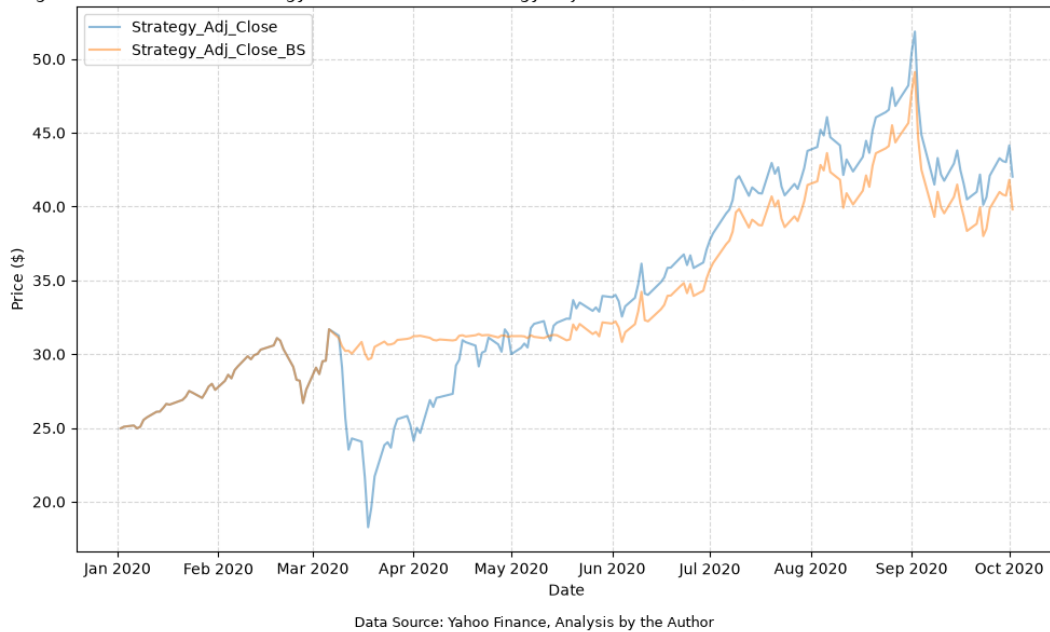
```

```

title="Figure 12b: Rebalance Strategy and Black Swan Strategy Adjusted_
Close Price (Normalized), 2020-01-01 to 2020-10-04",
x_label="Date",
x_format="Month",
x_tick_spacing=1,
x_tick_start=None,
x_tick_rotation=0,
y_label="Price ($)",
y_format="Decimal",
y_format_decimal_places=1,
y_tick_spacing="Auto",
y_tick_rotation=0,
caption="Data Source: Yahoo Finance, Analysis by the Author",
)

```

Figure 12b: Rebalance Strategy and Black Swan Strategy Adjusted Close Price (Normalized), 2020-01-01 to 2020-10-04



Interestingly, the BS strategy misses much of the COVID crash in March of 2020, while retaining nearly all of the performance throughout the rest of the date range.

0.10.2 Figure 13: Maximum daily draw downs with IEF filter used for mitigation strategy

Finally, figure 13 shows the maximum DDD with the BS filter applied.

```

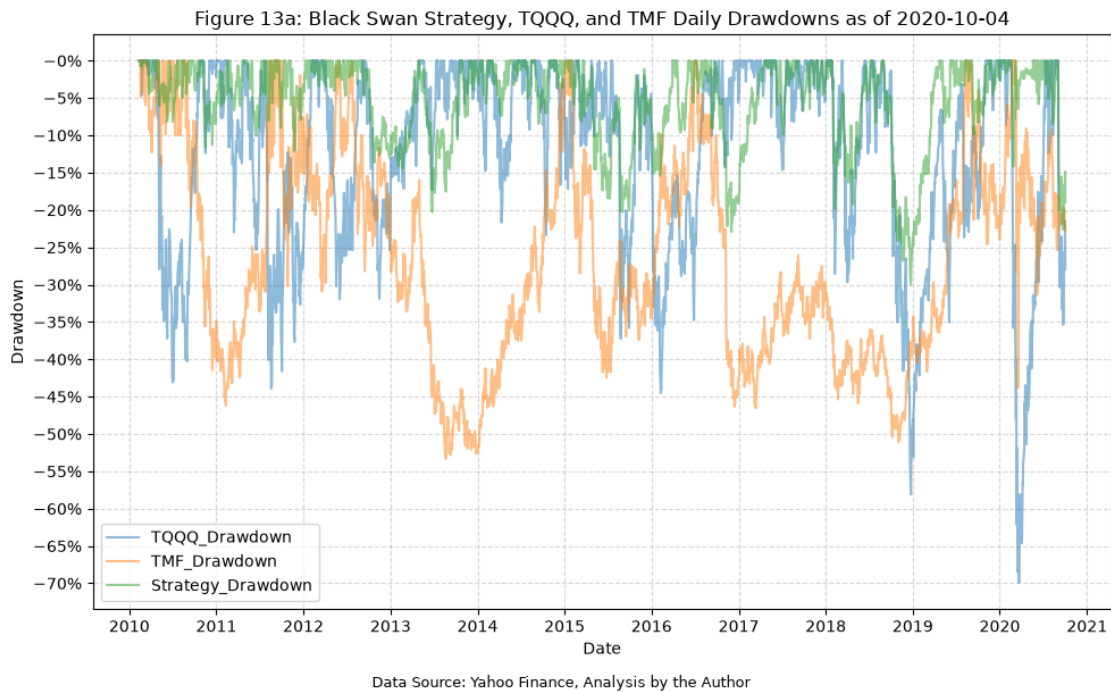
[45]: plot_time_series(
        df=tqqq_tmf_bs,
        plot_start_date=None,

```

```

plot_end_date=None,
plot_columns=["TQQQ_Drawdown", "TMF_Drawdown", "Strategy_Drawdown"],
title="Figure 13a: Black Swan Strategy, TQQQ, and TMF Daily Drawdowns as of 2020-10-04",
x_label="Date",
x_format="Year",
x_tick_spacing=1,
x_tick_start=None,
x_tick_rotation=0,
y_label="Drawdown",
y_format="Percentage",
y_format_decimal_places=0,
y_tick_spacing="Auto",
y_tick_rotation=0,
caption="Data Source: Yahoo Finance, Analysis by the Author",
)

```



Next, the maximum DDD.

```

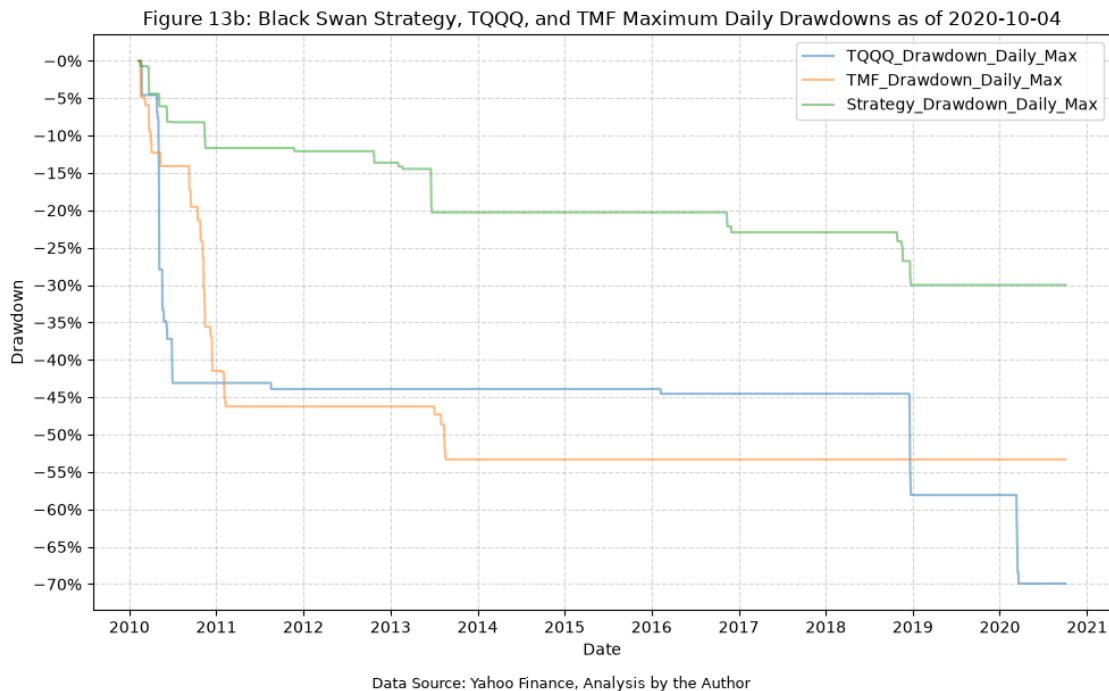
[46]: plot_time_series(
    df=tqqq_tmf_bs,
    plot_start_date=None,
    plot_end_date=None,
    plot_columns=[
        "TQQQ_Drawdown_Daily_Max",

```

```

    "TMF_Drawdown_Daily_Max",
    "Strategy_Drawdown_Daily_Max",
],
title="Figure 13b: Black Swan Strategy, TQQQ, and TMF Maximum Daily_
↳Drawdowns as of 2020-10-04",
x_label="Date",
x_format="Year",
x_tick_spacing=1,
x_tick_start=None,
x_tick_rotation=0,
y_label="Drawdown",
y_format="Percentage",
y_format_decimal_places=0,
y_tick_spacing="Auto",
y_tick_rotation=0,
caption="Data Source: Yahoo Finance, Analysis by the Author",
)

```



The above plot is the true replication of figure 13, and it does in fact appear to match. We see a maximum DDD of 30% for the BS strategy, and this does appear to match what is in the paper.

Finally, the maximum MDD.

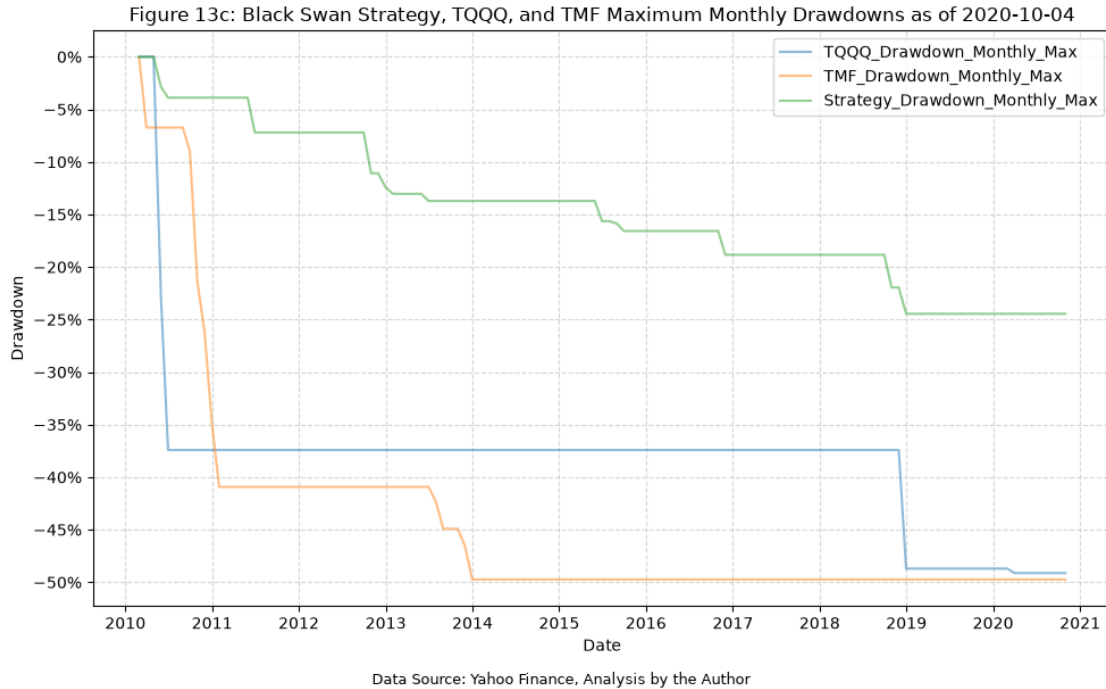
```
[47]: tqqq_tmf_bs_monthly = tqqq_tmf_bs.resample("ME").last()
```

```

# Calculate the monthly drawdowns using end-of-month prices only, then the
↳ running maximum monthly drawdown
for fund in ["TQQQ", "TMF", "Strategy"]:
    tqqq_tmf_bs_monthly[f"{fund}_Drawdown_Monthly"] = (
        tqqq_tmf_bs_monthly[f"{fund}_Adj_Close"]
        / tqqq_tmf_bs_monthly[f"{fund}_Adj_Close"].cummax()
        - 1
    )
    tqqq_tmf_bs_monthly[f"{fund}_Drawdown_Monthly_Max"] = tqqq_tmf_bs_monthly[
        f"{fund}_Drawdown_Monthly"
    ].cummin()

plot_time_series(
    df=tqqq_tmf_bs_monthly,
    plot_start_date=None,
    plot_end_date=None,
    plot_columns=[
        "TQQQ_Drawdown_Monthly_Max",
        "TMF_Drawdown_Monthly_Max",
        "Strategy_Drawdown_Monthly_Max",
    ],
    title="Figure 13c: Black Swan Strategy, TQQQ, and TMF Maximum Monthly
↳ Drawdowns as of 2020-10-04",
    x_label="Date",
    x_format="Year",
    x_tick_spacing=1,
    x_tick_start=None,
    x_tick_rotation=0,
    y_label="Drawdown",
    y_format="Percentage",
    y_format_decimal_places=0,
    y_tick_spacing="Auto",
    y_tick_rotation=0,
    caption="Data Source: Yahoo Finance, Analysis by the Author",
)

```



0.11 Extension 1: Expanding the date range for analysis

Above we saw that the rebalance strategy performed well during a time when the correlations between equity and bond funds were negative, which is the entirety of the range between 2010 and 2020. Now, let's expand the date range to present, and see if the results of the rebalance strategy hold up.

```
[48]: tqqq_tmf_full = rebalance_strategy(
    fund_data=fund_data,
    fund_1="TQQQ",
    fund_2="TMF",
    fund_1_weight=0.5,
    rebalance_months=2,
    transaction_fee=0.0025, # 0.25% transaction fee
    start_date=None,
    end_date=None,
)

print("Table E21: TQQQ/TMF Rebalance Strategy Data (Full Period)")
display(tqqq_tmf_full)
```

Table E21: TQQQ/TMF Rebalance Strategy Data (Full Period)

Date	TQQQ_Adj_Close	TMF_Adj_Close	TQQQ_Return	TMF_Return	\
2010-02-11	1.00	1.00	NaN	NaN	

2010-02-12	1.00	1.01	0.00	0.01
2010-02-16	1.04	1.02	0.04	0.01
2010-02-17	1.06	0.99	0.02	-0.03
2010-02-18	1.08	0.97	0.02	-0.02
...	...	...	...	...
2026-10-16	332.43	0.48	-0.08	0.00
2026-10-19	344.27	0.47	0.04	-0.02
2026-10-20	343.15	0.48	-0.00	0.02
2026-10-21	353.88	0.48	0.03	0.00
2026-10-22	323.71	0.51	-0.09	0.06

Date	TQQQ_Drawdown	TMF_Drawdown	TQQQ_Drawdown_Daily_Max	\
2010-02-11	0.00	0.00		0.00
2010-02-12	0.00	0.00		0.00
2010-02-16	0.00	0.00		0.00
2010-02-17	0.00	-0.03		0.00
2010-02-18	0.00	-0.05		0.00
...	...	...	...	...
2026-10-16	-0.21	-0.93		-0.82
2026-10-19	-0.19	-0.94		-0.82
2026-10-20	-0.19	-0.93		-0.82
2026-10-21	-0.16	-0.93		-0.82
2026-10-22	-0.24	-0.93		-0.82

Date	TMF_Drawdown_Daily_Max	TQQQ_Value	TMF_Value	Strategy_Adj_Close	\
2010-02-11	0.00	0.50	0.50	1.00	
2010-02-12	0.00	0.50	0.51	1.01	
2010-02-16	0.00	0.52	0.51	1.03	
2010-02-17	-0.03	0.53	0.49	1.02	
2010-02-18	-0.05	0.54	0.49	1.03	
...	...	...	...	...	...
2026-10-16	-0.94	15.13	18.89	34.02	
2026-10-19	-0.94	15.67	18.48	34.15	
2026-10-20	-0.94	15.61	18.93	34.54	
2026-10-21	-0.94	16.10	19.02	35.12	
2026-10-22	-0.94	14.73	20.07	34.80	

Date	Strategy_Return	Strategy_Drawdown	Strategy_Drawdown_Daily_Max	\
2010-02-11	NaN	0.00		0.00
2010-02-12	0.01	0.00		0.00
2010-02-16	0.02	0.00		0.00
2010-02-17	-0.01	-0.01		-0.01
2010-02-18	0.00	-0.01		-0.01
...	...	...	...	...
2026-10-16	-0.03	-0.46		-0.78

2026-10-19	0.00	-0.46	-0.78
2026-10-20	0.01	-0.46	-0.78
2026-10-21	0.02	-0.45	-0.78
2026-10-22	-0.01	-0.45	-0.78

Date	Rebalance	Transaction_Fee
2010-02-11	False	0.00
2010-02-12	False	0.00
2010-02-16	False	0.00
2010-02-17	False	0.00
2010-02-18	False	0.00
...	...	...
2026-10-16	False	0.00
2026-10-19	False	0.00
2026-10-20	False	0.00
2026-10-21	False	0.00
2026-10-22	False	0.00

[4200 rows x 16 columns]

```
[49]: tqqq_tmf_full_sum_stats = summary_stats(
    fund_list=["TQQQ", "TMF"],
    df=tqqq_tmf_full[["Strategy_Return"]].dropna(),
    period="Daily",
    use_calendar_days=False,
    excel_export=False,
    pickle_export=False,
    output_confirmation=False,
)

# Rename the index to indicate the strategy
tqqq_tmf_full_sum_stats.index = ["TQQQ/TMF Full Strategy"]

all_strat_sum_stats = pd.concat([all_strat_sum_stats, tqqq_tmf_full_sum_stats])

print("Table E22: Strategy Summary Statistics")
display(all_strat_sum_stats)
```

Table E22: Strategy Summary Statistics

	Number of Observations	Data Start Date	Data End Date	\
TQQQ/TMF Strategy	2679	2010-02-12	2020-10-02	
TQQQ	2679	2010-02-12	2020-10-02	
TMF	2679	2010-02-12	2020-10-02	
TQQQ/TMF BS	2679	2010-02-12	2020-10-02	
TQQQ/TMF Full Strategy	4199	2010-02-12	2026-10-22	
Annual Mean Return (Arithmetic) \				

TQQQ/TMF Strategy	0.39
TQQQ	0.57
TMF	0.26
TQQQ/TMF BS	0.38
TQQQ/TMF Full Strategy	0.27

	Annualized Volatility	Annualized Sharpe Ratio \
TQQQ/TMF Strategy	0.28	1.40
TQQQ	0.58	0.99
TMF	0.44	0.59
TQQQ/TMF BS	0.26	1.48
TQQQ/TMF Full Strategy	0.34	0.80

	CAGR (Geometric)	Daily Max Return \
TQQQ/TMF Strategy	0.42	0.11
TQQQ	0.50	0.27
TMF	0.18	0.20
TQQQ/TMF BS	0.41	0.07
TQQQ/TMF Full Strategy	0.24	0.17

	Daily Max Return (Date)	Daily Min Return \
TQQQ/TMF Strategy	2020-03-20	-0.16
TQQQ	2020-03-13	-0.34
TMF	2020-03-20	-0.18
TQQQ/TMF BS	2020-03-06	-0.09
TQQQ/TMF Full Strategy	2022-11-10	-0.16

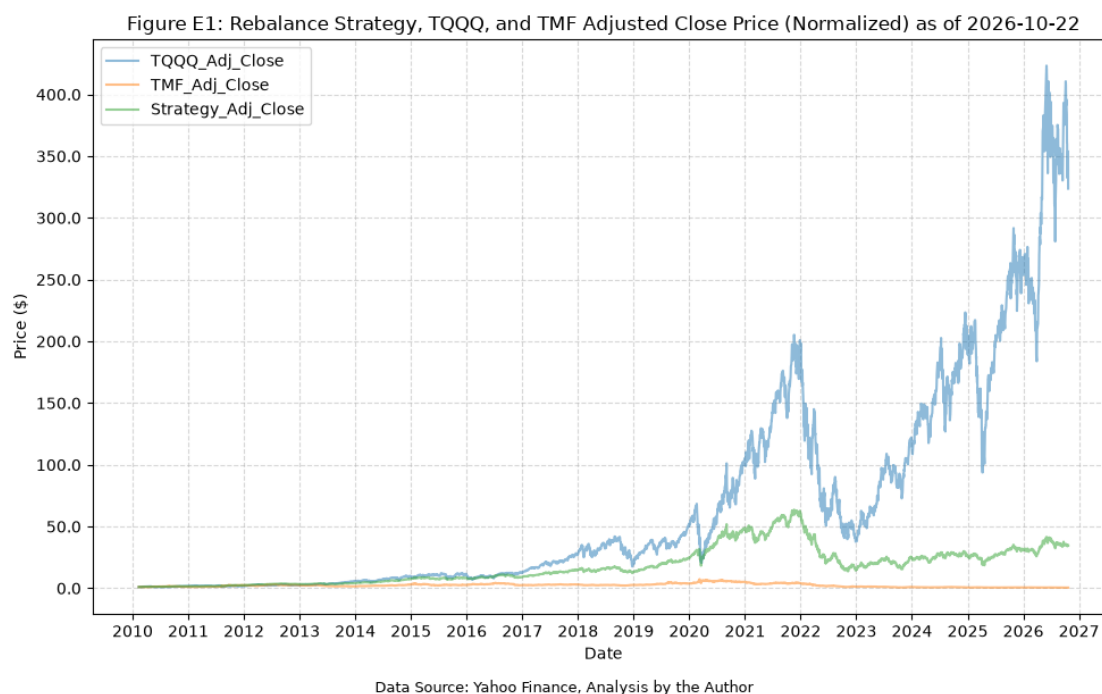
	Daily Min Return (Date)	Max Drawdown	Peak \
TQQQ/TMF Strategy	2020-03-18	-0.42	2020-03-06
TQQQ	2020-03-16	-0.70	2020-02-19
TMF	2020-03-18	-0.53	2012-07-25
TQQQ/TMF BS	2020-09-03	-0.30	2018-08-29
TQQQ/TMF Full Strategy	2020-03-18	-0.78	2021-11-19

	Trough	Recovery Date	Calendar Days to Recovery \
TQQQ/TMF Strategy	2020-03-18	2020-05-07	50.00
TQQQ	2020-03-20	2020-07-10	112.00
TMF	2013-08-21	2015-01-05	502.00
TQQQ/TMF BS	2018-12-24	2019-04-23	120.00
TQQQ/TMF Full Strategy	2022-11-09	NaT	NaN

	MAR Ratio
TQQQ/TMF Strategy	1.00
TQQQ	0.71
TMF	0.33
TQQQ/TMF BS	1.38
TQQQ/TMF Full Strategy	0.30

Here we can see that the performance has degraded over time, especially due to the significant drawdown in TMF. Next, we will look at the plots.

```
[50]: plot_time_series(
    df=tqqq_tmf_full,
    plot_start_date=None,
    plot_end_date=None,
    plot_columns=["TQQQ_Adj_Close", "TMF_Adj_Close", "Strategy_Adj_Close"],
    title=f"Figure E1: Rebalance Strategy, TQQQ, and TMF Adjusted Close Price_
↪(Normalized) as of {tqqq_tmf_full.index[-1].date()}",
    x_label="Date",
    x_format="Year",
    x_tick_spacing=1,
    x_tick_start=None,
    x_tick_rotation=0,
    y_label="Price ($)",
    y_format="Decimal",
    y_format_decimal_places=1,
    y_tick_spacing="Auto",
    y_tick_rotation=0,
    caption="Data Source: Yahoo Finance, Analysis by the Author",
)
```



We can see that the strategy peaked in late 2021, and appears to have struggled since then. This could be due to the TQQQ performance, but since 2023 TQQQ has shown significant growth.

Therefore, we suspect that TMF is actually the cause of the strategy underperformance, and we should look at its performance closely.

```
[51]: lev_etf = "TMF"
      etf = "TLT"
      end_date = None

      df = combine_normalize(
          fund_data=fund_data,
          lev_etf=lev_etf,
          etf=etf,
          start_date=None,
          end_date=end_date,
      )

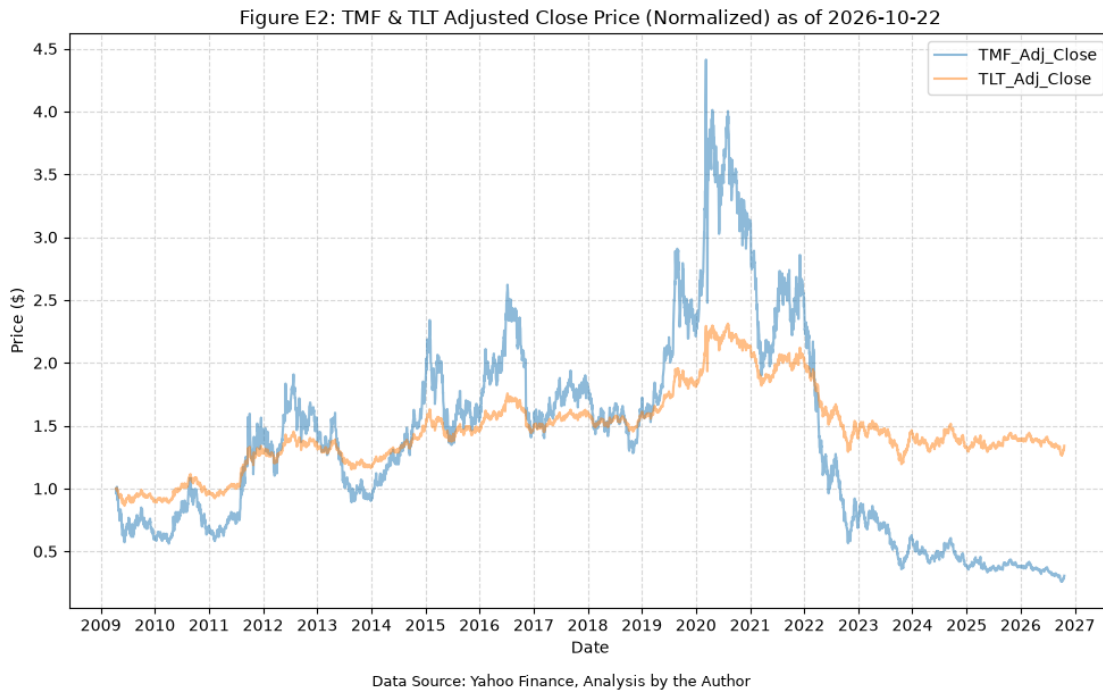
      print(f"Table E23: {lev_etf} & {etf} Normalized Data (Full Period)")
      display(df)
```

Table E23: TMF & TLT Normalized Data (Full Period)

	TMF_Adj_Close	TMF_Close	TMF_High	TMF_Low	TMF_Open	TMF_Volume	\
Date							
2009-04-16	1.00	1.00	1.00	1.00	1.00	1.00	
2009-04-17	0.96	0.96	0.97	0.95	0.98	2.64	
2009-04-20	1.01	1.01	1.00	0.99	0.98	0.60	
2009-04-21	0.98	0.98	1.03	0.99	1.03	1.37	
2009-04-22	0.95	0.95	0.97	0.94	0.98	2.22	
...	...	...	...	...	...	...	
2026-10-16	0.29	0.19	0.19	0.19	0.19	1451.62	
2026-10-19	0.28	0.19	0.19	0.19	0.19	1054.76	
2026-10-20	0.29	0.19	0.19	0.19	0.19	1008.60	
2026-10-21	0.29	0.19	0.19	0.19	0.19	910.70	
2026-10-22	0.31	0.20	0.20	0.20	0.20	2202.68	
	TLT_Adj_Close	TLT_Close	TLT_High	TLT_Low	TLT_Open	TLT_Volume	
Date							
2009-04-16	1.00	1.00	1.00	1.00	1.00	1.00	
2009-04-17	0.99	0.99	0.99	0.99	1.00	2.45	
2009-04-20	1.00	1.00	1.00	1.00	1.00	1.62	
2009-04-21	0.99	0.99	1.01	0.99	1.01	0.86	
2009-04-22	0.99	0.99	0.99	0.98	1.00	2.16	
...	...	...	...	...	...	...	
2026-10-16	1.31	0.78	0.78	0.78	0.77	40.17	
2026-10-19	1.30	0.77	0.77	0.77	0.78	29.34	
2026-10-20	1.31	0.78	0.78	0.78	0.78	26.97	
2026-10-21	1.32	0.78	0.78	0.78	0.78	34.37	
2026-10-22	1.34	0.79	0.79	0.79	0.79	62.18	

[4408 rows x 12 columns]

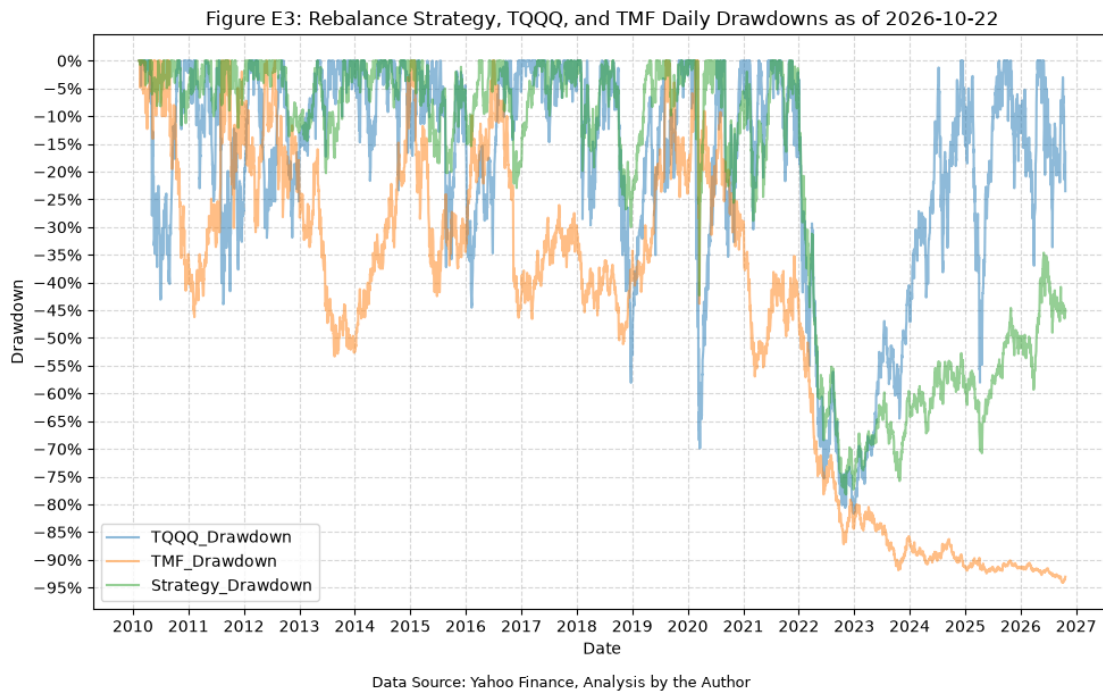
```
[52]: plot_time_series(
    df=df,
    plot_start_date=None,
    plot_end_date=None,
    plot_columns=[f"{lev_etf}_Adj_Close", f"{etf}_Adj_Close"],
    title=f"Figure E2: {lev_etf} & {etf} Adjusted Close Price (Normalized) as of {df.index[-1].date()}",
    x_label="Date",
    x_format="Year",
    x_tick_spacing=1,
    x_tick_start=None,
    x_tick_rotation=0,
    y_label="Price ($)",
    y_format="Decimal",
    y_format_decimal_places=1,
    y_tick_spacing="Auto",
    y_tick_rotation=0,
    caption="Data Source: Yahoo Finance, Analysis by the Author",
)
```



Here we have the answer. TMF has essentially collapsed since mid-2020, which explains the underperformance of the strategy despite the strong performance of TQQQ in recent years. Every time the strategy has been rebalancing, it has been selling TQQQ to buy TMF at depressed prices, which has hindered overall performance.

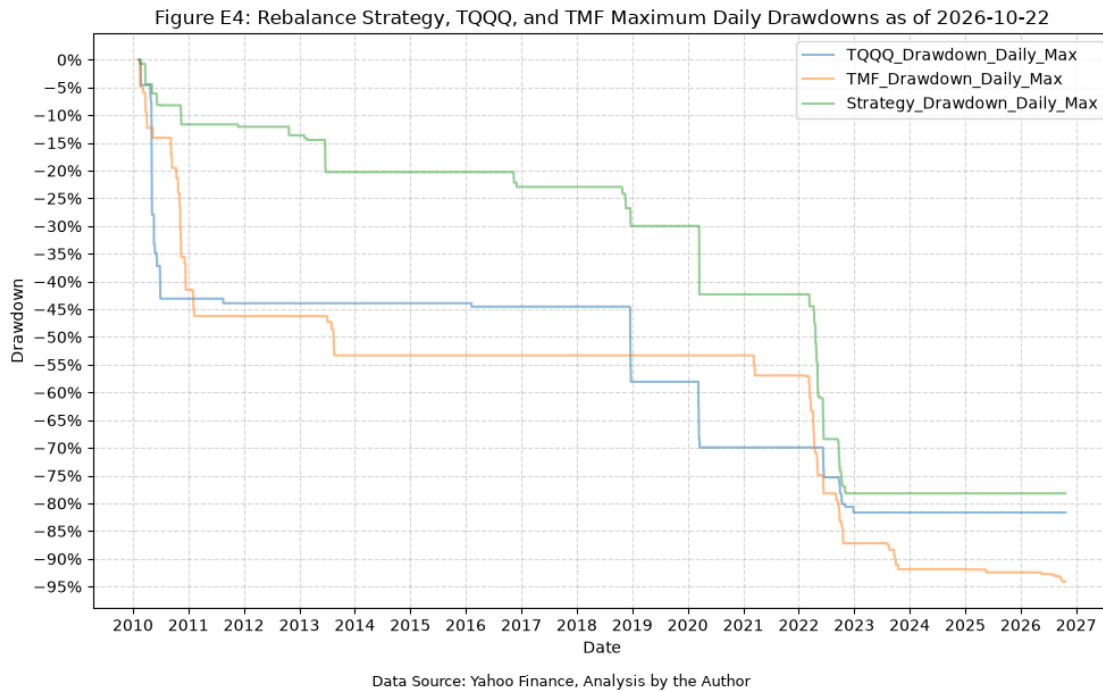
Let's look at the drawdown plots to see how the strategy has been affected by the TMF collapse.

```
[53]: plot_time_series(  
    df=tqqq_tmf_full,  
    plot_start_date=None,  
    plot_end_date=None,  
    plot_columns=["TQQQ_Drawdown", "TMF_Drawdown", "Strategy_Drawdown"],  
    title=f"Figure E3: Rebalance Strategy, TQQQ, and TMF Daily Drawdowns as of_{  
↪tqqq_tmf_full.index[-1].date()}",  
    x_label="Date",  
    x_format="Year",  
    x_tick_spacing=1,  
    x_tick_start=None,  
    x_tick_rotation=0,  
    y_label="Drawdown",  
    y_format="Percentage",  
    y_format_decimal_places=0,  
    y_tick_spacing="Auto",  
    y_tick_rotation=0,  
    caption="Data Source: Yahoo Finance, Analysis by the Author",  
)
```



Next, the maximum DDD.

```
[54]: plot_time_series(
    df=tqqq_tmf_full,
    plot_start_date=None,
    plot_end_date=None,
    plot_columns=[
        "TQQQ_Drawdown_Daily_Max",
        "TMF_Drawdown_Daily_Max",
        "Strategy_Drawdown_Daily_Max",
    ],
    title=f"Figure E4: Rebalance Strategy, TQQQ, and TMF Maximum Daily_
↳ Drawdowns as of {tqqq_tmf_full.index[-1].date()}",
    x_label="Date",
    x_format="Year",
    x_tick_spacing=1,
    x_tick_start=None,
    x_tick_rotation=0,
    y_label="Drawdown",
    y_format="Percentage",
    y_format_decimal_places=0,
    y_tick_spacing="Auto",
    y_tick_rotation=0,
    caption="Data Source: Yahoo Finance, Analysis by the Author",
)
```



To confirm the change in regime, we can look at the correlation between the daily returns of TQQQ

and TMF over time, first by calendar year, and then on a rolling 252 trading day (roughly one year) basis.

```
[55]: # Calculate the correlation between TQQQ and TMF daily returns for each
      ↪ calendar year
annual_corr = (
    tqqq_tmf_full.groupby(tqqq_tmf_full.index.year)
    .apply(lambda x: x["TQQQ_Return"].corr(x["TMF_Return"]))
    .to_frame(name="TQQQ_TMF_Correlation")
)
annual_corr.index.name = "Year"

print("Table E24: Annual Correlation of TQQQ and TMF Daily Returns")
display(annual_corr)
```

Table E24: Annual Correlation of TQQQ and TMF Daily Returns

TQQQ_TMF_Correlation	
Year	
2010	-0.55
2011	-0.62
2012	-0.56
2013	-0.20
2014	-0.41
2015	-0.33
2016	-0.29
2017	-0.16
2018	-0.23
2019	-0.41
2020	-0.39
2021	0.07
2022	0.08
2023	0.14
2024	0.00
2025	0.04
2026	0.20

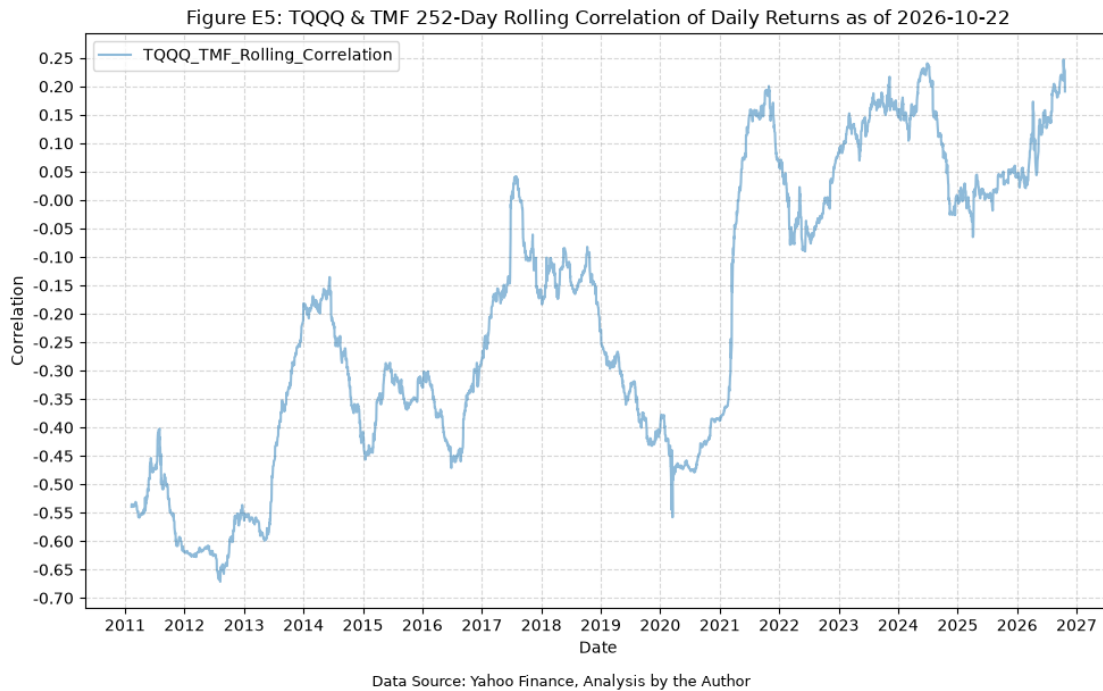
```
[56]: # Calculate the rolling 252 day correlation between TQQQ and TMF daily returns
tqqq_tmf_full["TQQQ_TMF_Rolling_Correlation"] = (
    tqqq_tmf_full["TQQQ_Return"].rolling(252).corr(tqqq_tmf_full["TMF_Return"])
)

plot_time_series(
    df=tqqq_tmf_full,
    plot_start_date=None,
    plot_end_date=None,
    plot_columns=["TQQQ_TMF_Rolling_Correlation"],
```

```

title=f"Figure E5: TQQQ & TMF 252-Day Rolling Correlation of Daily Returns_
as of {tqqq_tmf_full.index[-1].date()}",
x_label="Date",
x_format="Year",
x_tick_spacing=1,
x_tick_start=None,
x_tick_rotation=0,
y_label="Correlation",
y_format="Decimal",
y_format_decimal_places=2,
y_tick_spacing="Auto",
y_tick_rotation=0,
caption="Data Source: Yahoo Finance, Analysis by the Author",
)

```



The correlation was negative in every year from 2010 through 2020, ranging from -0.16 to -0.62. Since 2021, it has been zero or positive in every year. The rolling correlation shows the same shift. Before 2021, it was positive for only a brief period in 2017, whereas since 2021 it has been positive for the vast majority of days.

What we find here is almost the worst case scenario for long-term investing in triple leveraged exchange traded funds. For years, we have a strategy that performs very well, with relatively low risk (Sharpe ratio of 1.4), but due to a regime change in markets, and a reversal of the historic equity/bond negative correlation, the strategy experiences significant drawdowns and underperformance, highlighting the challenges and risks associated with long-term investing in such leveraged

instruments.

0.12 Extension 2: Expanding the date range for analysis (including “Black Swan”)

We can also extend the date range for the BS strategy, to see if that impacted the overall performance at all.

```
[57]: tqqq_tmf_bs_full = rebalance_strategy_bs(
      fund_data=fund_data,
      fund_1="TQQQ",
      fund_2="TMF",
      cash_fund="IEF",
      drop_threshold=0.2,
      fund_1_weight=0.5,
      rebalance_months=2,
      transaction_fee=0.0025, # 0.25% transaction fee
      start_date=None,
      end_date=None,
  )

  print("Table E25: TQQQ/TMF Black Swan Strategy Data (Full Period)")
  display(tqqq_tmf_bs_full)
```

Table E25: TQQQ/TMF Black Swan Strategy Data (Full Period)

	TQQQ_Adj_Close	TMF_Adj_Close	IEF_Adj_Close	TQQQ_Return \	
Date					
2010-02-11	1.00	1.00	1.00	NaN	
2010-02-12	1.00	1.01	1.00	0.00	
2010-02-16	1.04	1.02	1.00	0.04	
2010-02-17	1.06	0.99	1.00	0.02	
2010-02-18	1.08	0.97	1.00	0.02	
...	...	...	...	...	
2026-10-16	332.43	0.48	1.47	-0.08	
2026-10-19	344.27	0.47	1.47	0.04	
2026-10-20	343.15	0.48	1.47	-0.00	
2026-10-21	353.88	0.48	1.47	0.03	
2026-10-22	323.71	0.51	1.48	-0.09	
	TMF_Return	TQQQ_Drawdown	TMF_Drawdown	TQQQ_Drawdown_Daily_Max \	
Date					
2010-02-11	NaN	0.00	0.00	0.00	
2010-02-12	0.01	0.00	0.00	0.00	
2010-02-16	0.01	0.00	0.00	0.00	
2010-02-17	-0.03	0.00	-0.03	0.00	
2010-02-18	-0.02	0.00	-0.05	0.00	
...	...	...	...	...	
2026-10-16	0.00	-0.21	-0.93	-0.82	

2026-10-19	-0.02	-0.19	-0.94	-0.82
2026-10-20	0.02	-0.19	-0.93	-0.82
2026-10-21	0.00	-0.16	-0.93	-0.82
2026-10-22	0.06	-0.24	-0.93	-0.82

Date	TMF_Drawdown_Daily_Max	TQQQ_Value	TMF_Value	IEF_Value	\
2010-02-11	0.00	0.50	0.50	0.00	
2010-02-12	0.00	0.50	0.51	0.00	
2010-02-16	0.00	0.52	0.51	0.00	
2010-02-17	-0.03	0.53	0.49	0.00	
2010-02-18	-0.05	0.54	0.49	0.00	
...	...	...	...	...	
2026-10-16	-0.94	14.33	17.90	0.00	
2026-10-19	-0.94	14.84	17.51	0.00	
2026-10-20	-0.94	14.79	17.93	0.00	
2026-10-21	-0.94	15.25	18.01	0.00	
2026-10-22	-0.94	13.95	19.01	0.00	

Date	Strategy_Adj_Close	Strategy_Return	Strategy_Drawdown	\
2010-02-11	1.00	NaN	0.00	
2010-02-12	1.01	0.01	0.00	
2010-02-16	1.03	0.02	0.00	
2010-02-17	1.02	-0.01	-0.01	
2010-02-18	1.03	0.00	-0.01	
...	...	...	...	
2026-10-16	32.23	-0.03	-0.46	
2026-10-19	32.35	0.00	-0.46	
2026-10-20	32.72	0.01	-0.46	
2026-10-21	33.26	0.02	-0.45	
2026-10-22	32.96	-0.01	-0.45	

Date	Strategy_Drawdown_Daily_Max	Rebalance	Switch	In_IEF	\
2010-02-11	0.00	False	False	False	
2010-02-12	0.00	False	False	False	
2010-02-16	0.00	False	False	False	
2010-02-17	-0.01	False	False	False	
2010-02-18	-0.01	False	False	False	
...	...	...	...	...	
2026-10-16	-0.78	False	False	False	
2026-10-19	-0.78	False	False	False	
2026-10-20	-0.78	False	False	False	
2026-10-21	-0.78	False	False	False	
2026-10-22	-0.78	False	False	False	

Transaction_Fee

Date	
2010-02-11	0.00
2010-02-12	0.00
2010-02-16	0.00
2010-02-17	0.00
2010-02-18	0.00
...	...
2026-10-16	0.00
2026-10-19	0.00
2026-10-20	0.00
2026-10-21	0.00
2026-10-22	0.00

[4200 rows x 20 columns]

```
[58]: tqqq_tmf_bs_full_sum_stats = summary_stats(
    fund_list=["TQQQ", "TMF"],
    df=tqqq_tmf_bs_full[["Strategy_Return"]].dropna(),
    period="Daily",
    use_calendar_days=False,
    excel_export=False,
    pickle_export=False,
    output_confirmation=False,
)

# Rename the index to indicate the strategy
tqqq_tmf_bs_full_sum_stats.index = ["TQQQ/TMF Full BS"]

all_strat_sum_stats = pd.concat([all_strat_sum_stats,
    ↪tqqq_tmf_bs_full_sum_stats])

print("Table E26: Strategy Summary Statistics")
display(all_strat_sum_stats)
```

Table E26: Strategy Summary Statistics

	Number of Observations	Data Start Date	Data End Date	\
TQQQ/TMF Strategy	2679	2010-02-12	2020-10-02	
TQQQ	2679	2010-02-12	2020-10-02	
TMF	2679	2010-02-12	2020-10-02	
TQQQ/TMF BS	2679	2010-02-12	2020-10-02	
TQQQ/TMF Full Strategy	4199	2010-02-12	2026-10-22	
TQQQ/TMF Full BS	4199	2010-02-12	2026-10-22	
	Annual Mean Return (Arithmetic) \			
TQQQ/TMF Strategy		0.39		
TQQQ		0.57		
TMF		0.26		

TQQQ/TMF BS	0.38
TQQQ/TMF Full Strategy	0.27
TQQQ/TMF Full BS	0.26

	Annualized Volatility	Annualized Sharpe Ratio \
TQQQ/TMF Strategy	0.28	1.40
TQQQ	0.58	0.99
TMF	0.44	0.59
TQQQ/TMF BS	0.26	1.48
TQQQ/TMF Full Strategy	0.34	0.80
TQQQ/TMF Full BS	0.32	0.81

	CAGR (Geometric)	Daily Max Return \
TQQQ/TMF Strategy	0.42	0.11
TQQQ	0.50	0.27
TMF	0.18	0.20
TQQQ/TMF BS	0.41	0.07
TQQQ/TMF Full Strategy	0.24	0.17
TQQQ/TMF Full BS	0.23	0.17

	Daily Max Return (Date)	Daily Min Return \
TQQQ/TMF Strategy	2020-03-20	-0.16
TQQQ	2020-03-13	-0.34
TMF	2020-03-20	-0.18
TQQQ/TMF BS	2020-03-06	-0.09
TQQQ/TMF Full Strategy	2022-11-10	-0.16
TQQQ/TMF Full BS	2022-11-10	-0.12

	Daily Min Return (Date)	Max Drawdown	Peak \
TQQQ/TMF Strategy	2020-03-18	-0.42	2020-03-06
TQQQ	2020-03-16	-0.70	2020-02-19
TMF	2020-03-18	-0.53	2012-07-25
TQQQ/TMF BS	2020-09-03	-0.30	2018-08-29
TQQQ/TMF Full Strategy	2020-03-18	-0.78	2021-11-19
TQQQ/TMF Full BS	2022-05-05	-0.78	2021-11-19

	Trough	Recovery Date	Calendar Days to Recovery \
TQQQ/TMF Strategy	2020-03-18	2020-05-07	50.00
TQQQ	2020-03-20	2020-07-10	112.00
TMF	2013-08-21	2015-01-05	502.00
TQQQ/TMF BS	2018-12-24	2019-04-23	120.00
TQQQ/TMF Full Strategy	2022-11-09	NaT	NaN
TQQQ/TMF Full BS	2022-11-09	NaT	NaN

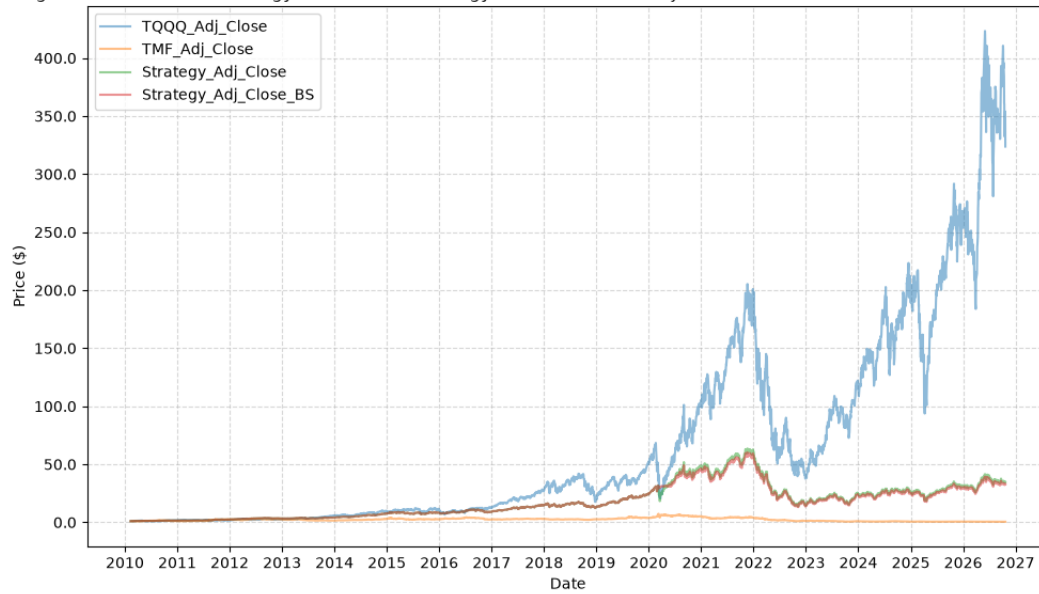
	MAR Ratio
TQQQ/TMF Strategy	1.00
TQQQ	0.71
TMF	0.33

TQQQ/TMF BS	1.38
TQQQ/TMF Full Strategy	0.30
TQQQ/TMF Full BS	0.30

```
[59]: tqqq_tmf_full_both = tqqq_tmf_full.merge(
      tqqq_tmf_bs_full, on="Date", how="outer", suffixes=("", "_BS")
    )

plot_time_series(
    df=tqqq_tmf_full_both,
    plot_start_date=None,
    plot_end_date=None,
    plot_columns=[
        "TQQQ_Adj_Close",
        "TMF_Adj_Close",
        "Strategy_Adj_Close",
        "Strategy_Adj_Close_BS",
    ],
    title=f"Figure E6: Rebalance Strategy, Black Swan Strategy, TQQQ, and TMF_
↪Adjusted Close Price (Normalized) as of {tqqq_tmf_full_both.index[-1]}.
↪date()}",
    x_label="Date",
    x_format="Year",
    x_tick_spacing=1,
    x_tick_start=None,
    x_tick_rotation=0,
    y_label="Price ($)",
    y_format="Decimal",
    y_format_decimal_places=1,
    y_tick_spacing="Auto",
    y_tick_rotation=0,
    caption="Data Source: Yahoo Finance, Analysis by the Author",
)
```

Figure E6: Rebalance Strategy, Black Swan Strategy, TQQQ, and TMF Adjusted Close Price (Normalized) as of 2026-10-22



Data Source: Yahoo Finance, Analysis by the Author

From the summary stats and the plot, we can see that the BS strategy did not materially change the performance. It might have had a risk-mitigation effect during COVID, but ultimately is not designed to handle nor is effective during long prolonged drawdowns.

0.13 Extension 3: Using UPRO and AGG

For our third extension, we'll examine the performance of a portfolio consisting of UPRO and AGG, to see how it compares to the previous strategies.

One might say that UPRO is a more "well rounded" fund than TQQQ, in that it includes not only tech but other sectors of the market as well. This diversification could potentially reduce the overall volatility of the portfolio while still providing leveraged exposure to the broader market.

Similarly, AGG (the iShares Core U.S. Aggregate Bond ETF) provides exposure to the bond market, which tends to have a low correlation with equities. Using a non-leveraged fund will likely reduce the performance, but it might also help stabilize the volatility and provide a smoother overall return profile.

To compare with the TQQQ/TMF strategy over the same period, we'll use the same start date of 2/11/2010, even though UPRO has data back to June 2009.

First, we need to pull data for AGG.

```
[60]: yf_pull_data(
    base_directory=DATA_DIR,
    ticker="AGG",
    adjusted=False,
    source="Yahoo_Finance",
```

```

asset_class="Exchange_Traded_Funds",
excel_export=True,
pickle_export=True,
output_confirmation=False,
)

```

```

[60]:
Adj Close  Close  High  Low  Open  Volume
Date
2003-09-29    48.74 102.17 102.30 102.10 102.29    13600
2003-09-30    49.00 102.70 102.74 102.29 102.30    62600
2003-10-01    48.97 102.65 102.75 102.60 102.64    66300
2003-10-02    48.90 102.49 102.65 102.01 102.20    68900
2003-10-03    48.54 101.75 102.05 101.70 102.05    64500
...
2026-10-16    95.38  95.38  95.58  95.22  95.29  11389000
2026-10-19    95.11  95.11  95.39  95.04  95.34  10400500
2026-10-20    95.27  95.27  95.51  95.16  95.31   9952700
2026-10-21    95.30  95.30  95.50  95.11  95.40   9833700
2026-10-22    96.12  96.12  96.23  95.62  95.69  10825400

```

[5804 rows x 6 columns]

Then append it to the existing dictionary.

```

[61]: fund_list.append("AGG")

# Load the data for each ETF, rename the columns to include the ticker, store_
↳ it in the fund_data dictionary
for fund in fund_list:
    data = load_data(
        base_directory=DATA_DIR,
        ticker=fund,
        source="Yahoo_Finance",
        asset_class="Exchange_Traded_Funds",
        timeframe="Daily",
        file_format="pickle",
    )

    data = data.rename(
        columns={
            "Adj Close": f"{fund}_Adj_Close",
            "Close": f"{fund}_Close",
            "High": f"{fund}_High",
            "Low": f"{fund}_Low",
            "Open": f"{fund}_Open",
            "Volume": f"{fund}_Volume",
        }
    )

```

```
fund_data[fund] = data
```

Now we can run the strategy.

```
[62]: upro_agg_full = rebalance_strategy(
    fund_data=fund_data,
    fund_1="UPRO",
    fund_2="AGG",
    fund_1_weight=0.5,
    rebalance_months=2,
    transaction_fee=0.0025, # 0.25% transaction fee
    start_date="2010-02-11",
    end_date=None,
)

print("Table E27: UPRO/AGG Rebalance Strategy Data")
display(upro_agg_full)
```

Table E27: UPRO/AGG Rebalance Strategy Data

	UPRO_Adj_Close	AGG_Adj_Close	UPRO_Return	AGG_Return	\
Date					
2010-02-11	1.00	1.00	NaN	NaN	
2010-02-12	1.00	1.00	-0.00	0.00	
2010-02-16	1.04	1.00	0.05	0.00	
2010-02-17	1.06	1.00	0.01	-0.00	
2010-02-18	1.08	1.00	0.02	-0.00	
...	...	...	...	...	
2026-10-16	82.92	1.47	-0.03	-0.00	
2026-10-19	83.96	1.46	0.01	-0.00	
2026-10-20	84.70	1.47	0.01	0.00	
2026-10-21	86.08	1.47	0.02	0.00	
2026-10-22	82.98	1.48	-0.04	0.01	

	UPRO_Drawdown	AGG_Drawdown	UPRO_Drawdown_Daily_Max	\
Date				
2010-02-11	0.00	0.00		0.00
2010-02-12	-0.00	0.00		-0.00
2010-02-16	0.00	0.00		-0.00
2010-02-17	0.00	-0.00		-0.00
2010-02-18	0.00	-0.00		-0.00
...	...	...	...	
2026-10-16	-0.07	-0.04		-0.77
2026-10-19	-0.06	-0.04		-0.77
2026-10-20	-0.05	-0.04		-0.77
2026-10-21	-0.03	-0.04		-0.77
2026-10-22	-0.07	-0.03		-0.77

	AGG_Drawdown_Daily_Max	UPRO_Value	AGG_Value	Strategy_Adj_Close	\
Date					
2010-02-11	0.00	0.50	0.50	1.00	
2010-02-12	0.00	0.50	0.50	1.00	
2010-02-16	0.00	0.52	0.50	1.02	
2010-02-17	-0.00	0.53	0.50	1.03	
2010-02-18	-0.00	0.54	0.50	1.04	
...	...	...	...	...	
2026-10-16	-0.18	8.44	8.58	17.03	
2026-10-19	-0.18	8.55	8.56	17.11	
2026-10-20	-0.18	8.63	8.57	17.20	
2026-10-21	-0.18	8.77	8.57	17.34	
2026-10-22	-0.18	8.45	8.65	17.10	

	Strategy_Return	Strategy_Drawdown	Strategy_Drawdown_Daily_Max	\
Date				
2010-02-11	NaN	0.00	0.00	
2010-02-12	-0.00	-0.00	-0.00	
2010-02-16	0.03	0.00	-0.00	
2010-02-17	0.01	0.00	-0.00	
2010-02-18	0.01	0.00	-0.00	
...	...	...	...	
2026-10-16	-0.02	-0.04	-0.43	
2026-10-19	0.00	-0.04	-0.43	
2026-10-20	0.01	-0.03	-0.43	
2026-10-21	0.01	-0.03	-0.43	
2026-10-22	-0.01	-0.04	-0.43	

	Rebalance	Transaction_Fee
Date		
2010-02-11	False	0.00
2010-02-12	False	0.00
2010-02-16	False	0.00
2010-02-17	False	0.00
2010-02-18	False	0.00
...	...	...
2026-10-16	False	0.00
2026-10-19	False	0.00
2026-10-20	False	0.00
2026-10-21	False	0.00
2026-10-22	False	0.00

[4200 rows x 16 columns]

```
[63]: upro_agg_full_sum_stats = summary_stats(
        fund_list=["UPRO", "AGG"],
        df=upro_agg_full[["Strategy_Return"]].dropna(),
```

```

period="Daily",
use_calendar_days=False,
excel_export=False,
pickle_export=False,
output_confirmation=False,
)

# Rename the index to indicate the strategy
upro_agg_full_sum_stats.index = ["UPRO/AGG Full Strategy"]

all_strat_sum_stats = pd.concat([all_strat_sum_stats, upro_agg_full_sum_stats])

print("Table E28: Strategy Summary Statistics")
display(all_strat_sum_stats)

```

Table E28: Strategy Summary Statistics

	Number of Observations	Data Start Date	Data End Date	\
TQQQ/TMF Strategy	2679	2010-02-12	2020-10-02	
TQQQ	2679	2010-02-12	2020-10-02	
TMF	2679	2010-02-12	2020-10-02	
TQQQ/TMF BS	2679	2010-02-12	2020-10-02	
TQQQ/TMF Full Strategy	4199	2010-02-12	2026-10-22	
TQQQ/TMF Full BS	4199	2010-02-12	2026-10-22	
UPRO/AGG Full Strategy	4199	2010-02-12	2026-10-22	

	Annual Mean Return (Arithmetic)	\
TQQQ/TMF Strategy	0.39	
TQQQ	0.57	
TMF	0.26	
TQQQ/TMF BS	0.38	
TQQQ/TMF Full Strategy	0.27	
TQQQ/TMF Full BS	0.26	
UPRO/AGG Full Strategy	0.20	

	Annualized Volatility	Annualized Sharpe Ratio	\
TQQQ/TMF Strategy	0.28	1.40	
TQQQ	0.58	0.99	
TMF	0.44	0.59	
TQQQ/TMF BS	0.26	1.48	
TQQQ/TMF Full Strategy	0.34	0.80	
TQQQ/TMF Full BS	0.32	0.81	
UPRO/AGG Full Strategy	0.24	0.83	

	CAGR (Geometric)	Daily Max Return	\
TQQQ/TMF Strategy	0.42	0.11	
TQQQ	0.50	0.27	
TMF	0.18	0.20	

TQQQ/TMF BS	0.41	0.07
TQQQ/TMF Full Strategy	0.24	0.17
TQQQ/TMF Full BS	0.23	0.17
UPRO/AGG Full Strategy	0.19	0.12

	Daily Max Return (Date)	Daily Min Return \
TQQQ/TMF Strategy	2020-03-20	-0.16
TQQQ	2020-03-13	-0.34
TMF	2020-03-20	-0.18
TQQQ/TMF BS	2020-03-06	-0.09
TQQQ/TMF Full Strategy	2022-11-10	-0.16
TQQQ/TMF Full BS	2022-11-10	-0.12
UPRO/AGG Full Strategy	2025-04-09	-0.13

	Daily Min Return (Date)	Max Drawdown	Peak \
TQQQ/TMF Strategy	2020-03-18	-0.42	2020-03-06
TQQQ	2020-03-16	-0.70	2020-02-19
TMF	2020-03-18	-0.53	2012-07-25
TQQQ/TMF BS	2020-09-03	-0.30	2018-08-29
TQQQ/TMF Full Strategy	2020-03-18	-0.78	2021-11-19
TQQQ/TMF Full BS	2022-05-05	-0.78	2021-11-19
UPRO/AGG Full Strategy	2020-03-12	-0.43	2021-12-27

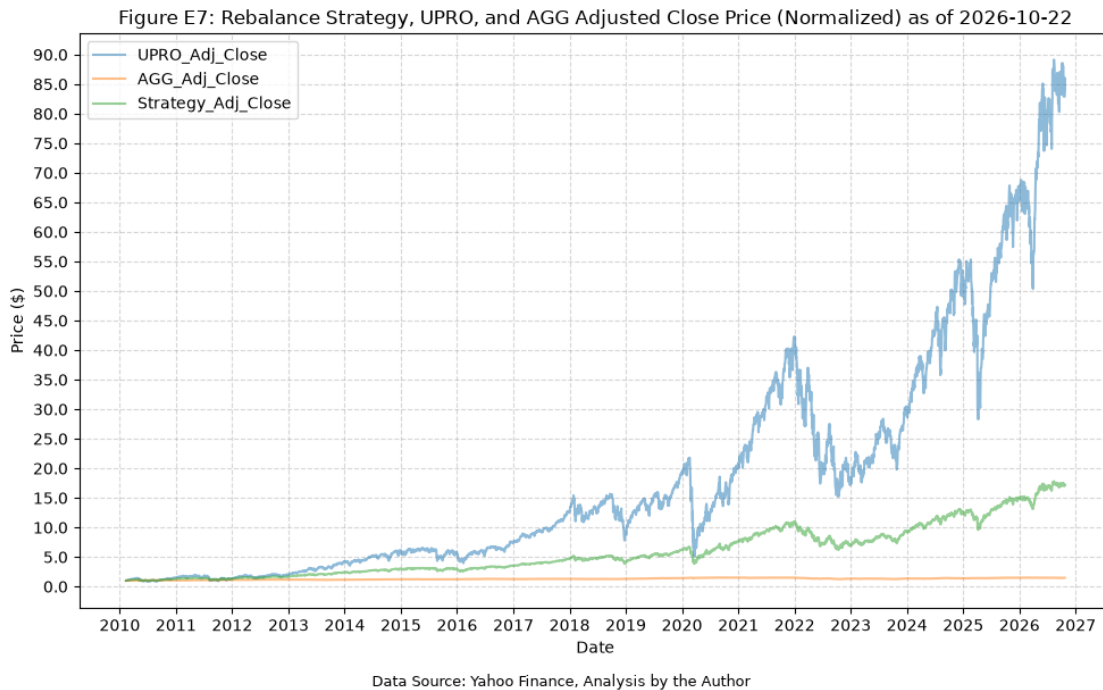
	Trough	Recovery Date	Calendar Days to Recovery \
TQQQ/TMF Strategy	2020-03-18	2020-05-07	50.00
TQQQ	2020-03-20	2020-07-10	112.00
TMF	2013-08-21	2015-01-05	502.00
TQQQ/TMF BS	2018-12-24	2019-04-23	120.00
TQQQ/TMF Full Strategy	2022-11-09	NaT	NaN
TQQQ/TMF Full BS	2022-11-09	NaT	NaN
UPRO/AGG Full Strategy	2022-10-14	2024-06-12	607.00

	MAR Ratio
TQQQ/TMF Strategy	1.00
TQQQ	0.71
TMF	0.33
TQQQ/TMF BS	1.38
TQQQ/TMF Full Strategy	0.30
TQQQ/TMF Full BS	0.30
UPRO/AGG Full Strategy	0.43

Comparing the last 3 lines from above gives us an interesting result. Our CAGR for the UPRO/AGG strategy is lower than that of the TQQQ/TMF strategy, but the volatility and draw-downs are both significantly reduced, leading to a more stable overall performance, and an improved MAR ratio.

And now the plots for the UPRO/AGG strategy.

```
[64]: plot_time_series(
    df=upro_agg_full,
    plot_start_date=None,
    plot_end_date=None,
    plot_columns=["UPRO_Adj_Close", "AGG_Adj_Close", "Strategy_Adj_Close"],
    title=f"Figure E7: Rebalance Strategy, UPRO, and AGG Adjusted Close Price_
↪(Normalized) as of {upro_agg_full.index[-1].date()}",
    x_label="Date",
    x_format="Year",
    x_tick_spacing=1,
    x_tick_start=None,
    x_tick_rotation=0,
    y_label="Price ($)",
    y_format="Decimal",
    y_format_decimal_places=1,
    y_tick_spacing="Auto",
    y_tick_rotation=0,
    caption="Data Source: Yahoo Finance, Analysis by the Author",
)
```

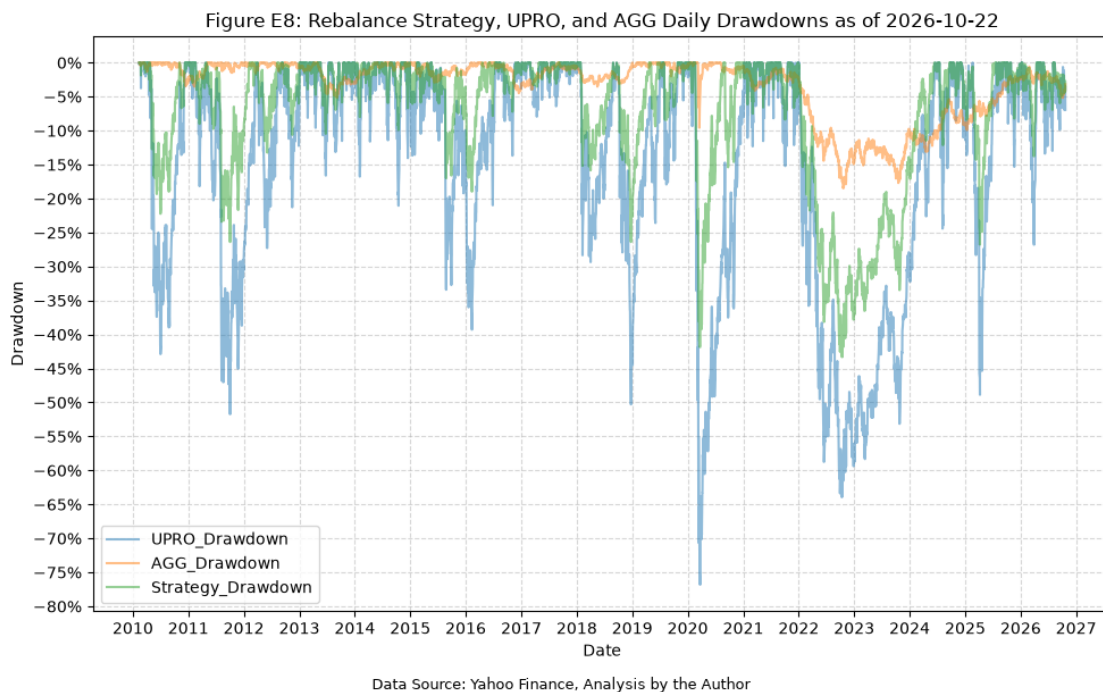


```
[65]: plot_time_series(
    df=upro_agg_full,
    plot_start_date=None,
    plot_end_date=None,
```

```

plot_columns=["UPRO_Drawdown", "AGG_Drawdown", "Strategy_Drawdown"],
title=f"Figure E8: Rebalance Strategy, UPRO, and AGG Daily Drawdowns as of_{upro_agg_full.index[-1].date()}",
x_label="Date",
x_format="Year",
x_tick_spacing=1,
x_tick_start=None,
x_tick_rotation=0,
y_label="Drawdown",
y_format="Percentage",
y_format_decimal_places=0,
y_tick_spacing="Auto",
y_tick_rotation=0,
caption="Data Source: Yahoo Finance, Analysis by the Author",
)

```



```

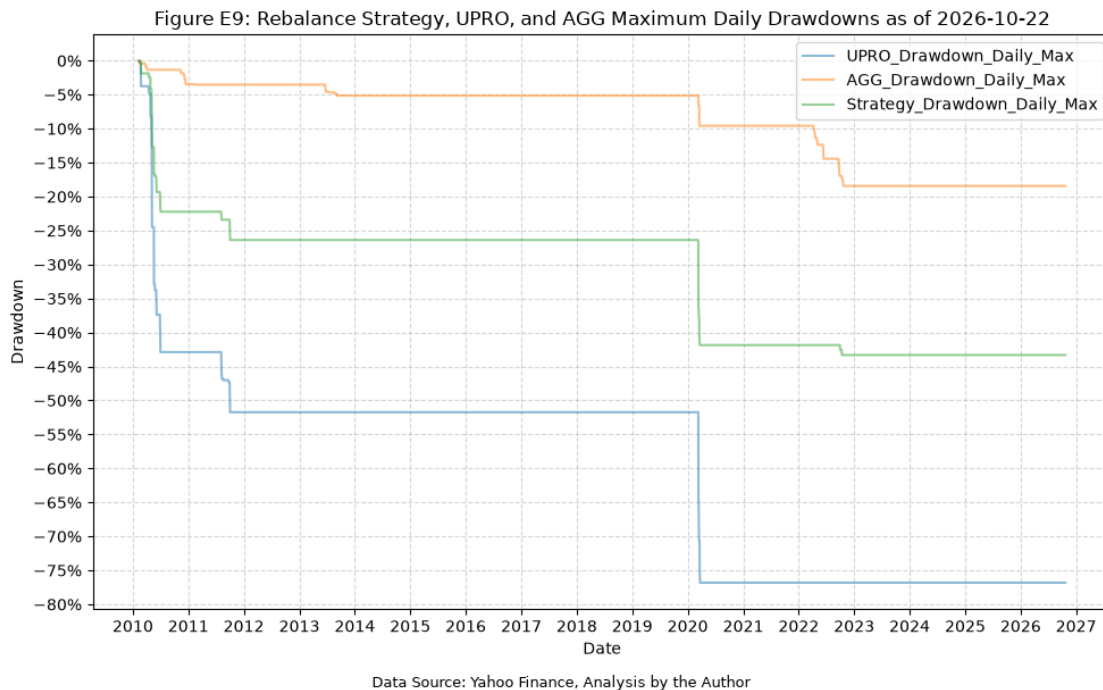
[66]: plot_time_series(
    df=upro_agg_full,
    plot_start_date=None,
    plot_end_date=None,
    plot_columns=[
        "UPRO_Drawdown_Daily_Max",
        "AGG_Drawdown_Daily_Max",
        "Strategy_Drawdown_Daily_Max",
    ],
)

```

```

],
title=f"Figure E9: Rebalance Strategy, UPRO, and AGG Maximum Daily_
↳Drawdowns as of {upro_agg_full.index[-1].date()}",
x_label="Date",
x_format="Year",
x_tick_spacing=1,
x_tick_start=None,
x_tick_rotation=0,
y_label="Drawdown",
y_format="Percentage",
y_format_decimal_places=0,
y_tick_spacing="Auto",
y_tick_rotation=0,
caption="Data Source: Yahoo Finance, Analysis by the Author",
)

```



0.14 Extension 4: Extrapolating UPRO Back To 1993

UPRO only has data back to its inception on 6/25/2009, which means that all of the results above come from a period that excludes both the dot-com crash and the 2008 financial crisis. To see how the strategy might have performed through those periods, we will extrapolate the returns for UPRO back to the start of SPY in January 1993, using a regression of the daily returns of UPRO on those of SPY over the period where both funds have data. Before the inception of UPRO we use the extrapolated data, and after its inception we use the actual data.

For the bond fund, we will use VBMFX (the Vanguard Total Bond Market Index Fund), which tracks the same index as AGG but has data back to 1986. We will then run the strategy on both UPRO/VBMFX and SPY/VBMFX from 1993 to present.

First, we need to pull data for VBMFX.

```
[67]: yf_pull_data(
    base_directory=DATA_DIR,
    ticker="VBMFX",
    adjusted=False,
    source="Yahoo_Finance",
    asset_class="Mutual_Funds",
    excel_export=True,
    pickle_export=True,
    output_confirmation=False,
)
```

```
[67]:
```

	Adj Close	Close	High	Low	Open	Volume
Date						
1986-12-11	1.38	10.00	10.00	10.00	10.00	0
1986-12-12	1.38	10.00	10.00	10.00	10.00	0
1986-12-15	1.37	9.97	9.97	9.97	9.97	0
1986-12-16	1.38	10.00	10.00	10.00	10.00	0
1986-12-17	1.38	10.00	10.00	10.00	10.00	0
...	...	...	...	...	...	...
2026-10-16	9.32	9.32	9.32	9.32	9.32	0
2026-10-19	9.29	9.29	9.29	9.29	9.29	0
2026-10-20	9.30	9.30	9.30	9.30	9.30	0
2026-10-21	9.30	9.30	9.30	9.30	9.30	0
2026-10-22	9.38	9.38	9.38	9.38	9.38	0

[10042 rows x 6 columns]

Then add it to the existing dictionary. Since VBMFX is a mutual fund rather than an ETF, we load it separately.

```
[68]: data = load_data(
    base_directory=DATA_DIR,
    ticker="VBMFX",
    source="Yahoo_Finance",
    asset_class="Mutual_Funds",
    timeframe="Daily",
    file_format="pickle",
)

data = data.rename(
    columns={
        "Adj Close": "VBMFX_Adj_Close",
        "Close": "VBMFX_Close",
    })
```

```

        "High": "VBMFX_High",
        "Low": "VBMFX_Low",
        "Open": "VBMFX_Open",
        "Volume": "VBMFX_Volume",
    }
)

fund_data["VBMFX"] = data

```

Next, we run a regression of the daily returns of UPRO on the daily returns of SPY, over the period where both funds have data. The slope (beta) is the leverage that UPRO actually delivers, and the constant (alpha) captures the daily costs of the fund, including financing and fees.

```

[69]: # Merge UPRO and SPY over the period where both funds have data
upro_spy = fund_data["UPRO"][["UPRO_Adj_Close"]].merge(
    fund_data["SPY"][["SPY_Adj_Close"]], left_index=True, right_index=True
)
upro_spy["UPRO_Return"] = upro_spy["UPRO_Adj_Close"].pct_change()
upro_spy["SPY_Return"] = upro_spy["SPY_Adj_Close"].pct_change()
upro_spy = upro_spy.dropna()

upro_spy_model = run_regression(
    df=upro_spy,
    x_plot_column="SPY_Return",
    y_plot_column="UPRO_Return",
    regression_model="OLS-statsmodels",
    regression_constant=True,
)

alpha = upro_spy_model.params.iloc[0]
beta = upro_spy_model.params.iloc[1]

print("Table E29: Regression of UPRO Daily Returns on SPY Daily Returns")
display(upro_spy_model.summary())

```

Table E29: Regression of UPRO Daily Returns on SPY Daily Returns

Dep. Variable:	UPRO_Return	R-squared:	0.997
Model:	OLS	Adj. R-squared:	0.997
Method:	Least Squares	F-statistic:	1.257e+06
Date:	Sat, 24 Oct 2026	Prob (F-statistic):	0.00
Time:	18:28:23	Log-Likelihood:	21156.
No. Observations:	4358	AIC:	-4.231e+04
Df Residuals:	4356	BIC:	-4.230e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P> t	[0.025	0.975]
const	-0.0002	2.86e-05	-6.754	0.000	-0.000	-0.000
SPY_Return	2.9894	0.003	1121.375	0.000	2.984	2.995
Omnibus:	2146.529		Durbin-Watson:	2.969		
Prob(Omnibus):	0.000		Jarque-Bera (JB):	3412005.980		
Skew:	-0.729		Prob(JB):	0.00		
Kurtosis:	140.070		Cond. No.	93.3		

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Before extrapolating, we can check how well the regression fits the actual UPRO data. Over the period where both funds have data, we compare the CAGR of the actual UPRO returns with the CAGR of the returns constructed from SPY, using only the beta, and using both the alpha and the beta.

```
[70]: # Compare the actual UPRO returns with the returns constructed from SPY
fit_check = pd.DataFrame(
    {
        "Actual UPRO": upro_spy["UPRO_Return"],
        "Beta x SPY": beta * upro_spy["SPY_Return"],
        "Alpha + Beta x SPY": alpha + beta * upro_spy["SPY_Return"],
    }
)

years = len(fit_check) / 252
fit_check_cagr = ((1 + fit_check).prod() ** (1 / years) - 1).
    ↪to_frame(name="CAGR")

print("Table E30: CAGR of Actual UPRO vs Returns Constructed From SPY")
display(fit_check_cagr)
```

Table E30: CAGR of Actual UPRO vs Returns Constructed From SPY

	CAGR
Actual UPRO	0.33
Beta x SPY	0.39
Alpha + Beta x SPY	0.33

The table above tells us that we have an adequate fit for the regression.

Next, we use the alpha and beta to extrapolate the UPRO returns back to the start of SPY, and then walk the price backwards from the first actual price of UPRO. Here is the data around the inception of UPRO after the extrapolation:

```
[71]: # Merge SPY with UPRO over the full history of SPY
upro_extrap = fund_data["SPY"][["SPY_Adj_Close"]].merge(
    fund_data["UPRO"][["UPRO_Adj_Close"]], left_index=True, right_index=True,
    ↪how="left"
)
```

```

upro_extrap["SPY_Return"] = upro_extrap["SPY_Adj_Close"].pct_change()
upro_extrap["UPRO_Return"] = upro_extrap["UPRO_Adj_Close"].pct_change()

# Before the inception of UPRO, fill the returns using the regression
# coefficients
upro_extrap["UPRO_Return"] = upro_extrap["UPRO_Return"].fillna(
    alpha + beta * upro_extrap["SPY_Return"]
)

# Walk backwards from the first actual price of UPRO: Price_{t} = Price_{t+1} /
# (1 + Return_{t+1})
first_valid_idx = upro_extrap["UPRO_Adj_Close"].first_valid_index()
splice_loc = upro_extrap.index.get_loc(first_valid_idx)
for i in range(splice_loc - 1, -1, -1):
    current_return = upro_extrap.iloc[i + 1]["UPRO_Return"]
    next_price = upro_extrap.iloc[i + 1]["UPRO_Adj_Close"]
    upro_extrap.loc[upro_extrap.index[i], "UPRO_Adj_Close"] = next_price / (
        1 + current_return
    )

# Store as a separate fund so the actual UPRO data is unchanged
fund_data["UPRO_Extrap"] = upro_extrap[["UPRO_Adj_Close"]].rename(
    columns={"UPRO_Adj_Close": "UPRO_Extrap_Adj_Close"}
)

print("Table E31: Extrapolated UPRO Data Around the Inception of UPRO")
display(upro_extrap.iloc[splice_loc - 3 : splice_loc + 3])

```

Table E31: Extrapolated UPRO Data Around the Inception of UPRO

Date	SPY_Adj_Close	UPRO_Adj_Close	SPY_Return	UPRO_Return
2009-06-22	65.80	1.03	-0.03	-0.09
2009-06-23	65.85	1.03	0.00	0.00
2009-06-24	66.41	1.06	0.01	0.03
2009-06-25	67.86	1.13	0.02	0.06
2009-06-26	67.68	1.12	-0.00	-0.01
2009-06-29	68.32	1.15	0.01	0.03

And now we can run the strategy for both UPRO/VBMFX and SPY/VBMFX, from 1993 to present.

```

[72]: upro_extrap_vbmfx_full = rebalance_strategy(
    fund_data=fund_data,
    fund_1="UPRO_Extrap",
    fund_2="VBMFX",
    fund_1_weight=0.5,
    rebalance_months=2,

```

```

transaction_fee=0.0025, # 0.25% transaction fee
start_date=None,
end_date=None,
)

print("Table E32: UPRO (Extrap)/VBMFX Rebalance Strategy Data")
display(upro_extrap_vbmfx_full)

spy_vbmfx_full = rebalance_strategy(
    fund_data=fund_data,
    fund_1="SPY",
    fund_2="VBMFX",
    fund_1_weight=0.5,
    rebalance_months=2,
    transaction_fee=0.0025, # 0.25% transaction fee
    start_date=None,
    end_date=None,
)

print("Table E33: SPY/VBMFX Rebalance Strategy Data")
display(spy_vbmfx_full)

```

Table E32: UPRO (Extrap)/VBMFX Rebalance Strategy Data

Date	UPRO_Extrap_Adj_Close	VBMFX_Adj_Close	UPRO_Extrap_Return \
1993-01-29	1.00	1.00	NaN
1993-02-01	1.02	1.00	0.02
1993-02-02	1.03	1.00	0.01
1993-02-03	1.06	1.00	0.03
1993-02-04	1.07	1.00	0.01
...	...	...	...
2026-10-16	183.52	4.00	-0.03
2026-10-19	185.82	3.99	0.01
2026-10-20	187.46	4.00	0.01
2026-10-21	190.51	4.00	0.02
2026-10-22	183.64	4.03	-0.04

Date	VBMFX_Return	UPRO_Extrap_Drawdown	VBMFX_Drawdown \
1993-01-29	NaN	0.00	0.00
1993-02-01	0.00	0.00	0.00
1993-02-02	-0.00	0.00	-0.00
1993-02-03	0.00	0.00	-0.00
1993-02-04	0.00	0.00	0.00
...	...	...	...
2026-10-16	0.00	-0.07	-0.05
2026-10-19	-0.00	-0.06	-0.05

2026-10-20	0.00	-0.05	-0.05
2026-10-21	0.00	-0.03	-0.05
2026-10-22	0.01	-0.07	-0.04

Date	UPRO_Extrap_Drawdown_Daily_Max	VBMFX_Drawdown_Daily_Max	\
1993-01-29	0.00	0.00	
1993-02-01	0.00	0.00	
1993-02-02	0.00	-0.00	
1993-02-03	0.00	-0.00	
1993-02-04	0.00	-0.00	
...	...	...	
2026-10-16	-0.98	-0.19	
2026-10-19	-0.98	-0.19	
2026-10-20	-0.98	-0.19	
2026-10-21	-0.98	-0.19	
2026-10-22	-0.98	-0.19	

Date	UPRO_Extrap_Value	VBMFX_Value	Strategy_Adj_Close	\
1993-01-29	0.50	0.50	1.00	
1993-02-01	0.51	0.50	1.01	
1993-02-02	0.51	0.50	1.01	
1993-02-03	0.53	0.50	1.03	
1993-02-04	0.54	0.50	1.04	
...	...	...	...	
2026-10-16	31.96	32.27	64.23	
2026-10-19	32.37	32.17	64.53	
2026-10-20	32.65	32.20	64.85	
2026-10-21	33.18	32.20	65.38	
2026-10-22	31.99	32.48	64.46	

Date	Strategy_Return	Strategy_Drawdown	Strategy_Drawdown_Daily_Max	\
1993-01-29	NaN	0.00	0.00	
1993-02-01	0.01	0.00	0.00	
1993-02-02	0.00	0.00	0.00	
1993-02-03	0.02	0.00	0.00	
1993-02-04	0.01	0.00	0.00	
...	...	...	...	
2026-10-16	-0.02	-0.04	-0.72	
2026-10-19	0.00	-0.04	-0.72	
2026-10-20	0.00	-0.04	-0.72	
2026-10-21	0.01	-0.03	-0.72	
2026-10-22	-0.01	-0.04	-0.72	

Date	Rebalance	Transaction_Fee
------	-----------	-----------------

1993-01-29	False	0.00
1993-02-01	False	0.00
1993-02-02	False	0.00
1993-02-03	False	0.00
1993-02-04	False	0.00
...	...	...
2026-10-16	False	0.00
2026-10-19	False	0.00
2026-10-20	False	0.00
2026-10-21	False	0.00
2026-10-22	False	0.00

[8491 rows x 16 columns]

Table E33: SPY/VBMFX Rebalance Strategy Data

Date	SPY_Adj_Close	VBMFX_Adj_Close	SPY_Return	VBMFX_Return	\
1993-01-29	1.00	1.00	NaN	NaN	
1993-02-01	1.01	1.00	0.01	0.00	
1993-02-02	1.01	1.00	0.00	-0.00	
1993-02-03	1.02	1.00	0.01	0.00	
1993-02-04	1.02	1.00	0.00	0.00	
...	...	...	...	...	
2026-10-16	31.73	4.00	-0.01	0.00	
2026-10-19	31.87	3.99	0.00	-0.00	
2026-10-20	31.97	4.00	0.00	0.00	
2026-10-21	32.14	4.00	0.01	0.00	
2026-10-22	31.76	4.03	-0.01	0.01	

Date	SPY_Drawdown	VBMFX_Drawdown	SPY_Drawdown_Daily_Max	\
1993-01-29	0.00	0.00		0.00
1993-02-01	0.00	0.00		0.00
1993-02-02	0.00	-0.00		0.00
1993-02-03	0.00	-0.00		0.00
1993-02-04	0.00	0.00		0.00
...	...	...	...	
2026-10-16	-0.02	-0.05		-0.55
2026-10-19	-0.02	-0.05		-0.55
2026-10-20	-0.01	-0.05		-0.55
2026-10-21	-0.01	-0.05		-0.55
2026-10-22	-0.02	-0.04		-0.55

Date	VBMFX_Drawdown_Daily_Max	SPY_Value	VBMFX_Value	\
1993-01-29		0.00	0.50	0.50
1993-02-01		0.00	0.50	0.50
1993-02-02		-0.00	0.50	0.50

1993-02-03	-0.00	0.51	0.50
1993-02-04	-0.00	0.51	0.50
...	...	...	...
2026-10-16	-0.19	6.13	6.06
2026-10-19	-0.19	6.16	6.04
2026-10-20	-0.19	6.18	6.04
2026-10-21	-0.19	6.21	6.04
2026-10-22	-0.19	6.14	6.10

Date	Strategy_Adj_Close	Strategy_Return	Strategy_Drawdown \
1993-01-29	1.00	NaN	0.00
1993-02-01	1.00	0.00	0.00
1993-02-02	1.00	0.00	0.00
1993-02-03	1.01	0.01	0.00
1993-02-04	1.01	0.00	0.00
...	...	...	...
2026-10-16	12.19	-0.01	-0.02
2026-10-19	12.19	0.00	-0.02
2026-10-20	12.22	0.00	-0.01
2026-10-21	12.25	0.00	-0.01
2026-10-22	12.23	-0.00	-0.01

Date	Strategy_Drawdown_Daily_Max	Rebalance	Transaction_Fee
1993-01-29	0.00	False	0.00
1993-02-01	0.00	False	0.00
1993-02-02	0.00	False	0.00
1993-02-03	0.00	False	0.00
1993-02-04	0.00	False	0.00
...	...	...	...
2026-10-16	-0.30	False	0.00
2026-10-19	-0.30	False	0.00
2026-10-20	-0.30	False	0.00
2026-10-21	-0.30	False	0.00
2026-10-22	-0.30	False	0.00

[8491 rows x 16 columns]

```
[73]: upro_extrap_vbmfx_full_sum_stats = summary_stats(
    fund_list=["UPRO (Extrap)/VBMFX"],
    df=upro_extrap_vbmfx_full[["Strategy_Return"]].dropna(),
    period="Daily",
    use_calendar_days=False,
    excel_export=False,
    pickle_export=False,
    output_confirmation=False,
```

```

)

# Rename the index to indicate the strategy
upro_extrap_vbmfx_full_sum_stats.index = ["UPRO (Extrap)/VBMFX Full Strategy"]

all_strat_sum_stats = pd.concat([all_strat_sum_stats,
    ↪upro_extrap_vbmfx_full_sum_stats])

upro_extrap_sum_stats = summary_stats(
    fund_list=["UPRO (Extrap)/VBMFX"],
    df=upro_extrap_vbmfx_full[["UPRO_Extrap_Return"]].dropna(),
    period="Daily",
    use_calendar_days=False,
    excel_export=False,
    pickle_export=False,
    output_confirmation=False,
)

# Rename the index to indicate the strategy
upro_extrap_sum_stats.index = ["UPRO (Extrap)"]

all_strat_sum_stats = pd.concat([all_strat_sum_stats, upro_extrap_sum_stats])

spy_vbmfx_full_sum_stats = summary_stats(
    fund_list=["SPY/VBMFX"],
    df=spy_vbmfx_full[["Strategy_Return"]].dropna(),
    period="Daily",
    use_calendar_days=False,
    excel_export=False,
    pickle_export=False,
    output_confirmation=False,
)

# Rename the index to indicate the strategy
spy_vbmfx_full_sum_stats.index = ["SPY/VBMFX Full Strategy"]

all_strat_sum_stats = pd.concat([all_strat_sum_stats, spy_vbmfx_full_sum_stats])

spy_sum_stats = summary_stats(
    fund_list=["SPY"],
    df=spy_vbmfx_full[["SPY_Return"]].dropna(),
    period="Daily",
    use_calendar_days=False,
    excel_export=False,
    pickle_export=False,
    output_confirmation=False,
)

```

```

# Rename the index to indicate the strategy
spy_sum_stats.index = ["SPY"]

all_strat_sum_stats = pd.concat([all_strat_sum_stats, spy_sum_stats])

print("Table E34: Strategy Summary Statistics")
display(all_strat_sum_stats)

```

Table E34: Strategy Summary Statistics

	Number of Observations	Data Start Date \
TQQQ/TMF Strategy	2679	2010-02-12
TQQQ	2679	2010-02-12
TMF	2679	2010-02-12
TQQQ/TMF BS	2679	2010-02-12
TQQQ/TMF Full Strategy	4199	2010-02-12
TQQQ/TMF Full BS	4199	2010-02-12
UPRO/AGG Full Strategy	4199	2010-02-12
UPRO (Extrap)/VBMFX Full Strategy	8490	1993-02-01
UPRO (Extrap)	8490	1993-02-01
SPY/VBMFX Full Strategy	8490	1993-02-01
SPY	8490	1993-02-01

	Data End Date \
TQQQ/TMF Strategy	2020-10-02
TQQQ	2020-10-02
TMF	2020-10-02
TQQQ/TMF BS	2020-10-02
TQQQ/TMF Full Strategy	2026-10-22
TQQQ/TMF Full BS	2026-10-22
UPRO/AGG Full Strategy	2026-10-22
UPRO (Extrap)/VBMFX Full Strategy	2026-10-22
UPRO (Extrap)	2026-10-22
SPY/VBMFX Full Strategy	2026-10-22
SPY	2026-10-22

	Annual Mean Return (Arithmetic) \
TQQQ/TMF Strategy	0.39
TQQQ	0.57
TMF	0.26
TQQQ/TMF BS	0.38
TQQQ/TMF Full Strategy	0.27
TQQQ/TMF Full BS	0.26
UPRO/AGG Full Strategy	0.20
UPRO (Extrap)/VBMFX Full Strategy	0.16
UPRO (Extrap)	0.31
SPY/VBMFX Full Strategy	0.08

SPY

0.12

	Annualized Volatility \
TQQQ/TMF Strategy	0.28
TQQQ	0.58
TMF	0.44
TQQQ/TMF BS	0.26
TQQQ/TMF Full Strategy	0.34
TQQQ/TMF Full BS	0.32
UPRO/AGG Full Strategy	0.24
UPRO (Extrap)/VBMFX Full Strategy	0.26
UPRO (Extrap)	0.55
SPY/VBMFX Full Strategy	0.09
SPY	0.19

	Annualized Sharpe Ratio	CAGR (Geometric) \
TQQQ/TMF Strategy	1.40	0.42
TQQQ	0.99	0.50
TMF	0.59	0.18
TQQQ/TMF BS	1.48	0.41
TQQQ/TMF Full Strategy	0.80	0.24
TQQQ/TMF Full BS	0.81	0.23
UPRO/AGG Full Strategy	0.83	0.19
UPRO (Extrap)/VBMFX Full Strategy	0.61	0.13
UPRO (Extrap)	0.56	0.17
SPY/VBMFX Full Strategy	0.86	0.08
SPY	0.65	0.11

	Daily Max Return	Daily Max Return (Date) \
TQQQ/TMF Strategy	0.11	2020-03-20
TQQQ	0.27	2020-03-13
TMF	0.20	2020-03-20
TQQQ/TMF BS	0.07	2020-03-06
TQQQ/TMF Full Strategy	0.17	2022-11-10
TQQQ/TMF Full BS	0.17	2022-11-10
UPRO/AGG Full Strategy	0.12	2025-04-09
UPRO (Extrap)/VBMFX Full Strategy	0.11	2009-03-23
UPRO (Extrap)	0.43	2008-10-13
SPY/VBMFX Full Strategy	0.06	2008-10-13
SPY	0.15	2008-10-13

	Daily Min Return	Daily Min Return (Date) \
TQQQ/TMF Strategy	-0.16	2020-03-18
TQQQ	-0.34	2020-03-16
TMF	-0.18	2020-03-18
TQQQ/TMF BS	-0.09	2020-09-03
TQQQ/TMF Full Strategy	-0.16	2020-03-18
TQQQ/TMF Full BS	-0.12	2022-05-05

UPRO/AGG Full Strategy	-0.13	2020-03-12
UPRO (Extrap)/VBMFX Full Strategy	-0.14	2020-03-16
UPRO (Extrap)	-0.35	2020-03-16
SPY/VBMFX Full Strategy	-0.05	2020-03-12
SPY	-0.11	2020-03-16

	Max Drawdown	Peak	Trough \
TQQQ/TMF Strategy	-0.42	2020-03-06	2020-03-18
TQQQ	-0.70	2020-02-19	2020-03-20
TMF	-0.53	2012-07-25	2013-08-21
TQQQ/TMF BS	-0.30	2018-08-29	2018-12-24
TQQQ/TMF Full Strategy	-0.78	2021-11-19	2022-11-09
TQQQ/TMF Full BS	-0.78	2021-11-19	2022-11-09
UPRO/AGG Full Strategy	-0.43	2021-12-27	2022-10-14
UPRO (Extrap)/VBMFX Full Strategy	-0.72	2007-10-09	2009-03-09
UPRO (Extrap)	-0.98	2000-03-24	2009-03-09
SPY/VBMFX Full Strategy	-0.30	2007-10-09	2009-03-09
SPY	-0.55	2007-10-09	2009-03-09

	Recovery Date	Calendar Days to Recovery \
TQQQ/TMF Strategy	2020-05-07	50.00
TQQQ	2020-07-10	112.00
TMF	2015-01-05	502.00
TQQQ/TMF BS	2019-04-23	120.00
TQQQ/TMF Full Strategy	NaT	NaN
TQQQ/TMF Full BS	NaT	NaN
UPRO/AGG Full Strategy	2024-06-12	607.00
UPRO (Extrap)/VBMFX Full Strategy	2013-01-18	1411.00
UPRO (Extrap)	2017-11-28	3186.00
SPY/VBMFX Full Strategy	2010-04-15	402.00
SPY	2012-08-16	1256.00

	MAR Ratio
TQQQ/TMF Strategy	1.00
TQQQ	0.71
TMF	0.33
TQQQ/TMF BS	1.38
TQQQ/TMF Full Strategy	0.30
TQQQ/TMF Full BS	0.30
UPRO/AGG Full Strategy	0.43
UPRO (Extrap)/VBMFX Full Strategy	0.18
UPRO (Extrap)	0.17
SPY/VBMFX Full Strategy	0.26
SPY	0.20

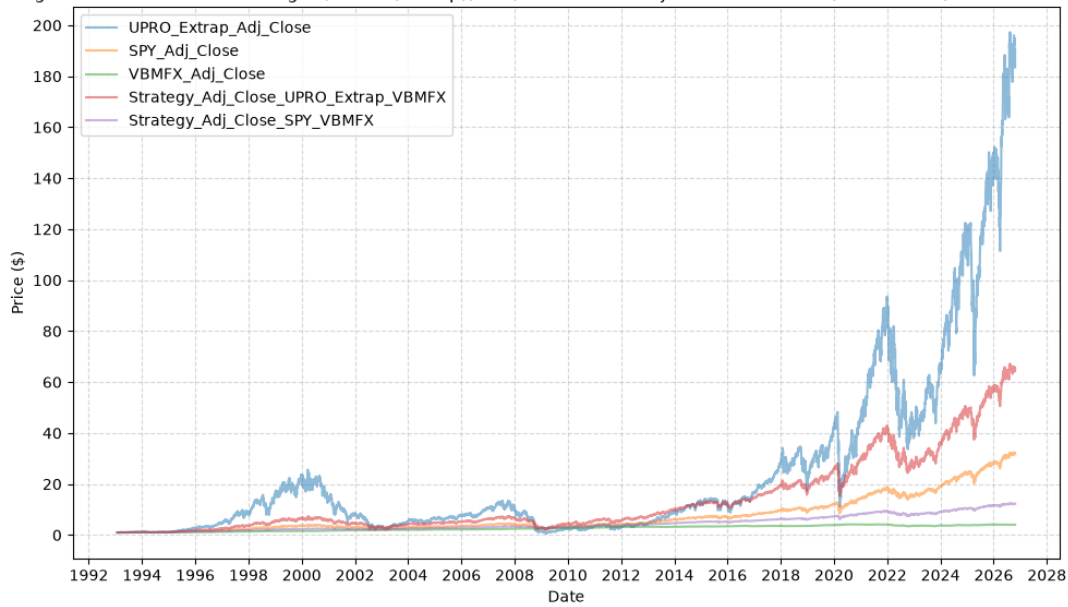
To make the comparison as easy as possible, we will plot both the UPRO (Extrap)/VBMFX strategy and the actual SPY/VBMFX strategy together, along with each of the individual components.

```

[74]: plot_time_series(
    df=upro_extrap_vbmfx_full.merge(
        spy_vbmfx_full,
        on="Date",
        how="outer",
        suffixes=("_UPRO_Extrap_VBMFX", "_SPY_VBMFX")),
    ).rename(columns={"VBMFX_Adj_Close_UPRO_Extrap_VBMFX": "VBMFX_Adj_Close"}),
    plot_start_date=None,
    plot_end_date=None,
    plot_columns=[
        "UPRO_Extrap_Adj_Close",
        "SPY_Adj_Close",
        "VBMFX_Adj_Close",
        "Strategy_Adj_Close_UPRO_Extrap_VBMFX",
        "Strategy_Adj_Close_SPY_VBMFX",
    ],
    title=f"Figure E10: Rebalance Strategies, UPRO (Extrap), SPY, and VBMFX_
↪Adjusted Close Price (Normalized) as of {upro_extrap_vbmfx_full.index[-1].
↪date()}",
    x_label="Date",
    x_format="Year",
    x_tick_spacing=2,
    x_tick_start=None,
    x_tick_rotation=0,
    y_label="Price ($)",
    y_format="Decimal",
    y_format_decimal_places=0,
    y_tick_spacing="Auto",
    y_tick_rotation=0,
    caption="Data Source: Yahoo Finance, Analysis by the Author",
)

```

Figure E10: Rebalance Strategies, UPRO (Extrap), SPY, and VBMFX Adjusted Close Price (Normalized) as of 2026-10-22

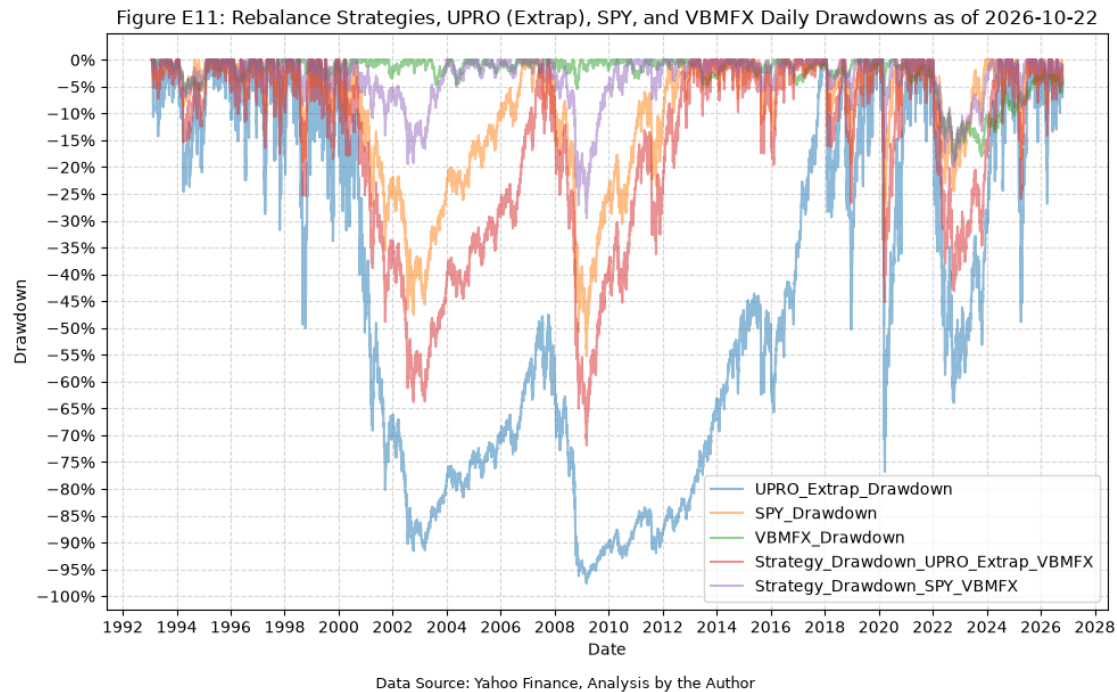


```
[75]: plot_time_series(
    df=upro_extrap_vbmfx_full.merge(
        spy_vbmfx_full,
        on="Date",
        how="outer",
        suffixes=("_UPRO_Extrap_VBMFX", "_SPY_VBMFX"),
    ).rename(columns={"VBMFX_Drawdown_UPRO_Extrap_VBMFX": "VBMFX_Drawdown"}),
    plot_start_date=None,
    plot_end_date=None,
    plot_columns=[
        "UPRO_Extrap_Drawdown",
        "SPY_Drawdown",
        "VBMFX_Drawdown",
        "Strategy_Drawdown_UPRO_Extrap_VBMFX",
        "Strategy_Drawdown_SPY_VBMFX",
    ],
    title=f"Figure E11: Rebalance Strategies, UPRO (Extrap), SPY, and VBMFX_␣
    ↪Daily Drawdowns as of {upro_extrap_vbmfx_full.index[-1].date()}",
    x_label="Date",
    x_format="Year",
    x_tick_spacing=2,
    x_tick_start=None,
    x_tick_rotation=0,
    y_label="Drawdown",
    y_format="Percentage",
```

```

y_format_decimal_places=0,
y_tick_spacing="Auto",
y_tick_rotation=0,
caption="Data Source: Yahoo Finance, Analysis by the Author",
)

```



```

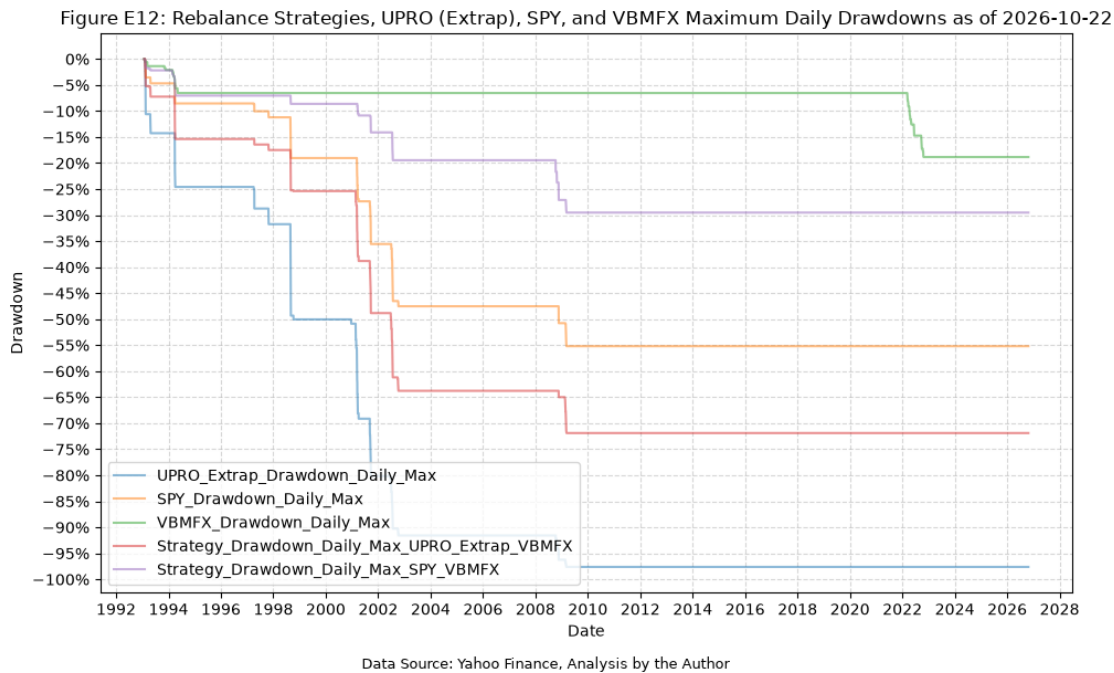
[76]: plot_time_series(
    df=upro_extrap_vbmfx_full.merge(
        spy_vbmfx_full,
        on="Date",
        how="outer",
        suffixes=("_UPRO_Extrap_VBMFX", "_SPY_VBMFX"),
    ).rename(
        columns={
            "VBMFX_Drawdown_Daily_Max_UPRO_Extrap_VBMFX": ↵
↵ "VBMFX_Drawdown_Daily_Max"
        }
    ),
    plot_start_date=None,
    plot_end_date=None,
    plot_columns=[
        "UPRO_Extrap_Drawdown_Daily_Max",
        "SPY_Drawdown_Daily_Max",
        "VBMFX_Drawdown_Daily_Max",
    ]
)

```

```

    "Strategy_Drawdown_Daily_Max_UPRO_Extrap_VBMFX",
    "Strategy_Drawdown_Daily_Max_SPY_VBMFX",
],
title=f"Figure E12: Rebalance Strategies, UPRO (Extrap), SPY, and VBMFX_
↪Maximum Daily Drawdowns as of {upro_extrap_vbmfx_full.index[-1].date()}",
x_label="Date",
x_format="Year",
x_tick_spacing=2,
x_tick_start=None,
x_tick_rotation=0,
y_label="Drawdown",
y_format="Percentage",
y_format_decimal_places=0,
y_tick_spacing="Auto",
y_tick_rotation=0,
caption="Data Source: Yahoo Finance, Analysis by the Author",
)

```



0.15 Conclusion

First, the way the paper handles the different dates for the CAGR, DDD, and MDD is quite unusual and requires careful consideration when attempting to reproduce the results accurately. Typically, one would expect a specific data range to be consistently applied across all metrics to ensure comparability and validity of the results. The unusual way that the paper handled the dates actually required quite a bit of extra effort to align the data correctly and update the summary

stats tables with additional fields to accurately portray what was being calculated in the paper.

Second, the pseudo-code for the “black swan” filter in figure 11 does not match the results reported in the paper. The pseudo-code sets the switch-back price v^* to the price of TQQQ on the day of the drop, $v(n)$, but implemented that way, the filter switches back to the standard strategy the very next day and whipsaws in and out of IEF three times during March 2020. Using the price of TQQQ on the day before the drop, $v(n - 1)$, instead reproduces both the 5/18/2020 switch-back date and the 30% maximum DDD reported in the paper. When the pseudo-code and the reported results disagree, the results (and the paper’s written description of the method) are the better guide for a replication.

0.16 Future Investigation

Nothing here yet.

0.17 References

1. Glenn, Lewis A., Long-Term Investing in Triple Leveraged Exchange Traded Funds (October 30, 2020). Available at SSRN: <https://ssrn.com/abstract=3722318> or <http://dx.doi.org/10.2139/ssrn.3722318>