

asset-class-performance-fed-policy-cycles

December 4, 2025

1 Performance Of Various Asset Classes During Fed Policy Cycles

1.1 Python Imports

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

from datetime import datetime, timedelta
from pathlib import Path

# Data Handling
import numpy as np
import pandas as pd

# Data Visualization
import matplotlib.dates as mdates
import matplotlib.pyplot as plt
import matplotlib.ticker as mtick
import seaborn as sns
from matplotlib.ticker import FormatStrFormatter, FuncFormatter, MultipleLocator

# Data Sources
import yfinance as yf
import pandas_datareader.data as web

# Statistical Analysis
import statsmodels.api as sm

# Machine Learning
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# Suppress warnings
```

```
warnings.filterwarnings("ignore")
```

1.2 Add Directories To Path

```
[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")

# Print system path
for i, path in enumerate(sys.path):
    print(f"{i}: {path}")
```

```
0: /usr/lib/python313.zip
1: /usr/lib/python3.13
2: /usr/lib/python3.13/lib-dynload
3:
4: /home/jared/python-virtual-envs/general_313/lib/python3.13/site-packages
5: /home/jared/python-virtual-envs/general_313/lib/python3.13/site-packages/setuptools/_vendor
6: /home/jared/Cloud_Storage/Dropbox/Websites/jaredszajkowski.github.io/src
```

1.3 Track Index Dependencies

```
[3]: # Create file to track markdown dependencies
dep_file = Path("index_dep.txt")
dep_file.write_text("")
```

[3]: 0

1.4 Python Functions

```
[4]: from calc_fed_cycle_asset_performance import calc_fed_cycle_asset_performance
from df_info import df_info
from df_info_markdown import df_info_markdown
from export_track_md_deps import export_track_md_deps
from load_data import load_data
from pandas_set_decimal_places import pandas_set_decimal_places
from plot_bar_returns_ffr_change import plot_bar_returns_ffr_change
from plot_timeseries import plot_timeseries
from plot_scatter_regression_ffr_vs_returns import plot_scatter_regression_ffr_vs_returns
from sm_ols_summary_markdown import sm_ols_summary_markdown
from summary_stats import summary_stats
from yf_pull_data import yf_pull_data
```

1.5 Data Overview

```
[5]: # Set timeframe
start_date = "2004-11-30" # GLD inception (month end)
end_date = "2025-10-31"
```

1.5.1 Acquire & Plot Fed Funds Data

```
[6]: # Set decimal places
pandas_set_decimal_places(4)

# Pull Effective Fed Funds Rate from FRED
fedfunds = web.DataReader("FEDFUNDS", "fred", start="1900-01-01", end=datetime.
    today())
fedfunds["FEDFUNDS"] = fedfunds["FEDFUNDS"] / 100 # Convert to decimal

# Resample to monthly frequency and compute change in rate
fedfunds_monthly = fedfunds.resample("M").last()
fedfunds_monthly = fedfunds_monthly[(fedfunds_monthly.index >= pd.
    to_datetime(start_date)) & (fedfunds_monthly.index <= pd.
    to_datetime(end_date))]
fedfunds_monthly["FedFunds_Change"] = fedfunds_monthly["FEDFUNDS"].diff()
```

```
[7]: df_info(fedfunds_monthly)
```

```
The columns, shape, and data types are:
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 252 entries, 2004-11-30 to 2025-10-31
Freq: ME
Data columns (total 2 columns):
#   Column          Non-Null Count  Dtype
---  -
0   FEDFUNDS         252 non-null    float64
1   FedFunds_Change  251 non-null    float64
dtypes: float64(2)
memory usage: 5.9 KB
None
The first 5 rows are:
```

	FEDFUNDS	FedFunds_Change
DATE		
2004-11-30	0.0193	NaN
2004-12-31	0.0216	0.0023
2005-01-31	0.0228	0.0012
2005-02-28	0.0250	0.0022
2005-03-31	0.0263	0.0013

```
The last 5 rows are:
```

	FEDFUNDS	FedFunds_Change
DATE		
2025-06-30	0.0433	0.0000
2025-07-31	0.0433	0.0000
2025-08-31	0.0433	0.0000
2025-09-30	0.0422	-0.0011
2025-10-31	0.0409	-0.0013

```
[8]: # Copy this <!-- INSERT_01_Fed_Funds_Monthly_Rate_Change_HERE --> to index_temp.
      ↪md
      export_track_md_deps(dep_file=dep_file,
      ↪md_filename="01_Fed_Funds_Monthly_Rate_Change.md",
      ↪content=df_info_markdown(df=fedfunds_monthly, decimal_places=4))
```

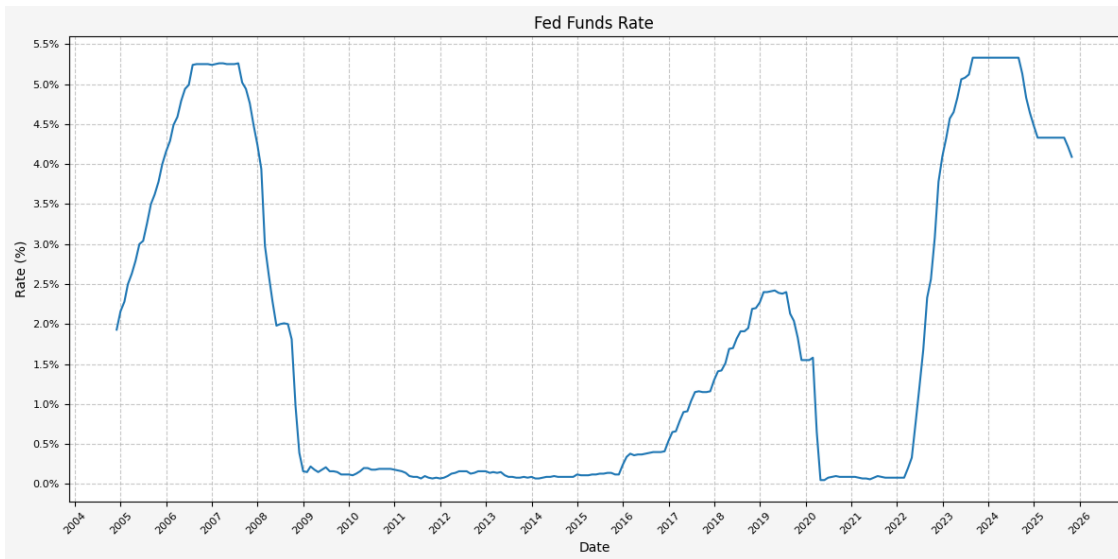
```
Exported and tracked: 01_Fed_Funds_Monthly_Rate_Change.md
```

```
[9]: plot_timeseries(
      price_df=fedfunds_monthly,
      plot_start_date=start_date,
      plot_end_date=end_date,
      plot_columns=["FEDFUNDS"],
      title="Fed Funds Rate",
      x_label="Date",
```

```

x_format="Year",
y_label="Rate (%)",
y_format="Percentage",
y_format_decimal_places=1,
y_tick_spacing=0.005,
grid=True,
legend=False,
export_plot=True,
plot_file_name="01_Fed_Funds_Rate",
)

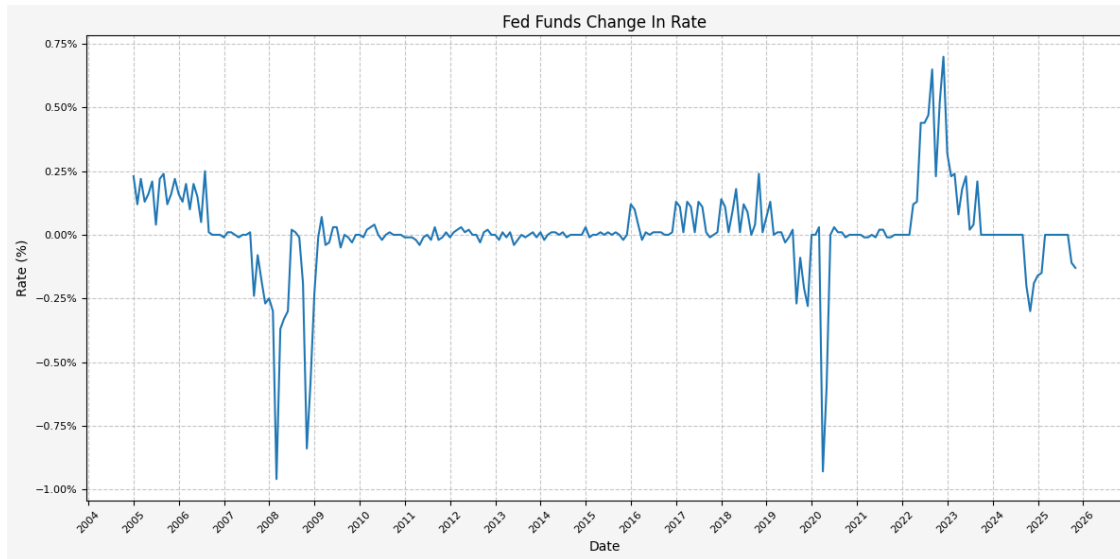
```



```

[10]: plot_timeseries(
    price_df=fedfunds_monthly,
    plot_start_date=start_date,
    plot_end_date=end_date,
    plot_columns=["FedFunds_Change"],
    title="Fed Funds Change In Rate",
    x_label="Date",
    x_format="Year",
    y_label="Rate (%)",
    y_format="Percentage",
    y_format_decimal_places=2,
    y_tick_spacing=0.0025,
    grid=True,
    legend=False,
    export_plot=True,
    plot_file_name="01_Fed_Funds_Change_In_Rate",
)

```



1.5.2 Define Fed Policy Cycles

```
[11]: # # Define manually specified Fed policy cycles
# fed_cycles = [
#     # ("2002-01-01", "2003-07-01"),
#     # ("2003-07-01", "2004-06-01"),
#     # ("2004-06-01", "2006-07-01"),
#     ("2004-11-01", "2006-07-01"),
#     ("2006-07-01", "2007-07-01"),
#     ("2007-07-01", "2008-12-01"),
#     ("2008-12-01", "2015-11-01"),
#     ("2015-11-01", "2019-01-01"),
#     ("2019-01-01", "2019-07-01"),
#     ("2019-07-01", "2020-04-01"),
#     ("2020-04-01", "2022-02-01"),
#     ("2022-02-01", "2023-08-01"),
#     ("2023-08-01", "2024-08-01"),
#     ("2024-08-01", datetime.today().strftime('%Y-%m-%d')),
# ]

# # Optional: assign a name to each cycle
# cycle_labels = [f"Cycle {i+1}" for i in range(len(fed_cycles))]
```

```
[12]: # Define manually specified Fed policy cycles
fed_cycles = [
    ("2004-11-01", "2006-07-01"),
    ("2006-07-01", "2007-07-01"),
    ("2007-07-01", "2008-12-01"),
```

```

("2008-12-01", "2015-11-01"),
("2015-11-01", "2019-01-01"),
("2019-01-01", "2019-07-01"),
("2019-07-01", "2020-04-01"),
("2020-04-01", "2022-02-01"),
("2022-02-01", "2023-08-01"),
("2023-08-01", "2024-08-01"),
("2024-08-01", datetime.today().strftime('%Y-%m-%d')),
]

# Optional: assign a name to each cycle
cycle_labels = [f"Cycle {i+1}" for i in range(len(fed_cycles))]

```

```

[13]: # Set decimal places
pandas_set_decimal_places(4)

# Calc changes by fed cycle defined above
fed_changes = []

for (start, end) in fed_cycles:
    start = pd.to_datetime(start)
    end = pd.to_datetime(end)

    try:
        rate_start = fedfunds.loc[start, "FEDFUNDS"]
    except KeyError:
        rate_start = fedfunds.loc[:start].iloc[-1]["FEDFUNDS"]

    try:
        rate_end = fedfunds.loc[end, "FEDFUNDS"]
    except KeyError:
        rate_end = fedfunds.loc[:end].iloc[-1]["FEDFUNDS"]

    change = rate_end - rate_start
    fed_changes.append(change)

fed_changes_df = pd.DataFrame({
    "Cycle": cycle_labels,
    "FedFunds_Change": fed_changes
})

fed_changes_df

```

```

[13]:      Cycle  FedFunds_Change
0    Cycle 1           0.0331
1    Cycle 2           0.0002
2    Cycle 3          -0.0510

```

3	Cycle 4	-0.0004
4	Cycle 5	0.0228
5	Cycle 6	0.0000
6	Cycle 7	-0.0235
7	Cycle 8	0.0003
8	Cycle 9	0.0525
9	Cycle 10	0.0000
10	Cycle 11	-0.0145

```
[14]: # Copy this <!-- INSERT_01_Fed_Funds_Cycle_Change_HERE --> to index_temp.md
export_track_md_deps(dep_file=dep_file, md_filename="01_Fed_Funds_Cycle_Change.
    ↪md", content=fed_changes_df.to_markdown(floatfmt=".4f"))
```

Exported and tracked: 01_Fed_Funds_Cycle_Change.md

1.6 Return Performance By Fed Policy Cycle

1.6.1 Stocks (SPY)

```
[15]: # Set decimal places
pandas_set_decimal_places(2)

yf_pull_data(
    base_directory=DATA_DIR,
    ticker="SPY",
    source="Yahoo_Finance",
    asset_class="Exchange_Traded_Funds",
    excel_export=True,
    pickle_export=True,
    output_confirmation=True,
)
```

[*****100%*****] 1 of 1 completed

The first and last date of data for SPY is:

	Close	High	Low	Open	Volume
Date					
1993-01-29	24.31	24.33	24.21	24.33	1003200
Date					
2025-12-03	683.89	684.91	679.69	680.57	57238500

Yahoo Finance data complete for SPY

```
[15]:
Date          Close    High    Low    Open    Volume
```



```

1993-01-29  24.31  24.33  24.21  24.33  1003200
1993-02-01  24.49  24.49  24.33  24.33   480500
1993-02-02  24.54  24.56  24.42  24.47   201300
1993-02-03  24.80  24.81  24.56  24.57   529400
1993-02-04  24.90  24.95  24.61  24.88   531500
...
2025-11-26  679.68  681.70  676.72  677.63  71879600
2025-11-28  683.39  683.67  680.50  680.86  49212000
2025-12-01  680.27  682.99  678.74  678.81  61201200
2025-12-02  681.53  683.82  679.33  681.92  62953800
2025-12-03  683.89  684.91  679.69  680.57  57238500

```

[8269 rows x 5 columns]

```

[16]: spy = load_data(
        base_directory=DATA_DIR,
        ticker="SPY",
        source="Yahoo_Finance",
        asset_class="Exchange_Traded_Funds",
        timeframe="Daily",
        file_format="pickle",
    )

    # Filter SPY to date range
    spy = spy[(spy.index >= pd.to_datetime(start_date)) & (spy.index <= pd.
        to_datetime(end_date))]

    # Resample to monthly frequency
    spy_monthly = spy.resample("M").last()
    spy_monthly["Monthly_Return"] = spy_monthly["Close"].pct_change()

```

```

[17]: df_info(spy_monthly)

```

The columns, shape, and data types are:

```
<class 'pandas.core.frame.DataFrame'>
```

DatetimeIndex: 252 entries, 2004-11-30 to 2025-10-31

Freq: ME

Data columns (total 6 columns):

#	Column	Non-Null Count	Dtype
0	Close	252 non-null	float64
1	High	252 non-null	float64
2	Low	252 non-null	float64
3	Open	252 non-null	float64
4	Volume	252 non-null	int64
5	Monthly_Return	251 non-null	float64

dtypes: float64(5), int64(1)

memory usage: 13.8 KB

None

The first 5 rows are:

	Close	High	Low	Open	Volume	Monthly_Return
Date						
2004-11-30	79.83	80.07	79.66	79.90	53685200	NaN
2004-12-31	82.23	82.77	82.19	82.53	28648800	0.03
2005-01-31	80.39	80.45	80.08	80.25	52532700	-0.02
2005-02-28	82.07	82.53	81.67	82.43	69381300	0.02
2005-03-31	80.57	80.91	80.51	80.73	64575400	-0.02

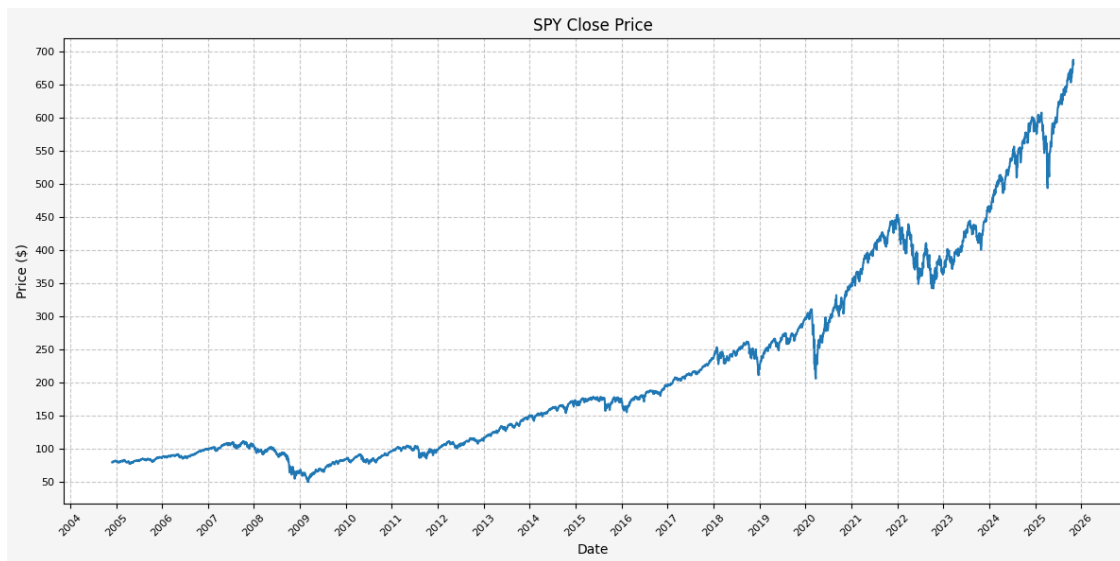
The last 5 rows are:

	Close	High	Low	Open	Volume	Monthly_Return
Date						
2025-06-30	616.14	617.51	613.34	615.67	92502500	0.05
2025-07-31	630.33	638.08	629.03	637.69	103385200	0.02
2025-08-31	643.27	646.05	641.36	645.68	74522200	0.02
2025-09-30	666.18	666.65	661.61	662.93	86288000	0.04
2025-10-31	682.06	685.08	679.24	685.04	87164100	0.02

```
[18]: # Copy this <!-- INSERT_02_SPY_Monthly_HERE --> to index_temp.md
export_track_md_deps(dep_file=dep_file, md_filename="02_SPY_Monthly.md",
↪content=df_info_markdown(df=spy_monthly, decimal_places=2))
```

Exported and tracked: 02_SPY_Monthly.md

```
[19]: plot_timeseries(
    price_df=spy,
    plot_start_date=start_date,
    plot_end_date=end_date,
    plot_columns=["Close"],
    title="SPY Close Price",
    x_label="Date",
    x_format="Year",
    y_label="Price ($)",
    y_format="Decimal",
    y_format_decimal_places=0,
    y_tick_spacing=50,
    grid=True,
    legend=False,
    export_plot=True,
    plot_file_name="02_SPY_Price",
)
```



```
[20]: spy_cycle_df = calc_fed_cycle_asset_performance(
    fed_cycles=fed_cycles,
    cycle_labels=cycle_labels,
    fed_changes=fed_changes,
    monthly_df=spy_monthly,
)

spy_cycle_df
```

```
[20]:
```

	Cycle	Start	End	Months	CumulativeReturn \
0	Cycle 1	2004-11-01	2006-07-01	20	0.11
1	Cycle 2	2006-07-01	2007-07-01	12	0.20
2	Cycle 3	2007-07-01	2008-12-01	17	-0.39
3	Cycle 4	2008-12-01	2015-11-01	83	1.67
4	Cycle 5	2015-11-01	2019-01-01	38	0.28
5	Cycle 6	2019-01-01	2019-07-01	6	0.18
6	Cycle 7	2019-07-01	2020-04-01	9	-0.11
7	Cycle 8	2020-04-01	2022-02-01	22	0.79
8	Cycle 9	2022-02-01	2023-08-01	18	0.04
9	Cycle 10	2023-08-01	2024-08-01	12	0.22
10	Cycle 11	2024-08-01	2025-12-04	15	0.26

	CumulativeReturnPct	AverageMonthlyReturn	AverageMonthlyReturnPct \
0	11.32	0.01	0.59
1	20.36	0.02	1.57
2	-38.55	-0.03	-2.67
3	167.34	0.01	1.28
4	28.30	0.01	0.70

5	18.33	0.03	2.95
6	-10.67	-0.01	-1.10
7	79.13	0.03	2.78
8	4.18	0.00	0.40
9	22.00	0.02	1.75
10	25.72	0.02	1.59

	AnnualizedReturn	AnnualizedReturnPct	Volatility	FedFundsChange	\
0	0.07	6.64	0.08	0.03	
1	0.20	20.36	0.07	0.00	
2	-0.29	-29.09	0.19	-0.05	
3	0.15	15.28	0.15	-0.00	
4	0.08	8.19	0.11	0.02	
5	0.40	40.01	0.18	0.00	
6	-0.14	-13.96	0.19	-0.02	
7	0.37	37.43	0.16	0.00	
8	0.03	2.77	0.21	0.05	
9	0.22	22.00	0.15	0.00	
10	0.20	20.09	0.11	-0.01	

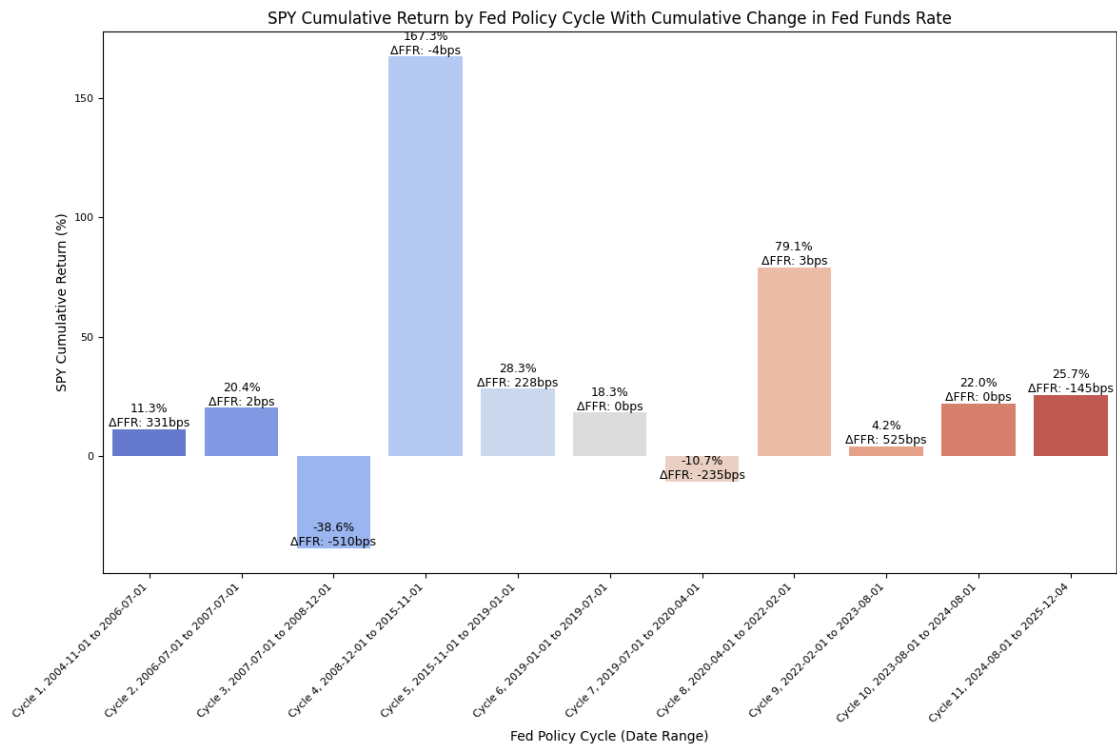
	FedFundsChange_bps	FFR_AnnualizedChange	FFR_AnnualizedChange_bps	\
0	331.00	0.02	198.60	
1	2.00	0.00	2.00	
2	-510.00	-0.04	-360.00	
3	-4.00	-0.00	-0.58	
4	228.00	0.01	72.00	
5	0.00	0.00	0.00	
6	-235.00	-0.03	-313.33	
7	3.00	0.00	1.64	
8	525.00	0.03	350.00	
9	0.00	0.00	0.00	
10	-145.00	-0.01	-116.00	

	Label
0	Cycle 1, 2004-11-01 to 2006-07-01
1	Cycle 2, 2006-07-01 to 2007-07-01
2	Cycle 3, 2007-07-01 to 2008-12-01
3	Cycle 4, 2008-12-01 to 2015-11-01
4	Cycle 5, 2015-11-01 to 2019-01-01
5	Cycle 6, 2019-01-01 to 2019-07-01
6	Cycle 7, 2019-07-01 to 2020-04-01
7	Cycle 8, 2020-04-01 to 2022-02-01
8	Cycle 9, 2022-02-01 to 2023-08-01
9	Cycle 10, 2023-08-01 to 2024-08-01
10	Cycle 11, 2024-08-01 to 2025-12-04

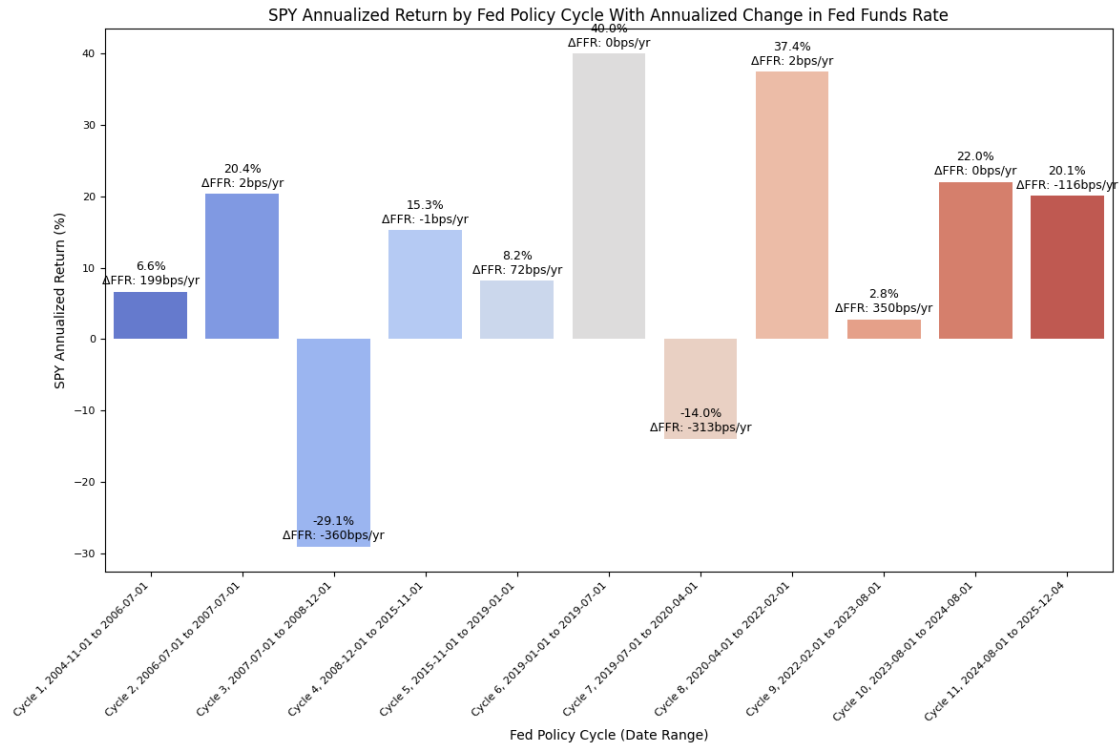
```
[21]: # Copy this <!-- INSERT_02_SPY_Cycle_DF_HERE --> to index_temp.md
export_track_md_deps(dep_file=dep_file, md_filename="02_SPY_Cycle_DF.md",
↪ content=spy_cycle_df.to_markdown(floatfmt=".2f"))
```

Exported and tracked: 02_SPY_Cycle_DF.md

```
[22]: plot_bar_returns_ffr_change(
    cycle_df=spy_cycle_df,
    asset_label="SPY",
    annualized_or_cumulative="Cumulative",
    index_num="02",
)
```



```
[23]: plot_bar_returns_ffr_change(
    cycle_df=spy_cycle_df,
    asset_label="SPY",
    annualized_or_cumulative="Annualized",
    index_num="02",
)
```



```
[24]: df = spy_cycle_df

#####
### Don't modify below this line ###
#####

# Run OLS regression with statsmodels
X = df["FFR_AnnualizedChange_bps"]
y = df["AnnualizedReturnPct"]
X = sm.add_constant(X)
model = sm.OLS(y, X).fit()
print(model.summary())
print(f"Intercept: {model.params[0]}, Slope: {model.params[1]}") # Intercept,
    ↪ and slope

# Calc X and Y values for regression line
X_vals = np.linspace(X.min(), X.max(), 100)
Y_vals = model.params[0] + model.params[1] * X_vals
```

OLS Regression Results

```
=====
Dep. Variable:    AnnualizedReturnPct    R-squared:                0.180
Model:            OLS                    Adj. R-squared:          0.089
```

```

Method:                Least Squares      F-statistic:                1.973
Date:                  Thu, 04 Dec 2025    Prob (F-statistic):         0.194
Time:                  14:45:19           Log-Likelihood:             -47.173
No. Observations:      11                AIC:                        98.35
Df Residuals:          9                 BIC:                        99.14
Df Model:              1
Covariance Type:       nonrobust

```

```

=====
=====
              coef      std err          t      P>|t|      [0.025
0.975]
-----
const              12.4404      5.894      2.111      0.064      -0.893
25.774
FFR_AnnualizedChange_bps  0.0430      0.031      1.405      0.194      -0.026
0.112
=====
Omnibus:              1.065      Durbin-Watson:              3.078
Prob(Omnibus):         0.587      Jarque-Bera (JB):           0.665
Skew:                  0.026      Prob(JB):                  0.717
Kurtosis:              1.796      Cond. No.                  193.
=====

```

Notes:

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

Intercept: 12.440379319933031, Slope: 0.0429736577476875

```

[25]: # Copy this <!-- INSERT_O2_SPY_Annualized_Regression_HERE --> to index_temp.md
export_track_md_deps(dep_file=dep_file,
    ↪md_filename="O2_SPY_Annualized_Regression.md",
    ↪content=sm_ols_summary_markdown(result=model,
    ↪file_path="O2_SPY_Annualized_Regression.md"))

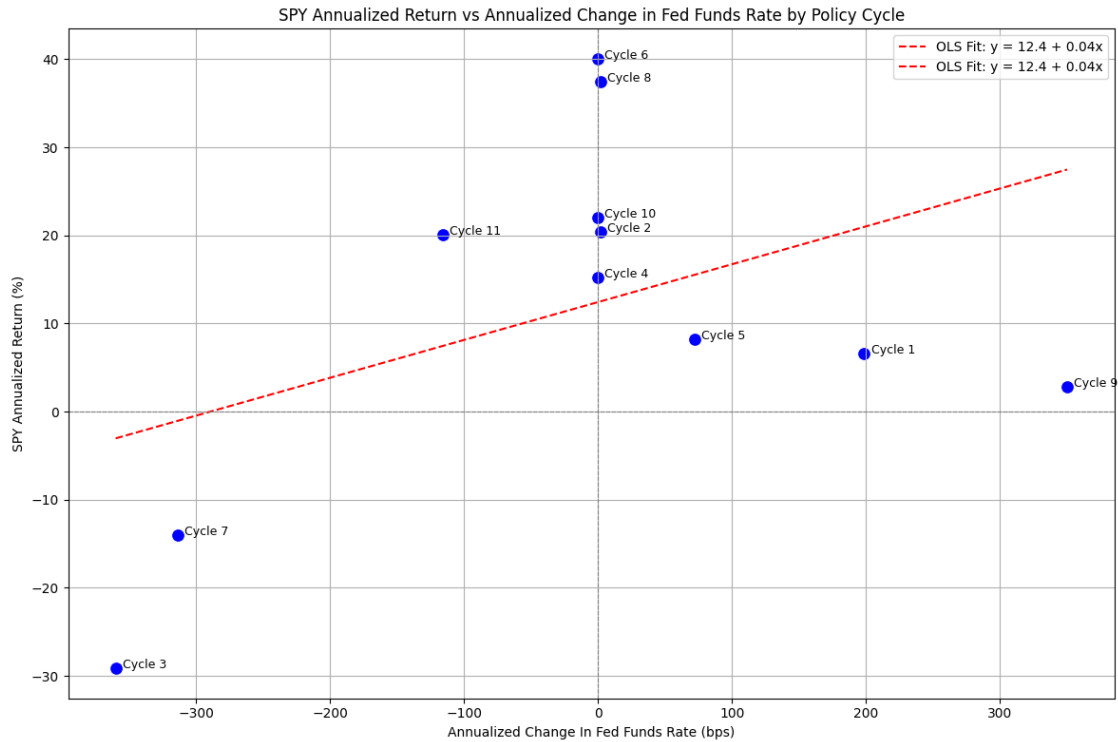
```

Exported and tracked: O2_SPY_Annualized_Regression.md

```

[26]: plot_scatter_regression_ffr_vs_returns(
    cycle_df=spy_cycle_df,
    asset_label="SPY",
    index_num="O2",
    x_vals=X_vals,
    y_vals=Y_vals,
    intercept=model.params[0],
    slope=model.params[1],
)

```



1.6.2 Bonds (TLT)

```
[27]: # Set decimal places
pandas_set_decimal_places(2)

yf_pull_data(
    base_directory=DATA_DIR,
    ticker="TLT",
    source="Yahoo_Finance",
    asset_class="Exchange_Traded_Funds",
    excel_export=True,
    pickle_export=True,
    output_confirmation=True,
)
```

[*****100%*****] 1 of 1 completed

The first and last date of data for TLT is:

	Close	High	Low	Open	Volume
Date					
2002-07-30	36.79	36.96	36.79	36.89	6100

	Close	High	Low	Open	Volume
Date					
2025-12-03	89.06	89.16	88.74	88.92	23278600

Yahoo Finance data complete for TLT

```
[27]:
```

	Close	High	Low	Open	Volume
Date					
2002-07-30	36.79	36.96	36.79	36.89	6100
2002-07-31	37.25	37.37	36.96	36.98	29400
2002-08-01	37.46	37.47	37.25	37.25	25000
2002-08-02	37.84	37.96	37.40	37.53	52800
2002-08-05	38.01	38.11	37.84	37.93	61100
...
2025-11-26	90.32	90.32	89.51	89.83	34099000
2025-11-28	89.89	90.21	89.63	90.14	40704300
2025-12-01	88.77	88.99	88.71	88.85	47086600
2025-12-02	88.81	88.96	88.46	88.58	27605700
2025-12-03	89.06	89.16	88.74	88.92	23278600

[5876 rows x 5 columns]

```
[28]: tlt = load_data(
        base_directory=DATA_DIR,
        ticker="TLT",
        source="Yahoo_Finance",
        asset_class="Exchange_Traded_Funds",
        timeframe="Daily",
        file_format="pickle",
    )

    # Filter TLT to date range
    tlt = tlt[(tlt.index >= pd.to_datetime(start_date)) & (tlt.index <= pd.
        ↪to_datetime(end_date))]

    # Resample to monthly frequency
    tlt_monthly = tlt.resample("M").last()
    tlt_monthly["Monthly_Return"] = tlt_monthly["Close"].pct_change()
```

```
[29]: df_info(tlt_monthly)
```

The columns, shape, and data types are:

```
<class 'pandas.core.frame.DataFrame'>
```

```
DatetimeIndex: 252 entries, 2004-11-30 to 2025-10-31
```

```
Freq: ME
```

```
Data columns (total 6 columns):
```

#	Column	Non-Null Count	Dtype
---	-----	-----	-----

```

0   Close          252 non-null    float64
1   High           252 non-null    float64
2   Low            252 non-null    float64
3   Open           252 non-null    float64
4   Volume         252 non-null    int64
5   Monthly_Return 251 non-null    float64

```

dtypes: float64(5), int64(1)

memory usage: 13.8 KB

None

The first 5 rows are:

	Close	High	Low	Open	Volume	Monthly_Return
Date						
2004-11-30	43.97	44.08	43.81	43.97	1754500	NaN
2004-12-31	45.14	45.19	45.01	45.05	1056400	0.03
2005-01-31	46.75	46.77	46.53	46.55	1313900	0.04
2005-02-28	46.06	46.61	45.99	46.61	2797300	-0.01
2005-03-31	45.85	45.88	45.61	45.78	2410900	-0.00

The last 5 rows are:

	Close	High	Low	Open	Volume	Monthly_Return
Date						
2025-06-30	86.33	86.52	85.71	85.95	53695200	0.03
2025-07-31	85.35	85.83	85.27	85.55	49814100	-0.01
2025-08-31	85.36	85.61	85.20	85.52	41686400	0.00
2025-09-30	88.42	89.09	88.27	88.71	38584000	0.04
2025-10-31	89.64	90.01	89.56	89.91	38247300	0.01

```

[30]: # Copy this <!-- INSERT_03_TLT_Monthly_HERE --> to index_temp.md
export_track_md_deps(dep_file=dep_file, md_filename="03_TLT_Monthly.md",
    ↪content=df_info_markdown(df=tl_t_monthly, decimal_places=2))

```

Exported and tracked: 03_TLT_Monthly.md

```

[31]: plot_timeseries(
    price_df=tl_t,
    plot_start_date=start_date,
    plot_end_date=end_date,
    plot_columns=["Close"],
    title="TLT Close Price",
    x_label="Date",
    x_format="Year",
    y_label="Price ($)",
    y_format="Decimal",
    y_format_decimal_places=0,
    y_tick_spacing=10,
    grid=True,
    legend=False,

```

```

export_plot=True,
plot_file_name="03_TLT_Price",
)

```



```

[32]: tlt_cycle_df = calc_fed_cycle_asset_performance(
        fed_cycles=fed_cycles,
        cycle_labels=cycle_labels,
        fed_changes=fed_changes,
        monthly_df=tlt_monthly,
    )

tlt_cycle_df

```

```

[32]:
   Cycle  Start      End  Months  CumulativeReturn  \
0  Cycle 1  2004-11-01  2006-07-01      20           0.04
1  Cycle 2  2006-07-01  2007-07-01      12           0.06
2  Cycle 3  2007-07-01  2008-12-01      17           0.32
3  Cycle 4  2008-12-01  2015-11-01      83           0.46
4  Cycle 5  2015-11-01  2019-01-01      38           0.07
5  Cycle 6  2019-01-01  2019-07-01       6           0.10
6  Cycle 7  2019-07-01  2020-04-01       9           0.26
7  Cycle 8  2020-04-01  2022-02-01      22          -0.11
8  Cycle 9  2022-02-01  2023-08-01      18          -0.27
9  Cycle 10 2023-08-01  2024-08-01      12          -0.02
10 Cycle 11 2024-08-01  2025-12-04      15           0.00

   CumulativeReturnPct  AverageMonthlyReturn  AverageMonthlyReturnPct  \
0                    4.23                    0.00                    0.25

```

1	5.76	0.00	0.49
2	32.42	0.02	1.73
3	45.67	0.01	0.55
4	7.42	0.00	0.23
5	10.48	0.02	1.73
6	26.18	0.03	2.73
7	-11.33	-0.00	-0.50
8	-26.96	-0.02	-1.62
9	-1.52	0.00	0.02
10	0.42	0.00	0.08

	AnnualizedReturn	AnnualizedReturnPct	Volatility	FedFundsChange	\
0	0.03	2.51	0.09	0.03	
1	0.06	5.76	0.07	0.00	
2	0.22	21.92	0.14	-0.05	
3	0.06	5.59	0.15	-0.00	
4	0.02	2.29	0.10	0.02	
5	0.22	22.05	0.13	0.00	
6	0.36	36.34	0.18	-0.02	
7	-0.06	-6.35	0.11	0.00	
8	-0.19	-18.90	0.17	0.05	
9	-0.02	-1.52	0.20	0.00	
10	0.00	0.33	0.11	-0.01	

	FedFundsChange_bps	FFR_AnnualizedChange	FFR_AnnualizedChange_bps	\
0	331.00	0.02	198.60	
1	2.00	0.00	2.00	
2	-510.00	-0.04	-360.00	
3	-4.00	-0.00	-0.58	
4	228.00	0.01	72.00	
5	0.00	0.00	0.00	
6	-235.00	-0.03	-313.33	
7	3.00	0.00	1.64	
8	525.00	0.03	350.00	
9	0.00	0.00	0.00	
10	-145.00	-0.01	-116.00	

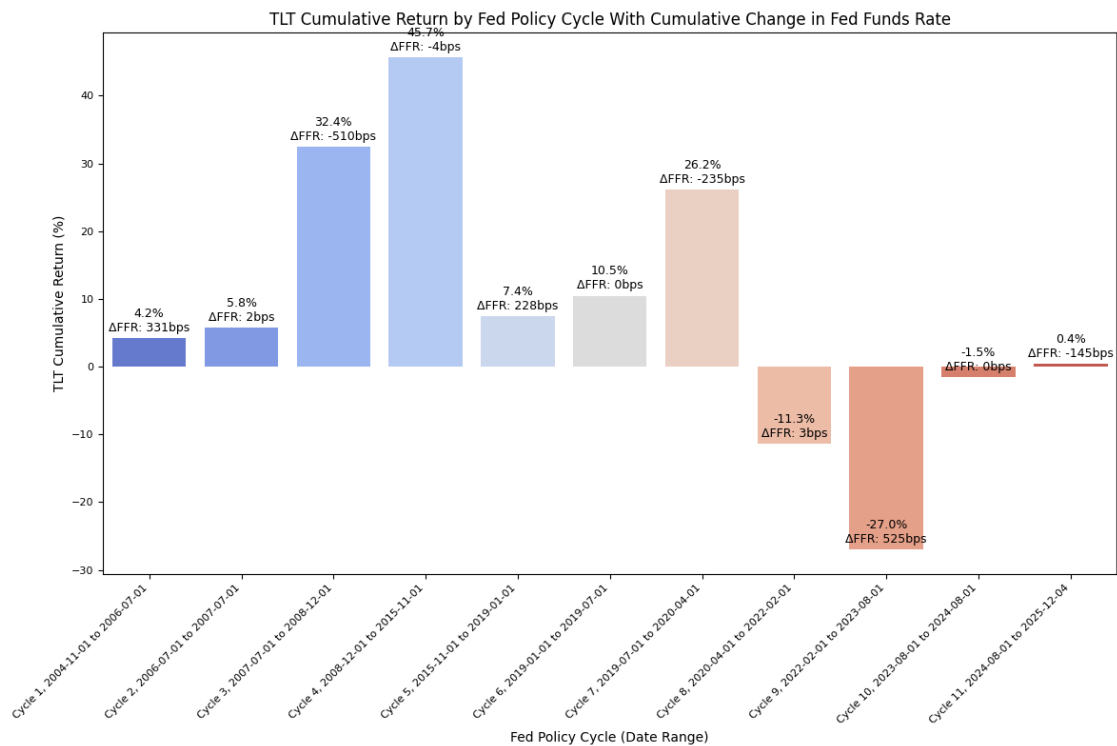
	Label
0	Cycle 1, 2004-11-01 to 2006-07-01
1	Cycle 2, 2006-07-01 to 2007-07-01
2	Cycle 3, 2007-07-01 to 2008-12-01
3	Cycle 4, 2008-12-01 to 2015-11-01
4	Cycle 5, 2015-11-01 to 2019-01-01
5	Cycle 6, 2019-01-01 to 2019-07-01
6	Cycle 7, 2019-07-01 to 2020-04-01
7	Cycle 8, 2020-04-01 to 2022-02-01
8	Cycle 9, 2022-02-01 to 2023-08-01

9 Cycle 10, 2023-08-01 to 2024-08-01
 10 Cycle 11, 2024-08-01 to 2025-12-04

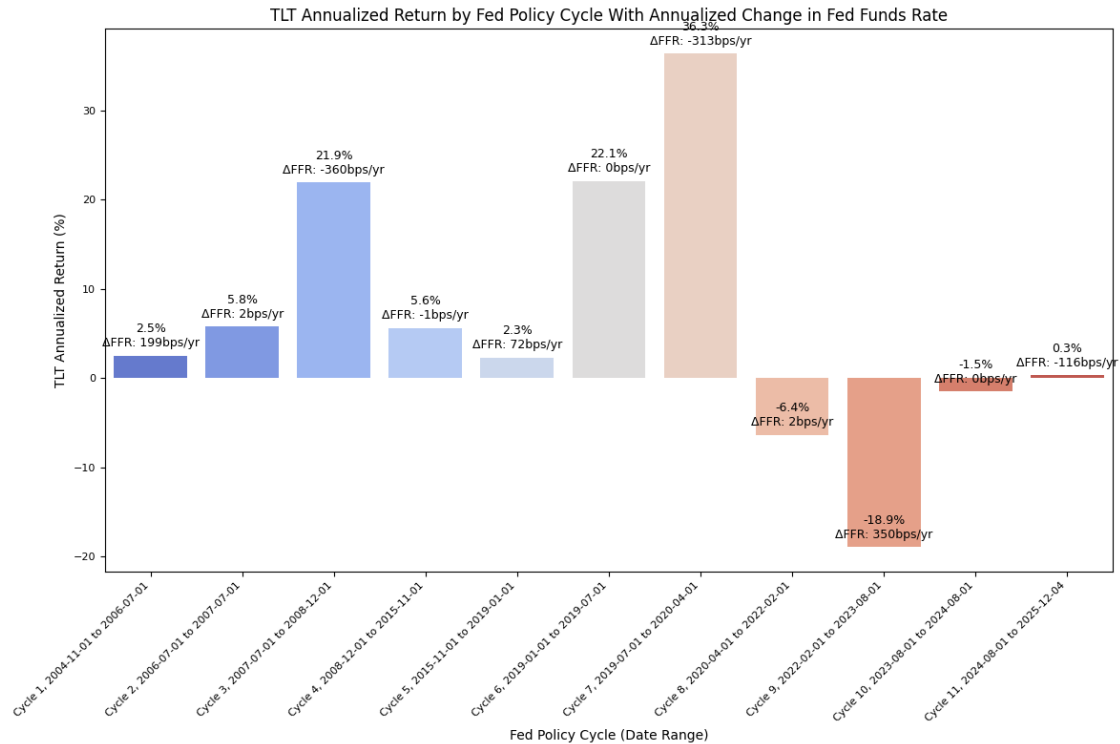
```
[33]: # Copy this <!-- INSERT_03_TLT_Cycle_DF_HERE --> to index_temp.md
export_track_md_deps(dep_file=dep_file, md_filename="03_TLT_Cycle_DF.md",
  ↪content=tlt_cycle_df.to_markdown(floatfmt=".2f"))
```

Exported and tracked: 03_TLT_Cycle_DF.md

```
[34]: plot_bar_returns_ffr_change(
  cycle_df=tlt_cycle_df,
  asset_label="TLT",
  annualized_or_cumulative="Cumulative",
  index_num="03",
)
```



```
[35]: plot_bar_returns_ffr_change(
  cycle_df=tlt_cycle_df,
  asset_label="TLT",
  annualized_or_cumulative="Annualized",
  index_num="03",
)
```



```
[36]: df = tlt_cycle_df

#####
### Don't modify below this line ###
#####

# Run OLS regression with statsmodels
X = df["FFR_AnnualizedChange_bps"]
y = df["AnnualizedReturnPct"]
X = sm.add_constant(X)
model = sm.OLS(y, X).fit()
print(model.summary())
print(f"Intercept: {model.params[0]}, Slope: {model.params[1]}") # Intercept,
    ↪ and slope

# Calc X and Y values for regression line
X_vals = np.linspace(X.min(), X.max(), 100)
Y_vals = model.params[0] + model.params[1] * X_vals
```

OLS Regression Results

```
=====
Dep. Variable:    AnnualizedReturnPct    R-squared:                0.623
Model:            OLS                    Adj. R-squared:           0.582
```

```

Method:                Least Squares      F-statistic:                14.90
Date:                  Thu, 04 Dec 2025    Prob (F-statistic):         0.00385
Time:                  14:45:23           Log-Likelihood:             -39.665
No. Observations:      11                AIC:                        83.33
Df Residuals:          9                 BIC:                        84.13
Df Model:              1
Covariance Type:       nonrobust

```

```

=====
=====
              coef      std err          t      P>|t|      [0.025
0.975]
-----
const              5.4676      2.978      1.836      0.100     -1.270
12.205
FFR_AnnualizedChange_bps -0.0597      0.015     -3.860      0.004     -0.095
-0.025
=====
Omnibus:            0.710    Durbin-Watson:              1.219
Prob(Omnibus):      0.701    Jarque-Bera (JB):            0.663
Skew:               0.412    Prob(JB):                   0.718
Kurtosis:           2.123    Cond. No.                    193.
=====

```

Notes:

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

Intercept: 5.4676015895163275, Slope: -0.05966490608532464

```

[37]: # Copy this <!-- INSERT_O3_TLT_Annualized_Regression_HERE --> to index_temp.md
export_track_md_deps(dep_file=dep_file,
    ↪md_filename="O3_TLT_Annualized_Regression.md",
    ↪content=sm_ols_summary_markdown(result=model,
    ↪file_path="O3_TLT_Annualized_Regression.md"))

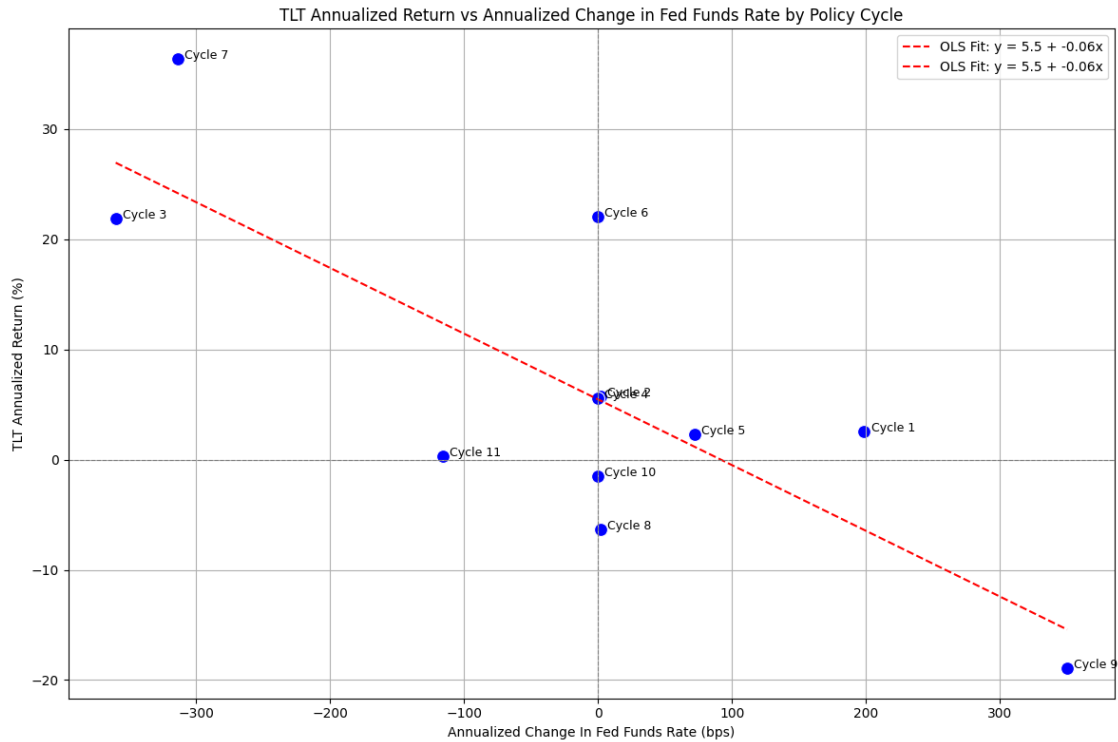
```

Exported and tracked: O3_TLT_Annualized_Regression.md

```

[38]: plot_scatter_regression_ffr_vs_returns(
    cycle_df=tlc_cycle_df,
    asset_label="TLT",
    index_num="O3",
    x_vals=X_vals,
    y_vals=Y_vals,
    intercept=model.params[0],
    slope=model.params[1],
)

```



1.6.3 Gold (GLD)

```
[39]: # Set decimal places
pandas_set_decimal_places(2)
```

```
yf_pull_data(
    base_directory=DATA_DIR,
    ticker="GLD",
    source="Yahoo_Finance",
    asset_class="Exchange_Traded_Funds",
    excel_export=True,
    pickle_export=True,
    output_confirmation=True,
)
```

```
[*****100%*****] 1 of 1 completed
```

The first and last date of data for GLD is:

Date	Close	High	Low	Open	Volume
2004-11-18	44.38	44.49	44.07	44.43	5992000

	Close	High	Low	Open	Volume
Date					
2025-12-03	386.88	390.13	385.83	388.46	7886800

Yahoo Finance data complete for GLD

```
[39]:
```

	Close	High	Low	Open	Volume
Date					
2004-11-18	44.38	44.49	44.07	44.43	5992000
2004-11-19	44.78	44.92	44.47	44.49	11655300
2004-11-22	44.95	44.97	44.74	44.75	11996000
2004-11-23	44.75	44.92	44.72	44.88	3169200
2004-11-24	45.05	45.05	44.79	44.93	6105100
...
2025-11-26	383.12	383.78	380.46	381.06	9184100
2025-11-28	387.88	388.00	384.99	385.98	7175400
2025-12-01	389.75	390.70	388.06	390.61	11177100
2025-12-02	387.24	388.98	382.91	388.87	8103900
2025-12-03	386.88	390.13	385.83	388.46	7886800

[5294 rows x 5 columns]

```
[40]: gld = load_data(
        base_directory=DATA_DIR,
        ticker="GLD",
        source="Yahoo_Finance",
        asset_class="Exchange_Traded_Funds",
        timeframe="Daily",
        file_format="pickle",
    )

# Filter GLD to date range
gld = gld[(gld.index >= pd.to_datetime(start_date)) & (gld.index <= pd.
    ↪to_datetime(end_date))]

# Resample to monthly frequency
gld_monthly = gld.resample("M").last()
gld_monthly["Monthly_Return"] = gld_monthly["Close"].pct_change()
```

```
[41]: df_info(gld_monthly)
```

The columns, shape, and data types are:

```
<class 'pandas.core.frame.DataFrame'>
```

```
DatetimeIndex: 252 entries, 2004-11-30 to 2025-10-31
```

```
Freq: ME
```

```
Data columns (total 6 columns):
```

#	Column	Non-Null Count	Dtype
---	-----	-----	-----

```

0   Close          252 non-null    float64
1   High           252 non-null    float64
2   Low            252 non-null    float64
3   Open           252 non-null    float64
4   Volume         252 non-null    int64
5   Monthly_Return 251 non-null    float64

```

dtypes: float64(5), int64(1)

memory usage: 13.8 KB

None

The first 5 rows are:

	Close	High	Low	Open	Volume	Monthly_Return
Date						
2004-11-30	45.12	45.41	44.82	45.37	3857200	NaN
2004-12-31	43.80	43.94	43.73	43.85	531600	-0.03
2005-01-31	42.22	42.30	41.96	42.21	1692400	-0.04
2005-02-28	43.53	43.74	43.52	43.68	755300	0.03
2005-03-31	42.82	42.87	42.70	42.87	1363200	-0.02

The last 5 rows are:

	Close	High	Low	Open	Volume	Monthly_Return
Date						
2025-06-30	304.83	304.92	301.95	302.39	8192100	0.00
2025-07-31	302.96	304.61	302.86	304.59	8981000	-0.01
2025-08-31	318.07	318.09	314.64	314.72	15642600	0.05
2025-09-30	355.47	355.57	350.87	351.13	13312400	0.12
2025-10-31	368.12	370.66	365.50	370.47	11077900	0.04

```

[42]: # Copy this <!-- INSERT_04_GLD_Monthly_HERE --> to index_temp.md
export_track_md_deps(dep_file=dep_file, md_filename="04_GLD_Monthly.md",
    ↪content=df_info_markdown(df=gld_monthly, decimal_places=2))

```

Exported and tracked: 04_GLD_Monthly.md

```

[43]: plot_timeseries(
    price_df=gld,
    plot_start_date=start_date,
    plot_end_date=end_date,
    plot_columns=["Close"],
    title="GLD Close Price",
    x_label="Date",
    x_format="Year",
    y_label="Price ($)",
    y_format="Decimal",
    y_format_decimal_places=0,
    y_tick_spacing=25,
    grid=True,
    legend=False,

```

```

export_plot=True,
plot_file_name="04_GLD_Price",
)

```



```

[44]: gld_cycle_df = calc_fed_cycle_asset_performance(
        fed_cycles=fed_cycles,
        cycle_labels=cycle_labels,
        fed_changes=fed_changes,
        monthly_df=gld_monthly,
    )

gld_cycle_df

```

```

[44]:
   Cycle  Start      End  Months  CumulativeReturn  \
0  Cycle 1 2004-11-01 2006-07-01     20           0.36
1  Cycle 2 2006-07-01 2007-07-01     12           0.05
2  Cycle 3 2007-07-01 2008-12-01     17           0.25
3  Cycle 4 2008-12-01 2015-11-01     83           0.36
4  Cycle 5 2015-11-01 2019-01-01     38           0.11
5  Cycle 6 2019-01-01 2019-07-01      6           0.10
6  Cycle 7 2019-07-01 2020-04-01      9           0.11
7  Cycle 8 2020-04-01 2022-02-01     22           0.14
8  Cycle 9 2022-02-01 2023-08-01     18           0.08
9  Cycle 10 2023-08-01 2024-08-01     12           0.24
10 Cycle 11 2024-08-01 2025-12-04     15           0.62

   CumulativeReturnPct  AverageMonthlyReturn  AverageMonthlyReturnPct  \
0                35.70                0.02                1.73

```

1	4.96	0.00	0.45
2	24.96	0.02	1.59
3	36.10	0.01	0.51
4	10.93	0.00	0.35
5	9.86	0.02	1.63
6	11.15	0.01	1.24
7	13.54	0.01	0.69
8	8.48	0.01	0.53
9	24.24	0.02	1.89
10	62.49	0.03	3.36

	AnnualizedReturn	AnnualizedReturnPct	Volatility	FedFundsChange	\
0	0.20	20.10	0.17	0.03	
1	0.05	4.96	0.11	0.00	
2	0.17	17.03	0.26	-0.05	
3	0.05	4.56	0.18	-0.00	
4	0.03	3.33	0.14	0.02	
5	0.21	20.68	0.12	0.00	
6	0.15	15.13	0.13	-0.02	
7	0.07	7.17	0.16	0.00	
8	0.06	5.58	0.14	0.05	
9	0.24	24.24	0.13	0.00	
10	0.47	47.46	0.14	-0.01	

	FedFundsChange_bps	FFR_AnnualizedChange	FFR_AnnualizedChange_bps	\
0	331.00	0.02	198.60	
1	2.00	0.00	2.00	
2	-510.00	-0.04	-360.00	
3	-4.00	-0.00	-0.58	
4	228.00	0.01	72.00	
5	0.00	0.00	0.00	
6	-235.00	-0.03	-313.33	
7	3.00	0.00	1.64	
8	525.00	0.03	350.00	
9	0.00	0.00	0.00	
10	-145.00	-0.01	-116.00	

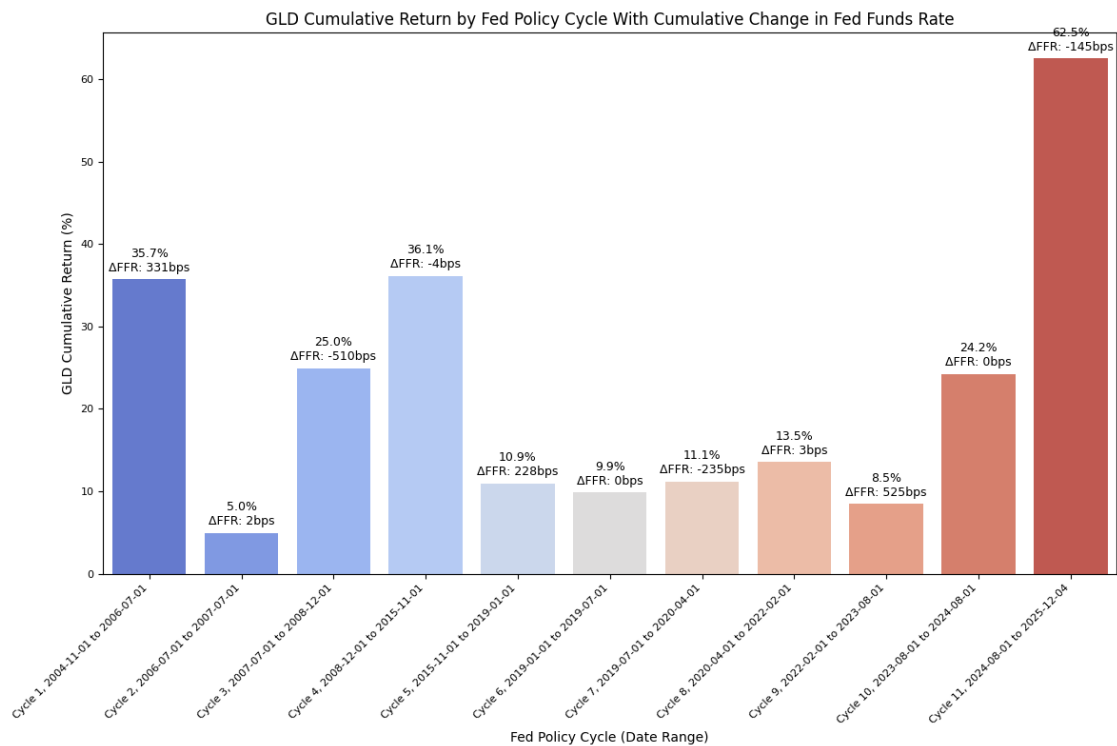
	Label
0	Cycle 1, 2004-11-01 to 2006-07-01
1	Cycle 2, 2006-07-01 to 2007-07-01
2	Cycle 3, 2007-07-01 to 2008-12-01
3	Cycle 4, 2008-12-01 to 2015-11-01
4	Cycle 5, 2015-11-01 to 2019-01-01
5	Cycle 6, 2019-01-01 to 2019-07-01
6	Cycle 7, 2019-07-01 to 2020-04-01
7	Cycle 8, 2020-04-01 to 2022-02-01
8	Cycle 9, 2022-02-01 to 2023-08-01

9 Cycle 10, 2023-08-01 to 2024-08-01
 10 Cycle 11, 2024-08-01 to 2025-12-04

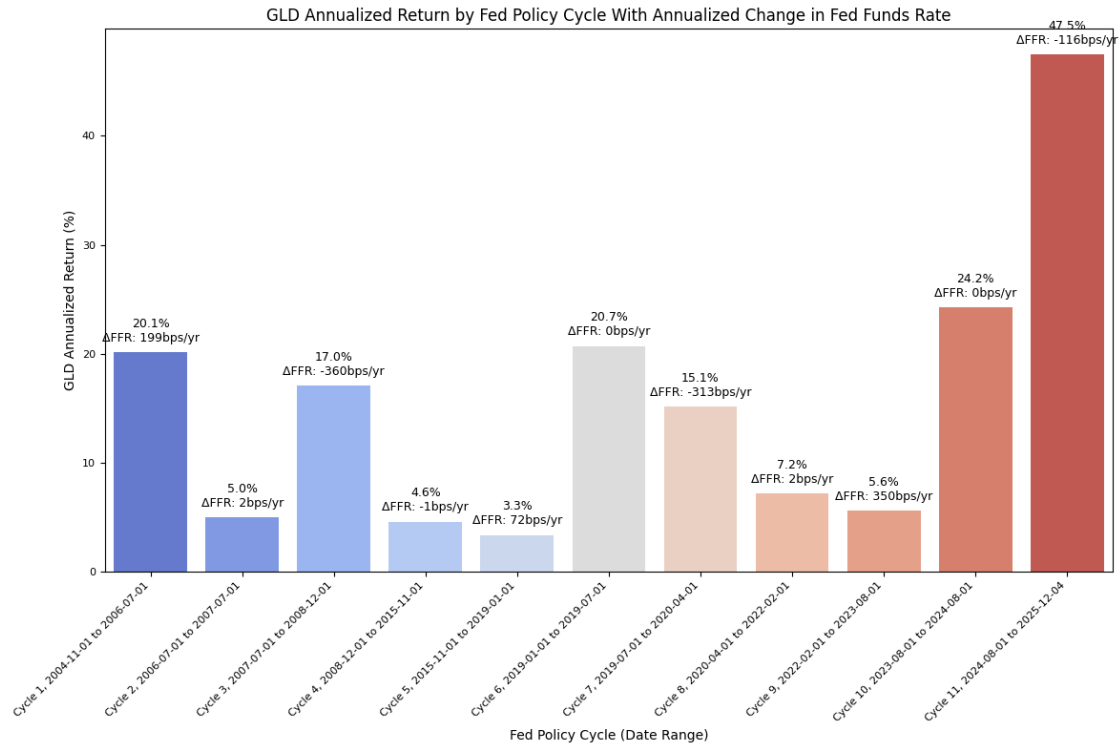
```
[45]: # Copy this <!-- INSERT_04_GLD_Cycle_DF_HERE --> to index_temp.md
export_track_md_deps(dep_file=dep_file, md_filename="04_GLD_Cycle_DF.md",
  ↪content=gld_cycle_df.to_markdown(floatfmt=".2f"))
```

Exported and tracked: 04_GLD_Cycle_DF.md

```
[46]: plot_bar_returns_ffr_change(
  cycle_df=gld_cycle_df,
  asset_label="GLD",
  annualized_or_cumulative="Cumulative",
  index_num="04",
)
```



```
[47]: plot_bar_returns_ffr_change(
  cycle_df=gld_cycle_df,
  asset_label="GLD",
  annualized_or_cumulative="Annualized",
  index_num="04",
)
```



```
[48]: df = gld_cycle_df

#####
### Don't modify below this line ###
#####

# Run OLS regression with statsmodels
X = df["FFR_AnnualizedChange_bps"]
y = df["AnnualizedReturnPct"]
X = sm.add_constant(X)
model = sm.OLS(y, X).fit()
print(model.summary())
print(f"Intercept: {model.params[0]}, Slope: {model.params[1]}") # Intercept,
    ↪ and slope

# Calc X and Y values for regression line
X_vals = np.linspace(X.min(), X.max(), 100)
Y_vals = model.params[0] + model.params[1] * X_vals
```

OLS Regression Results

```
=====
Dep. Variable:    AnnualizedReturnPct    R-squared:                0.084
Model:            OLS                    Adj. R-squared:          -0.018
```

```

Method:                Least Squares      F-statistic:           0.8274
Date:                  Thu, 04 Dec 2025    Prob (F-statistic):     0.387
Time:                  14:45:27            Log-Likelihood:        -42.830
No. Observations:      11                AIC:                   89.66
Df Residuals:          9                 BIC:                   90.46
Df Model:              1
Covariance Type:       nonrobust

```

```

=====
=====
              coef      std err          t      P>|t|      [0.025
0.975]
-----
const              15.1947      3.972      3.826      0.004      6.210
24.179
FFR_AnnualizedChange_bps -0.0187      0.021     -0.910      0.387     -0.065
0.028
=====
Omnibus:            8.035    Durbin-Watson:           0.915
Prob(Omnibus):       0.018    Jarque-Bera (JB):        3.686
Skew:                1.328    Prob(JB):                0.158
Kurtosis:            3.993    Cond. No.                193.
=====

```

Notes:

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

Intercept: 15.19466510751222, Slope: -0.018749970694326247

```

[49]: # Copy this <!-- INSERT_O4_GLD_Annualized_Regression_HERE --> to index_temp.md
export_track_md_deps(dep_file=dep_file,
    ↪md_filename="O4_GLD_Annualized_Regression.md",
    ↪content=sm_ols_summary_markdown(result=model,
    ↪file_path="O4_GLD_Annualized_Regression.md"))

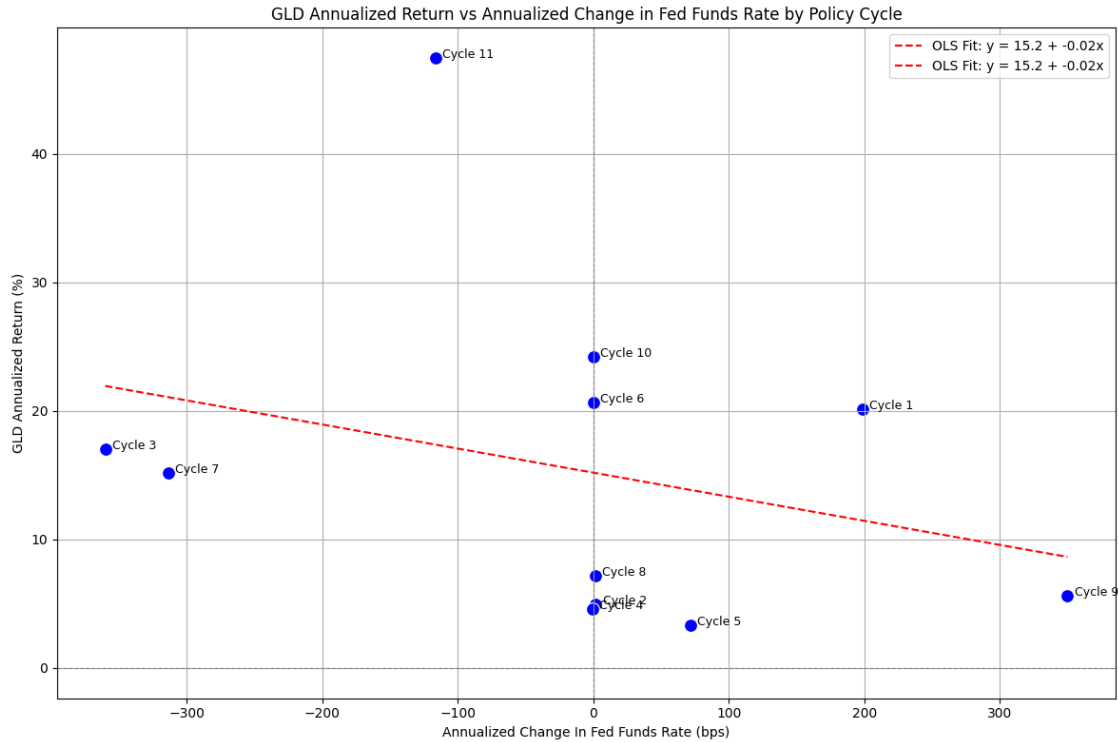
```

Exported and tracked: O4_GLD_Annualized_Regression.md

```

[50]: plot_scatter_regression_ffr_vs_returns(
    cycle_df=gld_cycle_df,
    asset_label="GLD",
    index_num="04",
    x_vals=X_vals,
    y_vals=Y_vals,
    intercept=model.params[0],
    slope=model.params[1],
)

```



1.7 Hybrid Portfolio

1.7.1 Asset Allocation

```
[51]: fed_cycles
```

```
[51]: [('2004-11-01', '2006-07-01'),
      ('2006-07-01', '2007-07-01'),
      ('2007-07-01', '2008-12-01'),
      ('2008-12-01', '2015-11-01'),
      ('2015-11-01', '2019-01-01'),
      ('2019-01-01', '2019-07-01'),
      ('2019-07-01', '2020-04-01'),
      ('2020-04-01', '2022-02-01'),
      ('2022-02-01', '2023-08-01'),
      ('2023-08-01', '2024-08-01'),
      ('2024-08-01', '2025-12-04')]
```

```
[52]: cycle_labels
```

```
[52]: ['Cycle 1',
      'Cycle 2',
      'Cycle 3',
      'Cycle 4',
```



```
'Cycle 5',
'Cycle 6',
'Cycle 7',
'Cycle 8',
'Cycle 9',
'Cycle 10',
'Cycle 11']
```

```
[53]: # Calculate cumulative returns and drawdown for SPY
spy_monthly['Cumulative_Return'] = (1 + spy_monthly['Monthly_Return']).
    ↪cumprod() - 1
spy_monthly['Cumulative_Return_Plus_One'] = 1 + spy_monthly['Cumulative_Return']
spy_monthly['Rolling_Max'] = spy_monthly['Cumulative_Return_Plus_One'].cummax()
spy_monthly['Drawdown'] = spy_monthly['Cumulative_Return_Plus_One'] / ↪
    ↪spy_monthly['Rolling_Max'] - 1
spy_monthly.drop(columns=['Cumulative_Return_Plus_One', 'Rolling_Max'], ↪
    ↪inplace=True)

# Calculate cumulative returns and drawdown for TLT
tlt_monthly['Cumulative_Return'] = (1 + tlt_monthly['Monthly_Return']).
    ↪cumprod() - 1
tlt_monthly['Cumulative_Return_Plus_One'] = 1 + tlt_monthly['Cumulative_Return']
tlt_monthly['Rolling_Max'] = tlt_monthly['Cumulative_Return_Plus_One'].cummax()
tlt_monthly['Drawdown'] = tlt_monthly['Cumulative_Return_Plus_One'] / ↪
    ↪tlt_monthly['Rolling_Max'] - 1
tlt_monthly.drop(columns=['Cumulative_Return_Plus_One', 'Rolling_Max'], ↪
    ↪inplace=True)

# Isolate the returns for SPY and TLT
spy_ret = spy_monthly['Monthly_Return']
tlt_ret = tlt_monthly['Monthly_Return']

# Create a blended portfolio based on Fed policy cycles
portfolio = (
    spy_ret[spy_ret.index <= "2007-07-01"]
    .combine_first(tlt_ret[(tlt_ret.index >= "2007-07-01") & (tlt_ret.index <= ↪
    ↪"2008-12-01")]))
    .combine_first(spy_ret[(spy_ret.index > "2008-12-01") & (spy_ret.index <= ↪
    ↪"2019-07-01")]))
    .combine_first(tlt_ret[(tlt_ret.index >= "2019-07-01") & (tlt_ret.index <= ↪
    ↪"2020-04-01")]))
    .combine_first(spy_ret[(spy_ret.index > "2020-04-01") & (spy_ret.index <= ↪
    ↪"2024-08-01")]))
    .combine_first(tlt_ret[tlr_ret.index > "2024-08-01"])
)
```

```

# Convert to DataFrame
portfolio_monthly = portfolio.to_frame(name="Portfolio_Monthly_Return")

# Calculate cumulative returns and drawdown for the portfolio
portfolio_monthly['Portfolio_Cumulative_Return'] = (1 +
    ↪portfolio_monthly['Portfolio_Monthly_Return']).cumprod() - 1
portfolio_monthly['Portfolio_Cumulative_Return_Plus_One'] = 1 +
    ↪portfolio_monthly['Portfolio_Cumulative_Return']
portfolio_monthly['Portfolio_Rolling_Max'] =
    ↪portfolio_monthly['Portfolio_Cumulative_Return_Plus_One'].cummax()
portfolio_monthly['Portfolio_Drawdown'] =
    ↪portfolio_monthly['Portfolio_Cumulative_Return_Plus_One'] /
    ↪portfolio_monthly['Portfolio_Rolling_Max'] - 1
portfolio_monthly.drop(columns=['Portfolio_Cumulative_Return_Plus_One',
    ↪'Portfolio_Rolling_Max'], inplace=True)

# Merge "spy_monthly" and "tlt_monthly" into "portfolio_monthly" to compare
    ↪cumulative returns
portfolio_monthly = portfolio_monthly.join(
    spy_monthly['Monthly_Return'].rename('SPY_Monthly_Return'),
    how='left'
).join(
    spy_monthly['Cumulative_Return'].rename('SPY_Cumulative_Return'),
    how='left'
).join(
    spy_monthly['Drawdown'].rename('SPY_Drawdown'),
    how='left'
).join(
    tlt_monthly['Monthly_Return'].rename('TLT_Monthly_Return'),
    how='left'
).join(
    tlt_monthly['Cumulative_Return'].rename('TLT_Cumulative_Return'),
    how='left'
).join(
    tlt_monthly['Drawdown'].rename('TLT_Drawdown'),
    how='left'
)

```

```
[54]: df_info(portfolio_monthly)
```

The columns, shape, and data types are:

```
<class 'pandas.core.frame.DataFrame'>
```

DatetimeIndex: 252 entries, 2004-11-30 to 2025-10-31

Freq: ME

Data columns (total 9 columns):

#	Column	Non-Null Count	Dtype
---	-----	-----	-----

0	Portfolio_Monthly_Return	251 non-null	float64
1	Portfolio_Cumulative_Return	251 non-null	float64
2	Portfolio_Drawdown	251 non-null	float64
3	SPY_Monthly_Return	251 non-null	float64
4	SPY_Cumulative_Return	251 non-null	float64
5	SPY_Drawdown	251 non-null	float64
6	TLT_Monthly_Return	251 non-null	float64
7	TLT_Cumulative_Return	251 non-null	float64
8	TLT_Drawdown	251 non-null	float64

dtypes: float64(9)

memory usage: 19.7 KB

None

The first 5 rows are:

	Portfolio_Monthly_Return	Portfolio_Cumulative_Return \
Date		
2004-11-30	NaN	NaN
2004-12-31	0.03	0.03
2005-01-31	-0.02	0.01
2005-02-28	0.02	0.03
2005-03-31	-0.02	0.01

	Portfolio_Drawdown	SPY_Monthly_Return	SPY_Cumulative_Return \
Date			
2004-11-30	NaN	NaN	NaN
2004-12-31	0.00	0.03	0.03
2005-01-31	-0.02	-0.02	0.01
2005-02-28	-0.00	0.02	0.03
2005-03-31	-0.02	-0.02	0.01

	SPY_Drawdown	TLT_Monthly_Return	TLT_Cumulative_Return \
Date			
2004-11-30	NaN	NaN	NaN
2004-12-31	0.00	0.03	0.03
2005-01-31	-0.02	0.04	0.06
2005-02-28	-0.00	-0.01	0.05
2005-03-31	-0.02	-0.00	0.04

	TLT_Drawdown
Date	
2004-11-30	NaN
2004-12-31	0.00
2005-01-31	0.00
2005-02-28	-0.01
2005-03-31	-0.02

The last 5 rows are:

	Portfolio_Monthly_Return	Portfolio_Cumulative_Return \
--	--------------------------	-------------------------------

Date			
2025-06-30		0.03	19.00
2025-07-31		-0.01	18.78
2025-08-31		0.00	18.78
2025-09-30		0.04	19.49
2025-10-31		0.01	19.77

	Portfolio_Drawdown	SPY_Monthly_Return	SPY_Cumulative_Return \
Date			
2025-06-30	-0.07	0.05	6.72
2025-07-31	-0.08	0.02	6.90
2025-08-31	-0.08	0.02	7.06
2025-09-30	-0.05	0.04	7.34
2025-10-31	-0.04	0.02	7.54

	SPY_Drawdown	TLT_Monthly_Return	TLT_Cumulative_Return \
Date			
2025-06-30	0.00	0.03	0.96
2025-07-31	0.00	-0.01	0.94
2025-08-31	0.00	0.00	0.94
2025-09-30	0.00	0.04	1.01
2025-10-31	0.00	0.01	1.04

	TLT_Drawdown
Date	
2025-06-30	-0.41
2025-07-31	-0.41
2025-08-31	-0.41
2025-09-30	-0.39
2025-10-31	-0.39

```
[55]: # Copy this <!-- INSERT_05_Portfolio_DF_HERE --> to index_temp.md
export_track_md_deps(dep_file=dep_file, md_filename="05_Portfolio_DF.md",
↳content=df_info_markdown(df=portfolio_monthly, decimal_places=3))
```

Exported and tracked: 05_Portfolio_DF.md

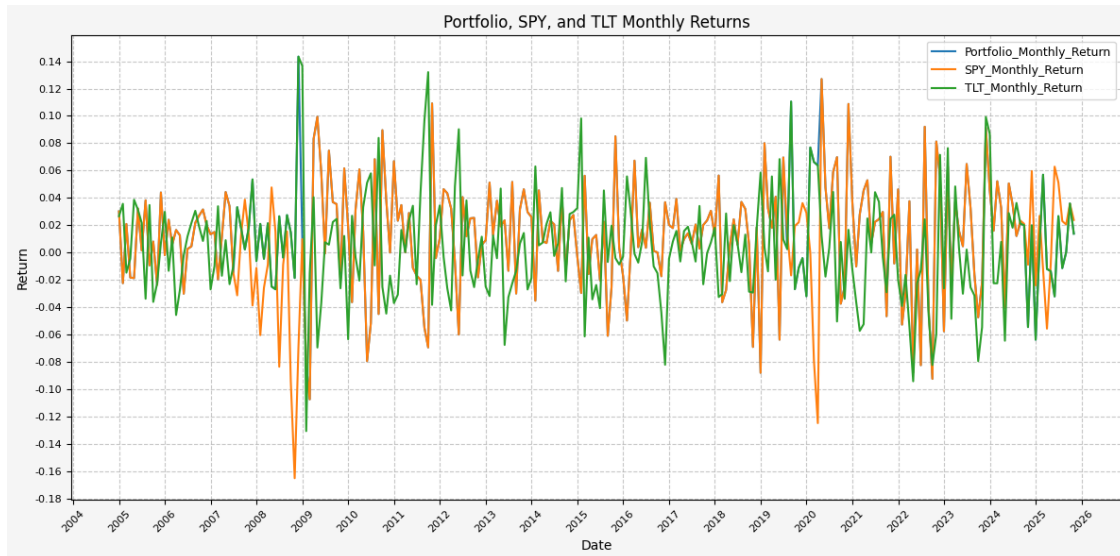
1.7.2 Performance Statistics

```
[56]: plot_timeseries(
    price_df=portfolio_monthly,
    plot_start_date=start_date,
    plot_end_date=end_date,
    plot_columns=["Portfolio_Monthly_Return", "SPY_Monthly_Return",
↳"TLT_Monthly_Return"],
    title="Portfolio, SPY, and TLT Monthly Returns",
    x_label="Date",
    x_format="Year",
```

```

y_label="Return",
y_format="Decimal",
y_format_decimal_places=2,
y_tick_spacing=0.02,
grid=True,
legend=True,
export_plot=True,
plot_file_name="05_Monthly>Returns",
)

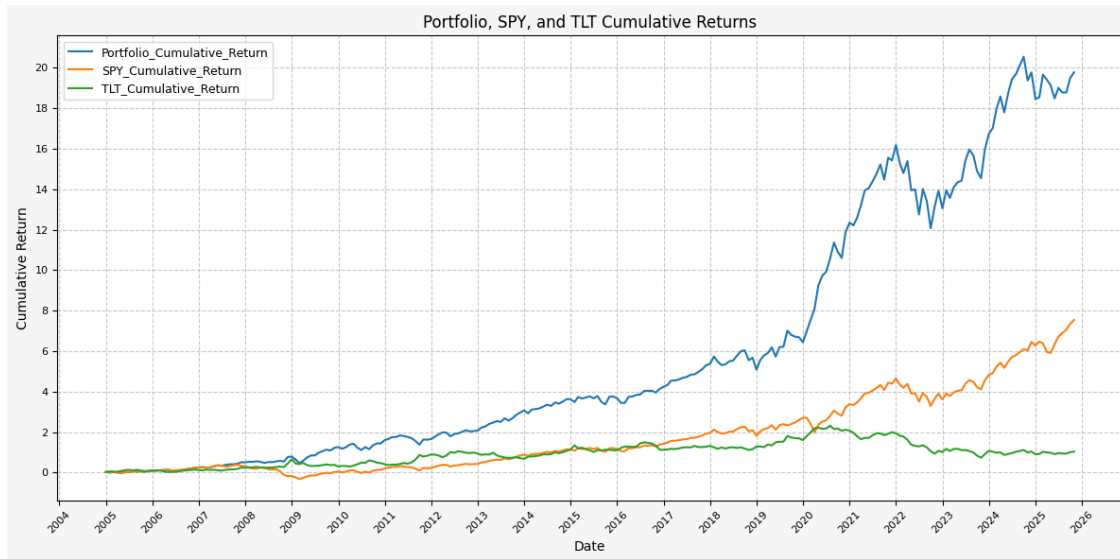
```



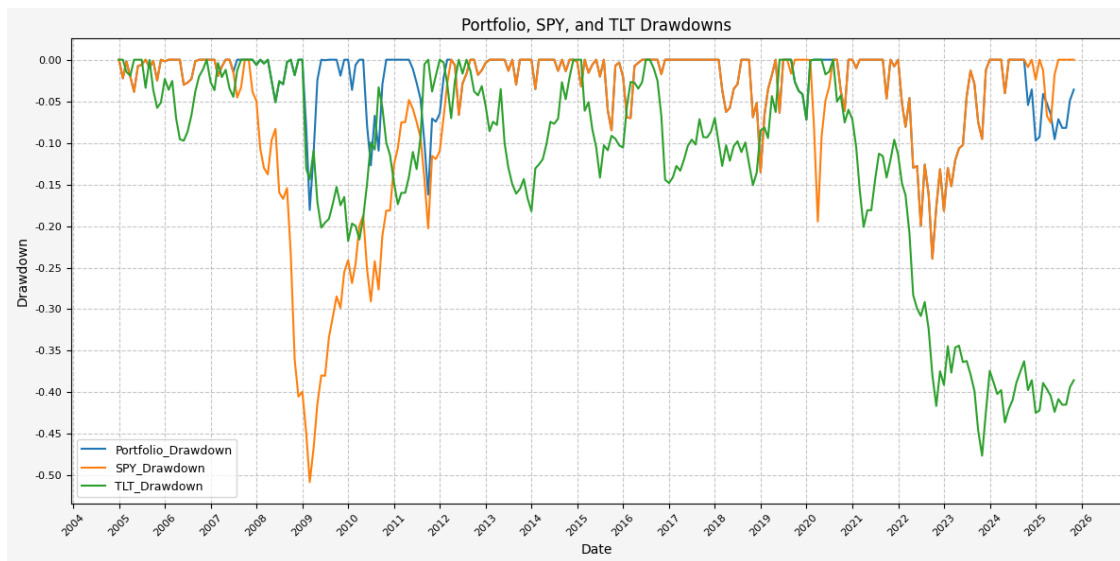
```

[57]: plot_timeseries(
    price_df=portfolio_monthly,
    plot_start_date=start_date,
    plot_end_date=end_date,
    plot_columns=["Portfolio_Cumulative_Return", "SPY_Cumulative_Return",
↪ "TLT_Cumulative_Return"],
    title="Portfolio, SPY, and TLT Cumulative Returns",
    x_label="Date",
    x_format="Year",
    y_label="Cumulative Return",
    y_format="Decimal",
    y_format_decimal_places=0,
    y_tick_spacing=2,
    grid=True,
    legend=True,
    export_plot=True,
    plot_file_name="05_Cumulative>Returns",
)

```



```
[58]: plot_timeseries(
    price_df=portfolio_monthly,
    plot_start_date=start_date,
    plot_end_date=end_date,
    plot_columns=["Portfolio_Drawdown", "SPY_Drawdown", "TLT_Drawdown"],
    title="Portfolio, SPY, and TLT Drawdowns",
    x_label="Date",
    x_format="Year",
    y_label="Drawdown",
    y_format="Decimal",
    y_format_decimal_places=2,
    y_tick_spacing=0.05,
    grid=True,
    legend=True,
    export_plot=True,
    plot_file_name="05_Drawdowns",
)
```



```
[59]: port_sum_stats = summary_stats(
    fund_list=["Portfolio", "SPY", "TLT"],
    df=portfolio_monthly[["Portfolio_Monthly_Return"]],
    period="Monthly",
    use_calendar_days=False,
    excel_export=False,
    pickle_export=False,
    output_confirmation=False,
)

spy_sum_stats = summary_stats(
    fund_list=["Portfolio", "SPY", "TLT"],
    df=portfolio_monthly[["SPY_Monthly_Return"]],
    period="Monthly",
    use_calendar_days=False,
    excel_export=False,
    pickle_export=False,
    output_confirmation=False,
)

tlt_sum_stats = summary_stats(
    fund_list=["Portfolio", "SPY", "TLT"],
    df=portfolio_monthly[["TLT_Monthly_Return"]],
    period="Monthly",
    use_calendar_days=False,
    excel_export=False,
    pickle_export=False,
    output_confirmation=False,
```

```
)  
  
sum_stats = port_sum_stats.combine_first(spy_sum_stats).  
    ↪ combine_first(tlt_sum_stats)
```

```
[60]: # Copy this <!-- INSERT_05_Portfolio_Stats_DF_HERE --> to index_temp.md  
export_track_md_deps(dep_file=dep_file, md_filename="05_Portfolio_Stats_DF.md",  
    ↪ content=sum_stats.to_markdown(floatfmt=".3f"))
```

Exported and tracked: 05_Portfolio_Stats_DF.md