

trend-following-rsi-minute-level-crypto-part-1-trading-strategy

November 29, 2025

1 Combining Trend Following With RSI Using Crypto Data

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

# 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
try:
    # Works for .py files
    current_directory = Path(__file__).resolve().parent
except NameError:
    # Fallback for notebooks / interactive shells
    current_directory = Path(os.getcwd()).resolve()

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/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 add_rsi_ma_bb import add_rsi_ma_bb
from analyze_trades import analyze_trades
from backtest_rsi_multi_asset_strategy import backtest_rsi_multi_asset_strategy
from compute_daily_performance import compute_daily_performance
from create_signals import create_signals
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_crypto_data import load_crypto_data
from load_data import load_data
from pandas_set_decimal_places import pandas_set_decimal_places
from plot_multi_asset_equity_and_drawdown import plot_multi_asset_equity_and_drawdown
from summary_stats import summary_stats
```

2 Initial Variables

```
[5]: TICKERS = ["BTC-USD"]
MA_DAYS = [7]
INITIAL_CAPITAL = 10_000
RSI_PERIOD = 20
RSI_THRESHOLD = 20
TRAILING_STOP_PCT = 0.02
ORDER_ENTRY = "limit"
USE_BBANDS = True
BB_WINDOW = 20
BB_NUM_STD = 2.0
BB_RULE = "touch_lower" # {"touch_lower", "cross_up_from_below", "below_lower"}
START_DATE = "2025-01-01"
END_DATE = "2025-07-31"
```

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

2.1 Load Crypto Data

```
[7]: crypto_prices_df = load_crypto_data(  
    tickers=TICKERS,  
    base_directory=DATA_DIR,  
    start_date=START_DATE,  
    end_date=END_DATE,  
)  
  
crypto_prices_df
```

```
[7]:
```

	Date	BTC-USD_low	BTC-USD_high	BTC-USD_open	\
0	2025-01-01 00:00:00	93324.33	93408.72	93347.59	
1	2025-01-01 00:01:00	93388.22	93440.00	93408.63	
2	2025-01-01 00:02:00	93426.80	93507.45	93440.00	
3	2025-01-01 00:03:00	93426.90	93489.13	93489.13	
4	2025-01-01 00:04:00	93427.16	93474.93	93474.93	
...	
303836	2025-07-30 23:56:00	117856.59	117880.00	117856.59	
303837	2025-07-30 23:57:00	117836.00	117867.37	117867.35	
303838	2025-07-30 23:58:00	117829.75	117836.01	117836.00	
303839	2025-07-30 23:59:00	117830.14	117830.15	117830.15	
303840	2025-07-31 00:00:00	117830.15	117869.50	117830.15	
	BTC-USD_close	BTC-USD_volume			
0	93408.63	3.58			
1	93440.00	4.56			
2	93489.12	15.22			
3	93477.31	8.17			
4	93458.33	3.30			
...			
303836	117867.34	0.27			
303837	117836.00	1.36			
303838	117830.15	0.70			
303839	117830.15	0.78			
303840	117833.75	1.23			

[303841 rows x 6 columns]

2.2 Add RSI, MA, BB

```
[8]: crypto_prices_technical_df = add_rsi_ma_bb(  
    tickers=TICKERS,  
    data=crypto_prices_df,  
    rsi_period=RSI_PERIOD,  
    ma_days=MA_DAYS,  
    bb_window=BB_WINDOW,
```

```

        bb_num_std=BB_NUM_STD,
    )

crypto_prices_technical_df

```

[8]:

	Date	BTC-USD_low	BTC-USD_high	BTC-USD_open	\
0	2025-01-01 00:00:00	93324.33	93408.72	93347.59	
1	2025-01-01 00:01:00	93388.22	93440.00	93408.63	
2	2025-01-01 00:02:00	93426.80	93507.45	93440.00	
3	2025-01-01 00:03:00	93426.90	93489.13	93489.13	
4	2025-01-01 00:04:00	93427.16	93474.93	93474.93	
...	
303836	2025-07-30 23:56:00	117856.59	117880.00	117856.59	
303837	2025-07-30 23:57:00	117836.00	117867.37	117867.35	
303838	2025-07-30 23:58:00	117829.75	117836.01	117836.00	
303839	2025-07-30 23:59:00	117830.14	117830.15	117830.15	
303840	2025-07-31 00:00:00	117830.15	117869.50	117830.15	

	BTC-USD_close	BTC-USD_volume	BTC-USD_close_prev	BTC-USD_RSI	\
0	93408.63	3.58	NaN	NaN	
1	93440.00	4.56	93408.63	100.00	
2	93489.12	15.22	93440.00	100.00	
3	93477.31	8.17	93489.12	98.11	
4	93458.33	3.30	93477.31	95.07	
...	
303836	117867.34	0.27	117856.60	64.10	
303837	117836.00	1.36	117867.34	60.30	
303838	117830.15	0.70	117836.00	59.60	
303839	117830.15	0.78	117830.15	59.60	
303840	117833.75	1.23	117830.15	59.92	

	BTC-USD_RSI_prev	BTC-USD_MA_7d	BTC-USD_MA_7d_prev	\
0	NaN	93408.63	NaN	
1	NaN	93424.32	93408.63	
2	100.00	93445.92	93424.32	
3	100.00	93453.76	93445.92	
4	98.11	93454.68	93453.76	
...	
303836	63.35	117993.77	117993.85	
303837	64.10	117993.68	117993.77	
303838	60.30	117993.58	117993.68	
303839	59.60	117993.49	117993.58	
303840	59.60	117993.40	117993.49	

	BTC-USD_BB_MID_prev	BTC-USD_BB_STD_prev	BTC-USD_BB_UPPER_prev	\
0	NaN	NaN	NaN	
1	NaN	NaN	NaN	

2	NaN	NaN	NaN
3	NaN	NaN	NaN
4	NaN	NaN	NaN
...
303836	117789.88	79.61	117949.11
303837	117800.17	75.35	117950.87
303838	117807.09	71.69	117950.47
303839	117813.50	67.40	117948.30
303840	117819.52	62.87	117945.27

	BTC-USD_BB_LOWER_prev	BTC-USD_BB_Z_prev
0	NaN	NaN
1	NaN	NaN
2	NaN	NaN
3	NaN	NaN
4	NaN	NaN
...
303836	117630.66	0.84
303837	117649.47	0.89
303838	117663.72	0.40
303839	117678.70	0.25
303840	117693.77	0.17

[303841 rows x 16 columns]

2.3 Create Signals

```
[9]: signals_df = create_signals(
    tickers=TICKERS,
    data=crypto_prices_technical_df,
    use_rsi=True,
    rsi_threshold=RSI_THRESHOLD,
    use_ma=True,
    ma_days=MA_DAYS,
    use_bbands=USE_BBANDS,
    bb_rule=BB_RULE,
)

signals_df
```

```
[9]:
```

	Date	open	high	low	close	close_prev \
0	2025-01-05 08:02:00	98100.00	98112.02	98072.34	98072.35	98105.39
1	2025-01-05 08:03:00	98072.35	98096.66	97934.43	97975.45	98072.35
2	2025-01-05 08:04:00	97975.45	98012.31	97932.20	97947.22	97975.45
3	2025-01-05 08:05:00	97940.09	97989.60	97932.20	97933.42	97947.22
4	2025-01-05 08:06:00	97946.43	98012.63	97932.20	98003.88	97933.42
..

```

84 2025-07-17 20:06:00 118598.54 118724.51 118598.53 118628.10 118598.54
85 2025-07-28 07:06:00 119100.00 119121.59 119069.00 119071.52 119099.99
86 2025-07-28 07:07:00 119071.52 119087.01 118965.47 118965.49 119071.52
87 2025-07-28 07:08:00 118965.49 119018.39 118929.75 118940.00 118965.49
88 2025-07-28 07:09:00 118940.01 118970.36 118911.11 118957.25 118940.00

```

	asset	ma_passes	allocation_pct	bb_rule	bb_mid_prev	bb_up_prev \
0	BTC-USD	1	1.00	touch_lower	98270.67	98376.47
1	BTC-USD	1	1.00	touch_lower	98258.25	98393.51
2	BTC-USD	1	1.00	touch_lower	98241.00	98422.84
3	BTC-USD	1	1.00	touch_lower	98222.28	98442.26
4	BTC-USD	1	1.00	touch_lower	98203.55	98454.41
..
84	BTC-USD	1	1.00	touch_lower	119094.52	119525.60
85	BTC-USD	1	1.00	touch_lower	119443.45	119655.58
86	BTC-USD	1	1.00	touch_lower	119421.85	119689.04
87	BTC-USD	1	1.00	touch_lower	119394.70	119727.19
88	BTC-USD	1	1.00	touch_lower	119366.99	119752.68

	bb_low_prev	bb_z_prev
0	98164.87	-3.12
1	98122.99	-2.75
2	98059.16	-2.92
3	98002.30	-2.50
4	97952.68	-2.15
..
84	118663.45	-2.30
85	119231.33	-3.24
86	119154.66	-2.62
87	119062.20	-2.58
88	118981.29	-2.21

[89 rows x 14 columns]

2.4 Run Backtest

```

[10]: trades_df = backtest_rsi_multi_asset_strategy(
    tickers=TICKERS,
    prices=crypto_prices_technical_df,
    signals=signals_df,
    initial_capital=INITIAL_CAPITAL,
    rsi_threshold=RSI_THRESHOLD,
    trailing_stop_pct=TRAILING_STOP_PCT,
    ma_days=MA_DAYS,
    order_entry=ORDER_ENTRY,
    trading_fees=True,
    trade_taker_fee=0.0020,

```

```

    trade_maker_fee=0.0010,
)

trades_df

```

```

[10]:
   asset      entry_time entry_type entry_price      exit_time \
0  BTC-USD 2025-01-05 08:02:00    limit    98105.39 2025-01-07 11:41:00
1  BTC-USD 2025-01-07 11:42:00    limit   100668.90 2025-01-07 15:07:00
2  BTC-USD 2025-01-18 03:42:00    limit  103330.59 2025-01-19 09:21:00
3  BTC-USD 2025-01-19 21:29:00    limit  104569.14 2025-01-19 22:03:00
4  BTC-USD 2025-03-03 06:41:00    limit   92098.84 2025-03-03 14:38:00
5  BTC-USD 2025-03-03 14:58:00    limit   89380.57 2025-03-03 18:07:00
6  BTC-USD 2025-03-06 17:02:00    limit   88726.66 2025-03-07 00:15:00
7  BTC-USD 2025-03-14 20:22:00    limit   83858.35 2025-03-16 11:16:00
8  BTC-USD 2025-03-16 19:17:00    limit   82816.04 2025-03-18 01:33:00
9  BTC-USD 2025-03-20 10:16:00    limit   85375.52 2025-03-20 15:54:00
10 BTC-USD 2025-03-22 06:34:00    limit   84138.15 2025-03-25 01:30:00
11 BTC-USD 2025-03-25 01:31:00    limit   86932.96 2025-03-26 13:58:00
12 BTC-USD 2025-03-28 04:41:00    limit   86215.44 2025-03-28 14:07:00
13 BTC-USD 2025-04-10 07:35:00    limit   81370.40 2025-04-10 13:43:00
14 BTC-USD 2025-04-13 19:40:00    limit   83555.83 2025-04-14 15:40:00
15 BTC-USD 2025-04-21 11:17:00    limit   86913.63 2025-04-21 17:02:00
16 BTC-USD 2025-04-21 17:04:00    limit   86743.90 2025-04-22 21:49:00
17 BTC-USD 2025-04-24 21:29:00    limit   93110.21 2025-04-26 11:42:00
18 BTC-USD 2025-05-03 23:28:00    limit   96019.87 2025-05-04 23:30:00
19 BTC-USD 2025-05-21 17:29:00    limit  107202.23 2025-05-23 11:48:00
20 BTC-USD 2025-05-23 11:49:00    limit  109693.02 2025-05-23 12:15:00
21 BTC-USD 2025-06-16 22:43:00    limit  107854.92 2025-06-17 10:38:00
22 BTC-USD 2025-06-28 13:28:00    limit  107254.84 2025-07-01 09:20:00
23 BTC-USD 2025-07-02 11:38:00    limit  107593.15 2025-07-04 13:34:00
24 BTC-USD 2025-07-12 20:54:00    limit  117247.93 2025-07-14 14:42:00
25 BTC-USD 2025-07-15 11:17:00    limit  116541.20 2025-07-15 14:54:00
26 BTC-USD 2025-07-15 14:56:00    limit  115981.01 2025-07-17 01:50:00
27 BTC-USD 2025-07-17 20:06:00    limit  118598.54 2025-07-18 08:41:00
28 BTC-USD 2025-07-28 07:06:00    limit  119099.99 2025-07-30 18:49:00

   exit_type  exit_price  quantity  allocation_pct    pnl  return \
0  trailing stop  100681.27     0.10           1.00   231.81    0.02
1  trailing stop   98948.87     0.10           1.00  -204.96   -0.02
2  trailing stop  103390.00     0.10           1.00   -24.30   -0.00
3  trailing stop  103093.73     0.10           1.00  -170.69   -0.02
4  trailing stop   91840.23     0.11           1.00   -56.99   -0.01
5  trailing stop   89429.23     0.11           1.00   -23.99   -0.00
6  trailing stop   89454.40     0.11           1.00    50.51    0.01
7  trailing stop   82996.62     0.12           1.00  -129.79   -0.01
8  trailing stop   83063.29     0.12           1.00    -0.20   -0.00
9  trailing stop   84780.94     0.11           1.00   -96.14   -0.01

```


10	trailing stop	87028.55	0.11	1.00	299.26	0.03
11	trailing stop	86812.65	0.11	1.00	-43.22	-0.00
12	trailing stop	84617.36	0.11	1.00	-211.15	-0.02
13	trailing stop	80750.93	0.12	1.00	-101.85	-0.01
14	trailing stop	84101.70	0.11	1.00	33.47	0.00
15	trailing stop	86804.11	0.11	1.00	-40.63	-0.00
16	exit at open (gap)	91955.78	0.11	1.00	541.25	0.06
17	trailing stop	94056.81	0.11	1.00	71.76	0.01
18	trailing stop	94386.26	0.11	1.00	-202.07	-0.02
19	trailing stop	109760.00	0.09	1.00	206.29	0.02
20	trailing stop	107715.51	0.09	1.00	-212.40	-0.02
21	trailing stop	105884.10	0.09	1.00	-210.37	-0.02
22	trailing stop	106624.00	0.09	1.00	-86.00	-0.01
23	trailing stop	108378.20	0.09	1.00	41.15	0.00
24	trailing stop	120766.45	0.08	1.00	260.09	0.03
25	trailing stop	116095.06	0.09	1.00	-67.60	-0.01
26	trailing stop	117732.10	0.08	1.00	118.79	0.01
27	trailing stop	118578.78	0.08	1.00	-31.54	-0.00
28	trailing stop	116899.08	0.08	1.00	-212.94	-0.02

	cash	entry_fee	exit_fee	cumulative_pnl	equity	cumulative_return
0	10231.81	9.99	20.50	231.81	10231.81	0.02
1	10026.84	10.22	20.09	26.84	10026.84	0.00
2	10002.54	10.02	20.05	2.54	10002.54	0.00
3	9831.86	9.99	19.70	-168.14	9831.86	-0.02
4	9774.86	9.82	19.59	-225.14	9774.86	-0.02
5	9750.88	9.77	19.54	-249.12	9750.88	-0.02
6	9801.39	9.74	19.64	-198.61	9801.39	-0.02
7	9671.60	9.79	19.38	-328.40	9671.60	-0.03
8	9671.40	9.66	19.38	-328.60	9671.40	-0.03
9	9575.26	9.66	19.19	-424.74	9575.26	-0.04
10	9874.52	9.57	19.79	-125.48	9874.52	-0.01
11	9831.30	9.86	19.70	-168.70	9831.30	-0.02
12	9620.15	9.82	19.28	-379.85	9620.15	-0.04
13	9518.30	9.61	19.07	-481.70	9518.30	-0.05
14	9551.77	9.51	19.14	-448.23	9551.77	-0.04
15	9511.14	9.54	19.06	-488.86	9511.14	-0.05
16	10052.39	9.50	20.15	52.39	10052.39	0.01
17	10124.15	10.04	20.29	124.15	10124.15	0.01
18	9922.08	10.11	19.88	-77.92	9922.08	-0.01
19	10128.37	9.91	20.30	128.37	10128.37	0.01
20	9915.97	10.12	19.87	-84.03	9915.97	-0.01
21	9705.60	9.91	19.45	-294.40	9705.60	-0.03
22	9619.60	9.70	19.28	-380.40	9619.60	-0.04
23	9660.75	9.61	19.36	-339.25	9660.75	-0.03
24	9920.84	9.65	19.88	-79.16	9920.84	-0.01
25	9853.24	9.91	19.75	-146.76	9853.24	-0.01

26	9972.03	9.84	19.98	-27.97	9972.03	-0.00
27	9940.49	9.96	19.92	-59.51	9940.49	-0.01
28	9727.55	9.93	19.49	-272.45	9727.55	-0.03

2.5 Calc Trading Volumes

```
[11]: entry_trades_df = trades_df[["entry_time", "entry_price", "quantity"]]
entry_trades_df["entry_trading_volume"] = entry_trades_df["entry_price"] *
    ↪entry_trades_df["quantity"]
entry_trades_df = entry_trades_df.resample("D", on="entry_time").sum()
entry_trades_df = entry_trades_df[["entry_trading_volume"]]
```

```
[12]: exit_trades_df = trades_df[["exit_time", "exit_price", "quantity"]]
exit_trades_df["exit_trading_volume"] = exit_trades_df["exit_price"] *
    ↪exit_trades_df["quantity"]
exit_trades_df = exit_trades_df.resample("D", on="exit_time").sum()
exit_trades_df = exit_trades_df[["exit_trading_volume"]]
```

```
[13]: all_trades_df = entry_trades_df.join(exit_trades_df, how="outer")
all_trades_df["total_trading_volume"] = all_trades_df["entry_trading_volume"]
    ↪fillna(0) + all_trades_df["exit_trading_volume"].fillna(0)
all_trades_df["rolling_30d_volume"] = all_trades_df
    ↪rolling(window="30D")["total_trading_volume"].sum()
all_trades_df.head(45)
```

```
[13]:
```

	entry_trading_volume	exit_trading_volume	total_trading_volume \
2025-01-05	9990.01	NaN	9990.01
2025-01-06	0.00	NaN	0.00
2025-01-07	10221.58	20299.25	30520.83
2025-01-08	0.00	0.00	0.00
2025-01-09	0.00	0.00	0.00
2025-01-10	0.00	0.00	0.00
2025-01-11	0.00	0.00	0.00
2025-01-12	0.00	0.00	0.00
2025-01-13	0.00	0.00	0.00
2025-01-14	0.00	0.00	0.00
2025-01-15	0.00	0.00	0.00
2025-01-16	0.00	0.00	0.00
2025-01-17	0.00	0.00	0.00
2025-01-18	10016.83	0.00	10016.83
2025-01-19	9992.55	19874.14	29866.69
2025-01-20	0.00	0.00	0.00
2025-01-21	0.00	0.00	0.00
2025-01-22	0.00	0.00	0.00
2025-01-23	0.00	0.00	0.00
2025-01-24	0.00	0.00	0.00
2025-01-25	0.00	0.00	0.00

2025-01-26	0.00	0.00	0.00
2025-01-27	0.00	0.00	0.00
2025-01-28	0.00	0.00	0.00
2025-01-29	0.00	0.00	0.00
2025-01-30	0.00	0.00	0.00
2025-01-31	0.00	0.00	0.00
2025-02-01	0.00	0.00	0.00
2025-02-02	0.00	0.00	0.00
2025-02-03	0.00	0.00	0.00
2025-02-04	0.00	0.00	0.00
2025-02-05	0.00	0.00	0.00
2025-02-06	0.00	0.00	0.00
2025-02-07	0.00	0.00	0.00
2025-02-08	0.00	0.00	0.00
2025-02-09	0.00	0.00	0.00
2025-02-10	0.00	0.00	0.00
2025-02-11	0.00	0.00	0.00
2025-02-12	0.00	0.00	0.00
2025-02-13	0.00	0.00	0.00
2025-02-14	0.00	0.00	0.00
2025-02-15	0.00	0.00	0.00
2025-02-16	0.00	0.00	0.00
2025-02-17	0.00	0.00	0.00
2025-02-18	0.00	0.00	0.00

	rolling_30d_volume
2025-01-05	9990.01
2025-01-06	9990.01
2025-01-07	40510.84
2025-01-08	40510.84
2025-01-09	40510.84
2025-01-10	40510.84
2025-01-11	40510.84
2025-01-12	40510.84
2025-01-13	40510.84
2025-01-14	40510.84
2025-01-15	40510.84
2025-01-16	40510.84
2025-01-17	40510.84
2025-01-18	50527.67
2025-01-19	80394.36
2025-01-20	80394.36
2025-01-21	80394.36
2025-01-22	80394.36
2025-01-23	80394.36
2025-01-24	80394.36
2025-01-25	80394.36

2025-01-26	80394.36
2025-01-27	80394.36
2025-01-28	80394.36
2025-01-29	80394.36
2025-01-30	80394.36
2025-01-31	80394.36
2025-02-01	80394.36
2025-02-02	80394.36
2025-02-03	80394.36
2025-02-04	70404.35
2025-02-05	70404.35
2025-02-06	39883.52
2025-02-07	39883.52
2025-02-08	39883.52
2025-02-09	39883.52
2025-02-10	39883.52
2025-02-11	39883.52
2025-02-12	39883.52
2025-02-13	39883.52
2025-02-14	39883.52
2025-02-15	39883.52
2025-02-16	39883.52
2025-02-17	29866.69
2025-02-18	0.00

2.6 Compute Daily Performance

```
[14]: daily_perf_df = compute_daily_performance(
    tickers=TICKERS,
    data=crypto_prices_technical_df,
    trades=trades_df,
    initial_capital=INITIAL_CAPITAL,
)

daily_perf_df
```

```
[14]:
```

	cash	BTC-USD_qty	BTC-USD_close	BTC-USD_position	equity \
Date					
2025-01-01	10000.00	0.00	94383.59	0.00	10000.00
2025-01-02	10000.00	0.00	96903.19	0.00	10000.00
2025-01-03	10000.00	0.00	98136.51	0.00	10000.00
2025-01-04	10000.00	0.00	98209.85	0.00	10000.00
2025-01-05	0.00	0.10	98345.33	10014.44	10014.44
...
2025-07-27	9940.49	0.00	119465.52	0.00	9940.49
2025-07-28	0.00	0.08	118070.59	9844.73	9844.73
2025-07-29	0.00	0.08	117933.39	9833.29	9833.29

2025-07-30	9727.55	0.00	117830.15	0.00	9727.55
2025-07-31	9727.55	0.00	117833.75	0.00	9727.55

	BTC-USD_return	BTC-USD_cum_return	BTC-USD_drawdown	Return	\
Date					
2025-01-01	0.00	0.00	0.00	0.00	
2025-01-02	0.03	0.03	0.00	0.00	
2025-01-03	0.01	0.04	0.00	0.00	
2025-01-04	0.00	0.04	0.00	0.00	
2025-01-05	0.00	0.04	0.00	0.00	
...	
2025-07-27	0.01	0.27	-0.00	0.00	
2025-07-28	-0.01	0.25	-0.02	-0.01	
2025-07-29	-0.00	0.25	-0.02	-0.00	
2025-07-30	-0.00	0.25	-0.02	-0.01	
2025-07-31	0.00	0.25	-0.02	0.00	

	Cum_Return	Drawdown
Date		
2025-01-01	0.00	0.00
2025-01-02	0.00	0.00
2025-01-03	0.00	0.00
2025-01-04	0.00	0.00
2025-01-05	0.00	0.00
...
2025-07-27	-0.01	-0.05
2025-07-28	-0.02	-0.05
2025-07-29	-0.02	-0.06
2025-07-30	-0.03	-0.07
2025-07-31	-0.03	-0.07

[212 rows x 11 columns]

2.7 Summary Stats

```
[15]: sum_stats_df = summary_stats(
    fund_list=TICKERS,
    df=daily_perf_df[['Return']],
    period="Daily",
    use_calendar_days=True,
    excel_export=False,
    pickle_export=False,
    output_confirmation=True,
)

sum_stats_df
```

Summary stats complete for BTC-USD

```
[15]:      Annualized Mean  Annualized Volatility  Annualized Sharpe Ratio  CAGR  \
Return      -0.04              0.15              -0.23 -0.05

      Daily Max Return  Daily Max Return (Date)  Daily Min Return  \
Return      0.05              2025-04-22              -0.04

      Daily Min Return (Date)  Max Drawdown      Peak      Trough  \
Return      2025-05-23      -0.09  2025-01-06  2025-04-10

      Recovery Date  Days to Recover  MAR Ratio
Return      NaT              NaN      -0.54
```

2.8 Plots

```
[16]: # Create MA_Days string for filename
if MA_DAYS == []:
    temp_ma_days = "[0]"
else:
    temp_ma_days = MA_DAYS

# Base title name
base_title_name = f"_{START_DATE}_{END_DATE}_{TICKERS}_MA-{temp_ma_days}_RP-{RSI_PERIOD}_RT-{RSI_THRESHOLD}_TS"

# Create title name
if USE_BBANDS == True:
    bb_part = f"BBR-{BB_RULE}_BBW-{BB_WINDOW}_BBS-{BB_NUM_STD}"
    title_name = f"{base_title_name}_{bb_part}"
else:
    title_name = f"{base_title_name}"

[17]: plot_multi_asset_equity_and_drawdown(
    tickers=TICKERS,
    daily_perf=daily_perf_df,
    trades=trades_df,
    data=crypto_prices_technical_df,
    title=title_name,
    show_plot=True,
    export_plot=False,
)
```



2.9 Analyze Trades

```
[18]: analyze_trades(
    trades_df=trades_df,
    daily_perf_df=daily_perf_df,
    print_summary=True,
)
```

Trade Summary:
 Total Trades: 29
 Win Rate (%): 34.48%
 Total Return (%): -2.72%
 Average Return Per Trade: -0.08%
 Max Trade Gain (%): 5.69%
 Max Trade Loss (%): -2.15%
 Total PnL (\$): \$-272.45
 Average PnL Per Trade (\$): \$-9.39
 Max Trade Gain (\$): \$541.25
 Max Trade Loss (\$): \$-212.94
 Max Drawdown (%): -8.61%

```
[18]: (29,
       0.3448275862068966,
       np.float64(-0.02724480895281578),
       np.float64(-0.0007879214312853157),
       np.float64(0.056906459725102),
       np.float64(-0.021477397402748958),
       np.float64(-272.448089528154),
       np.float64(-9.39476170786738),
       np.float64(541.2454281653718),
       np.float64(-212.93616495016613),
       np.float64(-8.610125966080789))
```

2.10 Iterate Through Possible Parameters

```
[19]: # See .py file for backtest iterations
```

2.11 Read Existing Results CSV, Fill NaN

```
[20]: INITIAL_CAPITAL = 10_000
       START_DATE = "2024-01-01"
       END_DATE = "2024-12-31"
```

```
[21]: # Read CSV
       combined_results_df = pd.
         ↳ read_csv(f"multi_asset_strategy_results_{START_DATE}_{END_DATE}.csv")
       # combined_results_df = pd.
         ↳ read_csv(f"multi_asset_strategy_results_{START_DATE}_{END_DATE}_{ORDER_ENTRY}.
         ↳ csv")

       # Fillna for any MA_DAYS that are NaN (when no MA filter is used)
       combined_results_df["MA_DAYS"] = combined_results_df["MA_DAYS"].fillna(0.0)

       # Deduplicate rows
       combined_results_df = combined_results_df.drop_duplicates(
```



```

subset=[
    "TICKERS",
    "MA_DAYS",
    "INITIAL_CAPITAL",
    "RSI_PERIOD",
    "RSI_THRESHOLD",
    "TRAILING_STOP_PCT",
    "Order Entry",
    "BB Rule",
    "BB Window",
    "BB Num Std",
    "Trade Taker Fee",
    "Trade Maker Fee",
]
)

# Export edited CSV
# combined_results_df.
↳ to_csv(f"multi_asset_strategy_results_{START_DATE}_{END_DATE}_{ORDER_ENTRY}".
↳ csv", index=False)
combined_results_df

```

```

[21]:
    TICKERS  MA_DAYS  INITIAL_CAPITAL  RSI_PERIOD  RSI_THRESHOLD  \
0    BTC-USD    0.00          10000         6             12
1    BTC-USD    0.00          10000         6             12
2    BTC-USD    0.00          10000         6             12
3    BTC-USD    0.00          10000         6             12
4    BTC-USD    0.00          10000         6             12
...
15995  BTC-USD    49.00          10000        24             30
15996  BTC-USD    49.00          10000        24             30
15997  BTC-USD    49.00          10000        24             30
15998  BTC-USD    49.00          10000        24             30
15999  BTC-USD    49.00          10000        24             30

    TRAILING_STOP_PCT  START_DATE  END_DATE  Total Trades  Win Rate  ...  \
0                0.01  2024-01-01  2024-12-31      807.00    0.27  ...
1                0.01  2024-01-01  2024-12-31      766.00    0.26  ...
2                0.01  2024-01-01  2024-12-31      970.00    0.30  ...
3                0.01  2024-01-01  2024-12-31      966.00    0.29  ...
4                0.01  2024-01-01  2024-12-31      718.00    0.29  ...
...
15995            0.03  2024-01-01  2024-12-31      207.00    0.30  ...
15996            0.03  2024-01-01  2024-12-31      140.00    0.31  ...
15997            0.03  2024-01-01  2024-12-31      152.00    0.30  ...
15998            0.03  2024-01-01  2024-12-31      139.00    0.32  ...
15999            0.03  2024-01-01  2024-12-31      150.00    0.31  ...

```

	Runtime	EMA (s)	Success-only	Runtime	EMA (s)	Success	Error	\
0		19.43			19.43	True	NaN	
1		18.55			18.55	True	NaN	
2		17.42			17.42	True	NaN	
3		16.45			16.45	True	NaN	
4		14.90			14.90	True	NaN	
...			
15995		3.81			3.81	True	NaN	
15996		3.67			3.67	True	NaN	
15997		3.59			3.59	True	NaN	
15998		3.48			3.48	True	NaN	
15999		3.39			3.39	True	NaN	

	Order	Entry	BB Rule	BB Window	BB Num	Std	Trade	Taker	Fee	\
0		market	touch_lower	20.00		2.00			0.00	
1		market	NaN	NaN		NaN			0.00	
2		limit	touch_lower	20.00		2.00			0.00	
3		limit	NaN	NaN		NaN			0.00	
4		market	touch_lower	20.00		2.00			0.00	
...					
15995		limit	NaN	NaN		NaN			0.00	
15996		market	touch_lower	20.00		2.00			0.00	
15997		market	NaN	NaN		NaN			0.00	
15998		limit	touch_lower	20.00		2.00			0.00	
15999		limit	NaN	NaN		NaN			0.00	

	Trade	Maker	Fee
0			0.00
1			0.00
2			0.00
3			0.00
4			0.00
...			...
15995			0.00
15996			0.00
15997			0.00
15998			0.00
15999			0.00

[16000 rows x 45 columns]

```
[22]: # Re-read CSV
combined_results_df = pd.
      ↪read_csv(f"multi_asset_strategy_results_{START_DATE}_{END_DATE}.csv")
```

```
[23]: # Sort values by MAR Ratio descending
combined_results_df.sort_values(by="MAR Ratio", ascending=False, inplace=True)
```

```
[24]: combined_results_df.columns
```

```
[24]: Index(['TICKERS', 'MA_DAYS', 'INITIAL_CAPITAL', 'RSI_PERIOD', 'RSI_THRESHOLD',
          'TRAILING_STOP_PCT', 'START_DATE', 'END_DATE', 'Total Trades',
          'Win Rate', 'Total Return', 'Average Return Per Trade',
          'Max Trade Gain (%)', 'Max Trade Loss (%)', 'Total PnL',
          'Average PnL Per Trade', 'Max Trade Gain ($)', 'Max Trade Loss ($)',
          'Annualized Mean Return', 'Annualized Volatility',
          'Annualized Sharpe Ratio', 'CAGR', 'Daily Max Return',
          'Daily Max Return Date', 'Daily Min Return', 'Daily Min Return Date',
          'Max Drawdown', 'Peak', 'Trough', 'Recovery Date', 'Days to Recover',
          'MAR Ratio', 'Total Runtime (s)', 'Average Runtime (s)', 'Runtime (s)',
          'Runtime EMA (s)', 'Success-only Runtime EMA (s)', 'Success', 'Error',
          'Order Entry', 'BB Rule', 'BB Window', 'BB Num Std', 'Trade Taker Fee',
          'Trade Maker Fee'],
          dtype='object')
```

```
[25]: combined_results_df[['TICKERS', 'MA_DAYS', 'RSI_PERIOD', 'RSI_THRESHOLD',
          'TRAILING_STOP_PCT', 'Total Trades',
          'Win Rate', 'Total Return', 'Average Return Per Trade',
          'Max Trade Gain (%)', 'Max Trade Loss (%)', 'Total PnL',
          'Average PnL Per Trade', 'Max Trade Gain ($)', 'Max Trade Loss ($)',
          'Max Drawdown', 'Annualized Mean Return', 'Annualized Volatility',
          'Annualized Sharpe Ratio', 'CAGR',
          'MAR Ratio', 'BB Rule', 'Trade Taker Fee', 'Trade Maker Fee']]
```

```
[25]:
```

	TICKERS	MA_DAYS	RSI_PERIOD	RSI_THRESHOLD	TRAILING_STOP_PCT	\
3862	BTC-USD	7.00	24	18	0.01	
3863	BTC-USD	7.00	24	18	0.01	
3860	BTC-USD	7.00	24	18	0.01	
3861	BTC-USD	7.00	24	18	0.01	
3442	BTC-USD	7.00	20	16	0.01	
...	
3815	BTC-USD	7.00	24	12	0.03	
3816	BTC-USD	7.00	24	12	0.03	
3817	BTC-USD	7.00	24	12	0.03	
3818	BTC-USD	7.00	24	12	0.03	
3819	BTC-USD	7.00	24	12	0.03	
	Total Trades	Win Rate	Total Return	Average Return Per Trade	\	
3862	12.00	0.67	0.12	0.01		
3863	12.00	0.67	0.12	0.01		
3860	12.00	0.67	0.11	0.01		
3861	12.00	0.67	0.11	0.01		

3442	14.00	0.57	0.10	0.01
...
3815	NaN	NaN	NaN	NaN
3816	NaN	NaN	NaN	NaN
3817	NaN	NaN	NaN	NaN
3818	NaN	NaN	NaN	NaN
3819	NaN	NaN	NaN	NaN

	Max Trade Gain (%)	...	Max Trade Loss (\$)	Max Drawdown	\
3862	0.05	...	-83.78	-0.02	
3863	0.05	...	-83.78	-0.02	
3860	0.05	...	-93.48	-0.02	
3861	0.05	...	-93.48	-0.02	
3442	0.05	...	-119.05	-0.02	
...	
3815	NaN	...	NaN	NaN	
3816	NaN	...	NaN	NaN	
3817	NaN	...	NaN	NaN	
3818	NaN	...	NaN	NaN	
3819	NaN	...	NaN	NaN	

	Annualized Mean Return	Annualized Volatility	Annualized Sharpe Ratio	\
3862	0.12	0.07	1.78	
3863	0.12	0.07	1.78	
3860	0.11	0.06	1.65	
3861	0.11	0.06	1.65	
3442	0.10	0.07	1.44	
...	
3815	NaN	NaN	NaN	
3816	NaN	NaN	NaN	
3817	NaN	NaN	NaN	
3818	NaN	NaN	NaN	
3819	NaN	NaN	NaN	

	CAGR	MAR Ratio	BB Rule	Trade Taker Fee	Trade Maker Fee
3862	0.12	8.11	touch_lower	0.00	0.00
3863	0.12	8.11	NaN	0.00	0.00
3860	0.11	6.81	touch_lower	0.00	0.00
3861	0.11	6.81	NaN	0.00	0.00
3442	0.10	6.54	touch_lower	0.00	0.00
...
3815	NaN	NaN	NaN	0.00	0.00
3816	NaN	NaN	touch_lower	0.00	0.00
3817	NaN	NaN	NaN	0.00	0.00
3818	NaN	NaN	touch_lower	0.00	0.00
3819	NaN	NaN	NaN	0.00	0.00

[16000 rows x 24 columns]

```
[26]: combined_results_df = combined_results_df[(combined_results_df["Trade Taker_Fee"] > 0.0) &
                                                (combined_results_df["Trade Maker_Fee"] > 0.0) &
                                                (combined_results_df["Success"] == True) &
                                                (combined_results_df["MAR Ratio"] > -10)]
combined_results_df
```

```
[26]:
```

	TICKERS	MA_DAYS	INITIAL_CAPITAL	RSI_PERIOD	RSI_THRESHOLD	\
3862	BTC-USD	7.00	10000	24	18	
3863	BTC-USD	7.00	10000	24	18	
3860	BTC-USD	7.00	10000	24	18	
3861	BTC-USD	7.00	10000	24	18	
3442	BTC-USD	7.00	10000	20	16	
...	
361	BTC-USD	NaN	10000	8	28	
141	BTC-USD	NaN	10000	6	26	
381	BTC-USD	NaN	10000	8	30	
161	BTC-USD	NaN	10000	6	28	
181	BTC-USD	NaN	10000	6	30	

	TRAILING_STOP_PCT	START_DATE	END_DATE	Total Trades	Win Rate	...	\
3862	0.01	2024-01-01	2024-12-31	12.00	0.67	...	
3863	0.01	2024-01-01	2024-12-31	12.00	0.67	...	
3860	0.01	2024-01-01	2024-12-31	12.00	0.67	...	
3861	0.01	2024-01-01	2024-12-31	12.00	0.67	...	
3442	0.01	2024-01-01	2024-12-31	14.00	0.57	...	
...	
361	0.01	2024-01-01	2024-12-31	769.00	0.24	...	
141	0.01	2024-01-01	2024-12-31	776.00	0.24	...	
381	0.01	2024-01-01	2024-12-31	768.00	0.24	...	
161	0.01	2024-01-01	2024-12-31	769.00	0.24	...	
181	0.01	2024-01-01	2024-12-31	769.00	0.24	...	

	Runtime EMA (s)	Success-only	Runtime EMA (s)	Success	Error	\
3862	2.39		2.39	True	NaN	
3863	2.39		2.39	True	NaN	
3860	2.32		2.33	True	NaN	
3861	2.31		2.31	True	NaN	
3442	2.51		2.51	True	NaN	
...		
361	15.89		15.89	True	NaN	
141	14.09		14.09	True	NaN	

381	18.36	18.36	True	NaN
161	16.43	16.43	True	NaN
181	20.65	20.65	True	NaN

	Order Entry	BB Rule	BB Window	BB Num Std	Trade Taker Fee \
3862	limit	touch_lower	20.00	2.00	0.00
3863	limit	NaN	NaN	NaN	0.00
3860	market	touch_lower	20.00	2.00	0.00
3861	market	NaN	NaN	NaN	0.00
3442	limit	touch_lower	20.00	2.00	0.00
...
361	market	NaN	NaN	NaN	0.00
141	market	NaN	NaN	NaN	0.00
381	market	NaN	NaN	NaN	0.00
161	market	NaN	NaN	NaN	0.00
181	market	NaN	NaN	NaN	0.00

	Trade Maker Fee
3862	0.00
3863	0.00
3860	0.00
3861	0.00
3442	0.00
...	...
361	0.00
141	0.00
381	0.00
161	0.00
181	0.00

[15928 rows x 45 columns]

```
[27]: def plot_iteration_results(
        results_df: pd.DataFrame,
    ) -> None:

        combined_results_df = results_df.copy()

        # Create histogram of results for MAR Ratio
        plt.figure(figsize=(12, 6))
        plt.hist(combined_results_df['MAR Ratio'], bins=200, density=True, alpha=0.
        ↪6, color='g')
        plt.title('Distribution of MAR Ratio')
        plt.xlabel('MAR Ratio')
        plt.ylabel('Frequency')
        plt.grid()
        plt.show()
```

```

# Create histogram of results for MA Days
plt.figure(figsize=(12, 6))
plt.hist(combined_results_df['MA_DAYS'], bins=200, density=True, alpha=0.6,
↪color='g')
plt.title('Distribution of MA Days')
plt.xlabel('MA Days')
plt.ylabel('Frequency')
plt.grid()
plt.show()

# Create histogram of results for RSI period
plt.figure(figsize=(12, 6))
plt.hist(combined_results_df['RSI_PERIOD'], bins=200, density=True, alpha=0.
↪6, color='g')
plt.title('Distribution of RSI Period')
plt.xlabel('RSI Period')
plt.ylabel('Frequency')
plt.grid()
plt.show()

# Create histogram of results for RSI threshold
plt.figure(figsize=(12, 6))
plt.hist(combined_results_df['RSI_THRESHOLD'], bins=200, density=True,
↪alpha=0.6, color='g')
plt.title('Distribution of RSI Threshold')
plt.xlabel('RSI Threshold')
plt.ylabel('Frequency')
plt.grid()
plt.show()

# Create histogram of results for trailing stop percentage
plt.figure(figsize=(12, 6))
plt.hist(combined_results_df['TRAILING_STOP_PCT'], bins=200, density=True,
↪alpha=0.6, color='g')
plt.title('Distribution of Trailing Stop Percentage')
plt.xlabel('Trailing Stop Percentage')
plt.ylabel('Frequency')
plt.grid()
plt.show()

# Plot MAR Ratio vs MA_DAYS
plt.figure(figsize=(12, 6))
plt.scatter(combined_results_df['MA_DAYS'], combined_results_df['MAR_
↪Ratio'], alpha=0.6)
plt.title('MAR Ratio vs. MA_DAYS')
plt.xlabel('MA_DAYS')

```

```

plt.ylabel('MAR Ratio')
plt.grid()
plt.show()

# Plot MAR Ratio vs RSI_PERIOD
plt.figure(figsize=(12, 6))
plt.scatter(combined_results_df['RSI_PERIOD'], combined_results_df['MAR_
↳Ratio'], alpha=0.6)
plt.title('MAR Ratio vs. RSI_PERIOD')
plt.xlabel('RSI_PERIOD')
plt.ylabel('MAR Ratio')
plt.grid()
plt.show()

# Plot MAR ratio vs RSI_THRESHOLD
plt.figure(figsize=(12, 6))
plt.scatter(combined_results_df['RSI_THRESHOLD'], combined_results_df['MAR_
↳Ratio'], alpha=0.6)
plt.title('MAR Ratio vs. RSI_THRESHOLD')
plt.xlabel('RSI_THRESHOLD')
plt.ylabel('MAR Ratio')
plt.grid()
plt.show()

# Plot MAR Ratio vs TRAILING_STOP
plt.figure(figsize=(12, 6))
plt.scatter(combined_results_df['TRAILING_STOP_PCT'],
↳combined_results_df['MAR Ratio'], alpha=0.6)
plt.title('MAR Ratio vs. TRAILING_STOP')
plt.xlabel('TRAILING_STOP')
plt.ylabel('MAR Ratio')
plt.grid()
plt.show()

# Plot MAR Ratio vs Total Return
plt.figure(figsize=(12, 6))
plt.scatter(combined_results_df['Total Return'], combined_results_df['MAR_
↳Ratio'], alpha=0.6)
plt.title('MAR Ratio vs. Total Return')
plt.xlabel('Total Return')
plt.ylabel('MAR Ratio')
plt.grid()
plt.show()

# Plot MAR Ratio vs Win Rate
plt.figure(figsize=(12, 6))

```



```

plt.scatter(combined_results_df['Win Rate'], combined_results_df['MAR_
↳Ratio'], alpha=0.6)
plt.title('MAR Ratio vs. Win Rate')
plt.xlabel('Win Rate')
plt.ylabel('MAR Ratio')
plt.grid()
plt.show()

# Plot MAR Ratio vs Max Drawdown
plt.figure(figsize=(12, 6))
plt.scatter(combined_results_df['Max Drawdown'], combined_results_df['MAR_
↳Ratio'], alpha=0.6)
plt.title('MAR Ratio vs. Max Drawdown')
plt.xlabel('Max Drawdown')
plt.ylabel('MAR Ratio')
plt.grid()
plt.show()

# Plot MAR Ratio vs Total Trades
plt.figure(figsize=(12, 6))
plt.scatter(combined_results_df['Total Trades'], combined_results_df['MAR_
↳Ratio'], alpha=0.6)
plt.title('MAR Ratio vs. Total Trades')
plt.xlabel('Total Trades')
plt.ylabel('MAR Ratio')
plt.grid()
plt.show()

# Plot MAR Ratio vs CAGR
plt.figure(figsize=(12, 6))
plt.scatter(combined_results_df['CAGR'], combined_results_df['MAR Ratio'],
↳alpha=0.6)
plt.title('MAR Ratio vs. CAGR')
plt.xlabel('CAGR')
plt.ylabel('MAR Ratio')
plt.grid()
plt.show()

# Plot MAR Ratio vs Annualized Mean Return
plt.figure(figsize=(12, 6))
plt.scatter(combined_results_df['Annualized Mean Return'],
↳combined_results_df['MAR Ratio'], alpha=0.6)
plt.title('MAR Ratio vs. Annualized Mean Return')
plt.xlabel('Annualized Mean Return')
plt.ylabel('MAR Ratio')
plt.grid()
plt.show()

```

```

# Plot MAR Ratio vs Annualized Volatility
plt.figure(figsize=(12, 6))
plt.scatter(combined_results_df['Annualized Volatility'],
combined_results_df['MAR Ratio'], alpha=0.6)
plt.title('MAR Ratio vs. Annualized Volatility')
plt.xlabel('Annualized Volatility')
plt.ylabel('MAR Ratio')
plt.grid()
plt.show()

# Plot MAR Ratio vs Annualized Sharpe Ratio
plt.figure(figsize=(12, 6))
plt.scatter(combined_results_df['Annualized Sharpe Ratio'],
combined_results_df['MAR Ratio'], alpha=0.6)
plt.title('MAR Ratio vs. Annualized Sharpe Ratio')
plt.xlabel('Annualized Sharpe Ratio')
plt.ylabel('MAR Ratio')
plt.grid()
plt.show()

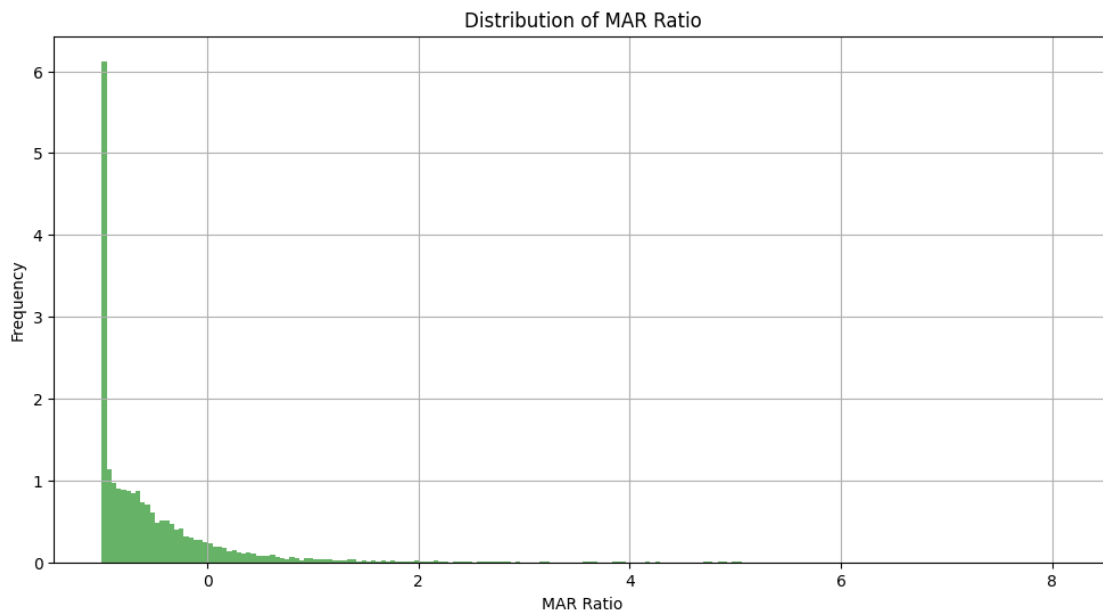
return None

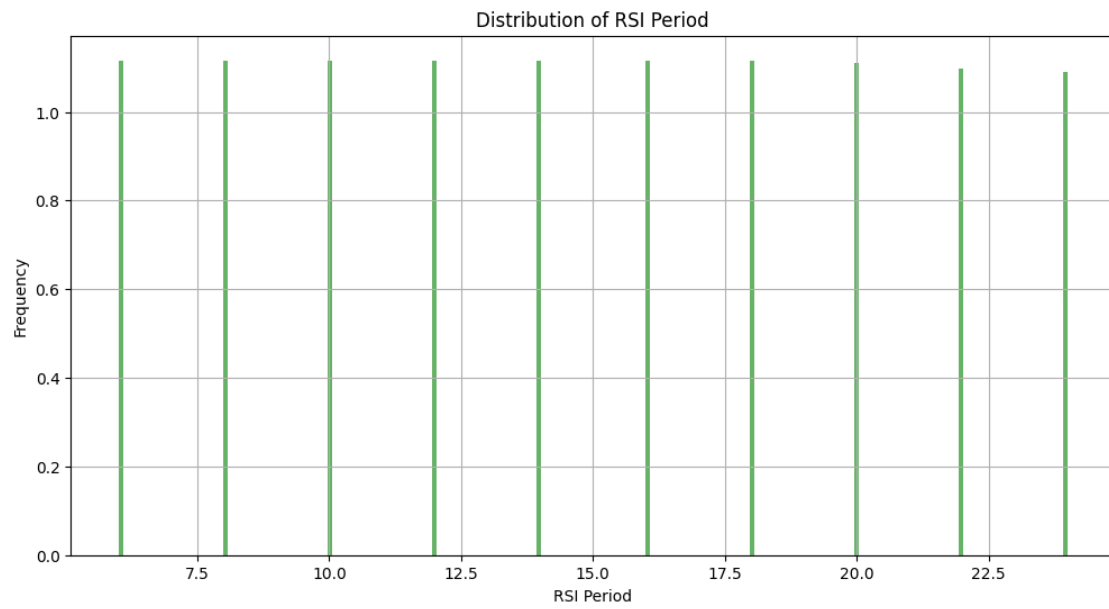
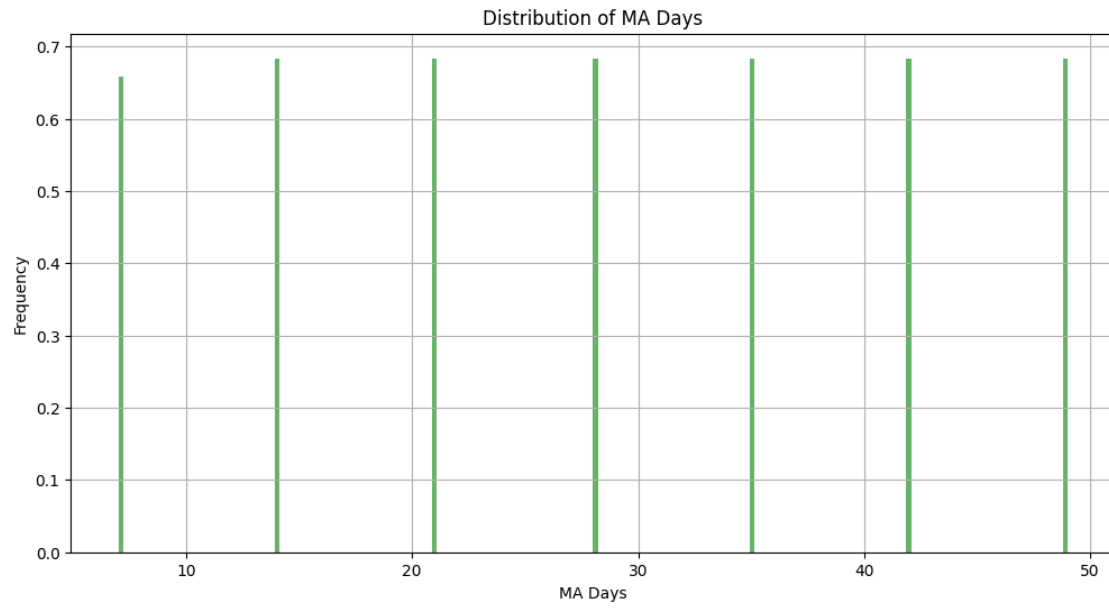
```

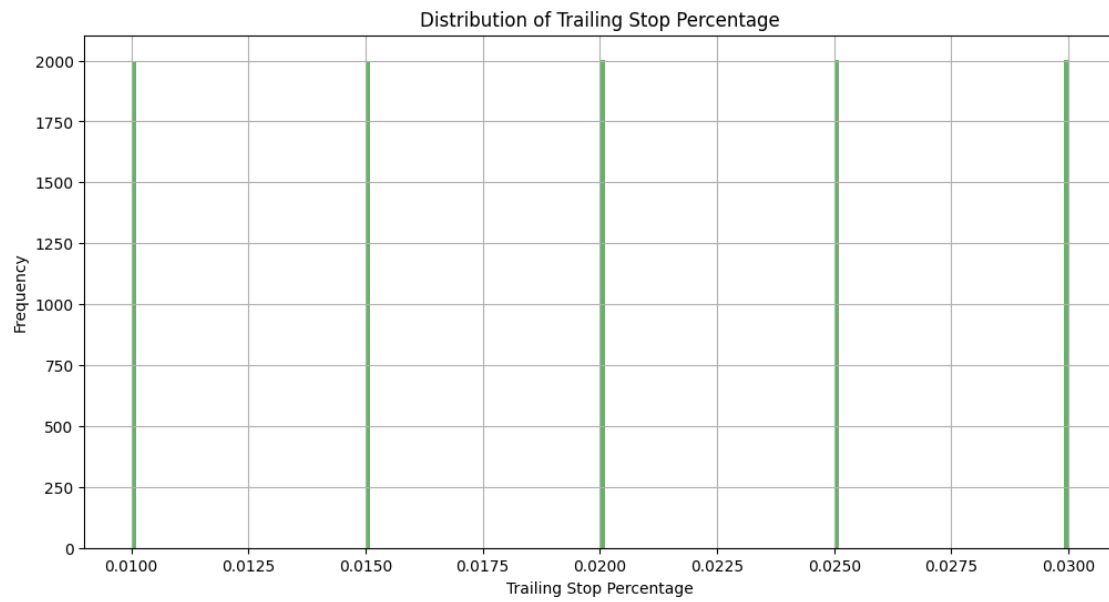
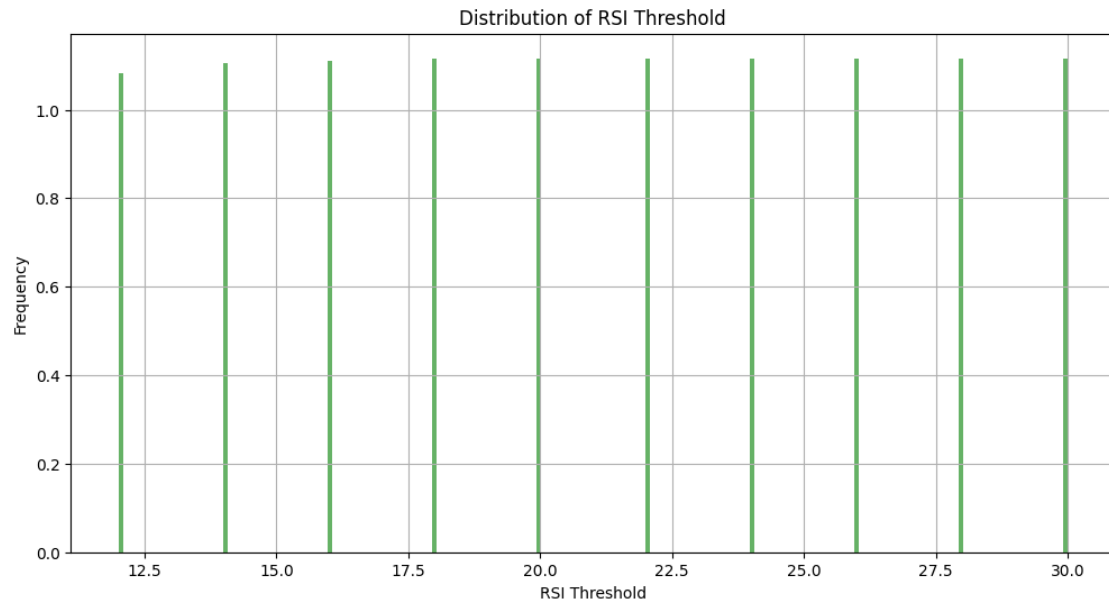
```

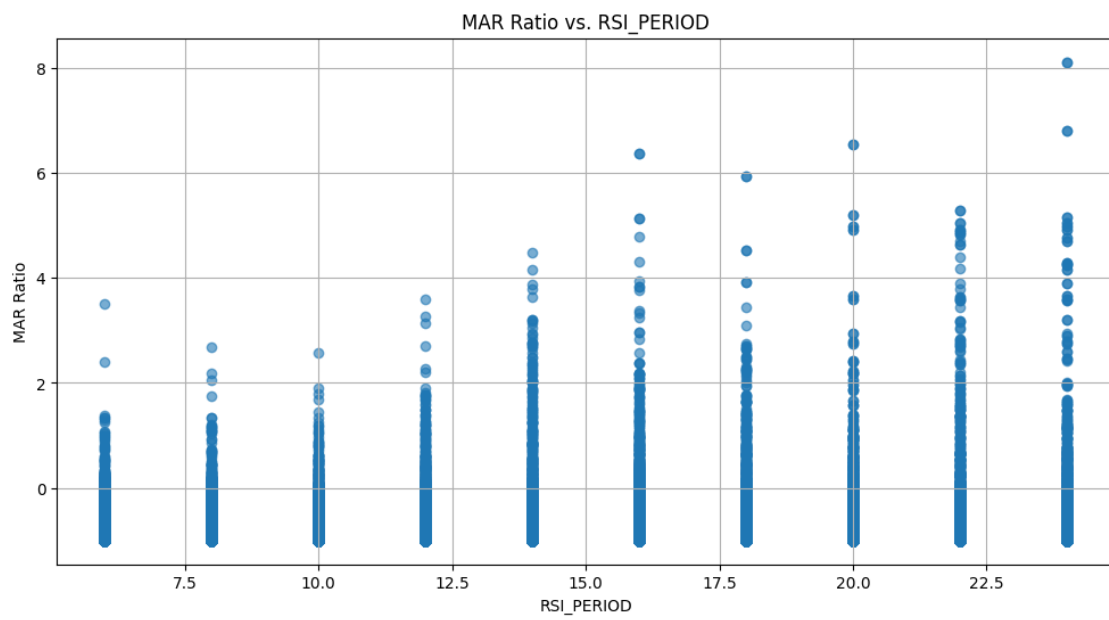
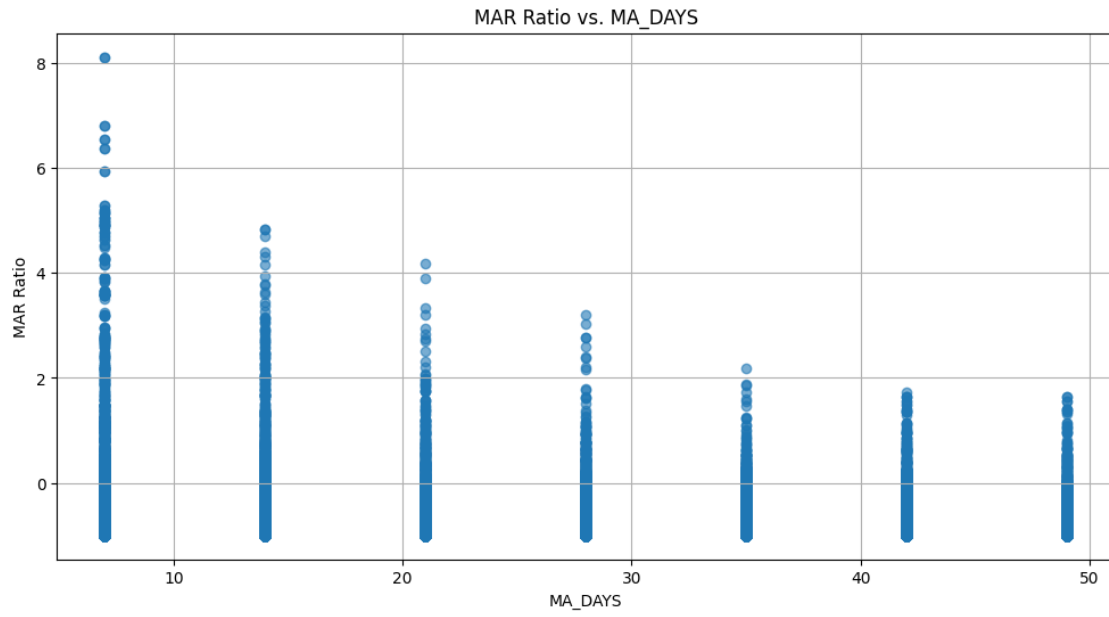
[28]: plot_iteration_results(
      results_df=combined_results_df,
      )

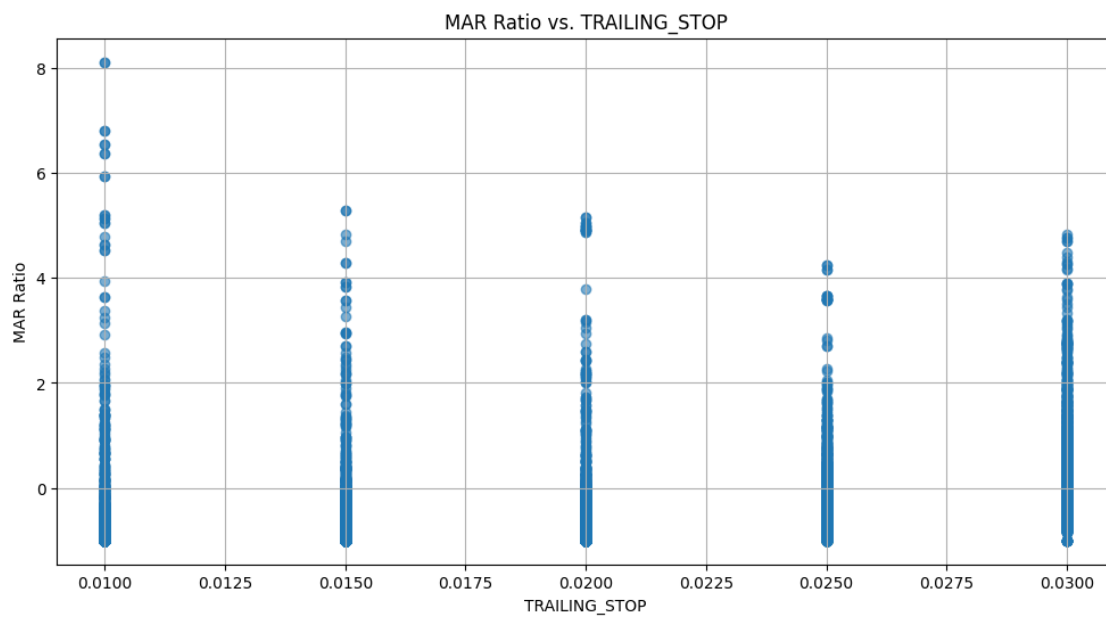
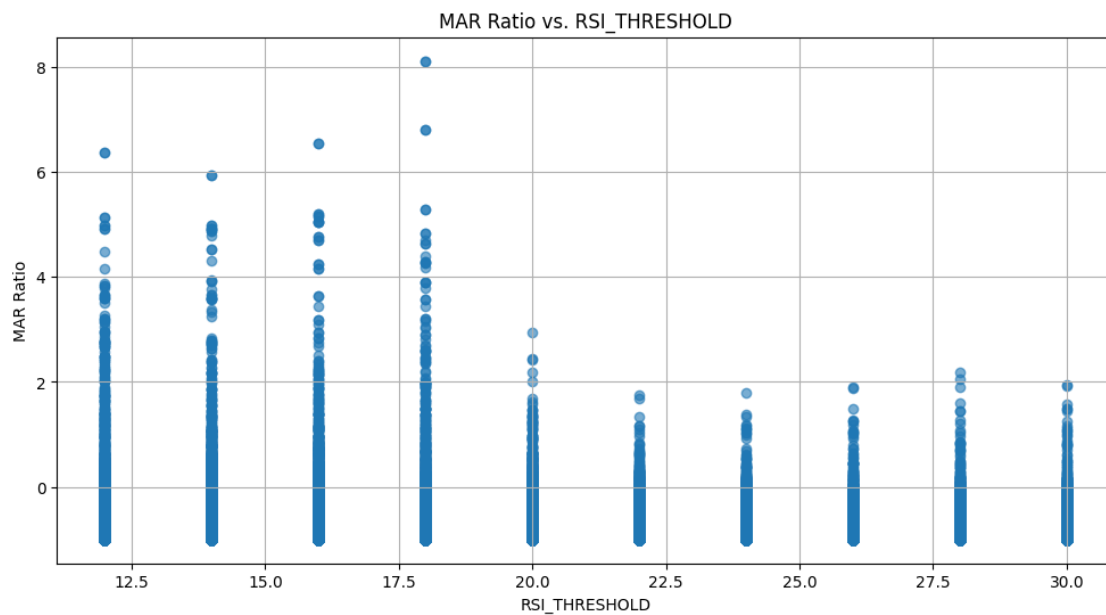
```

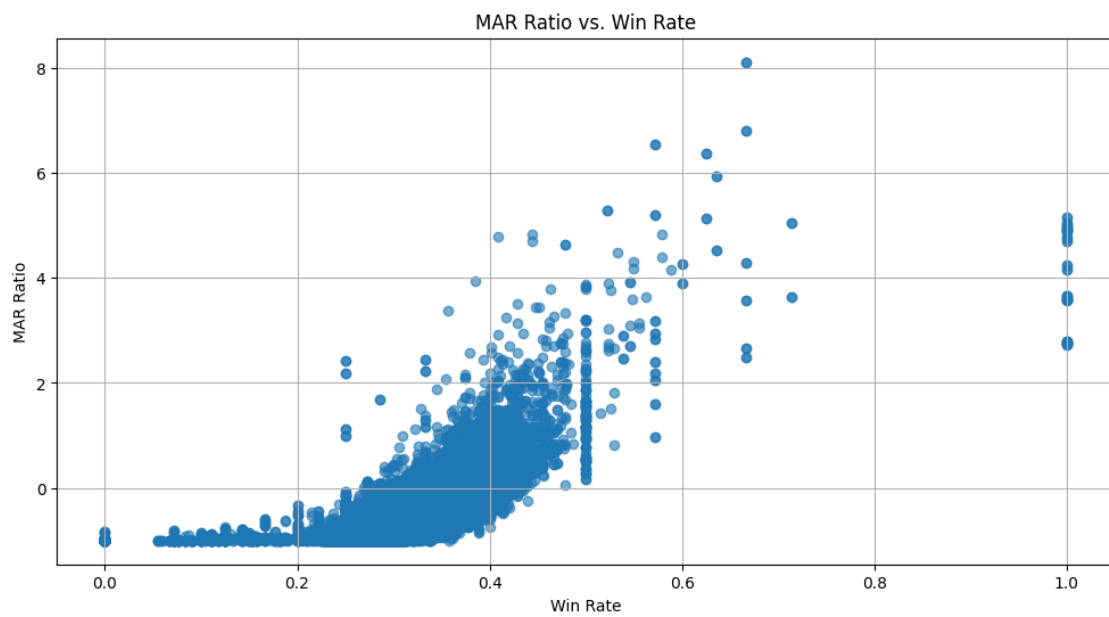
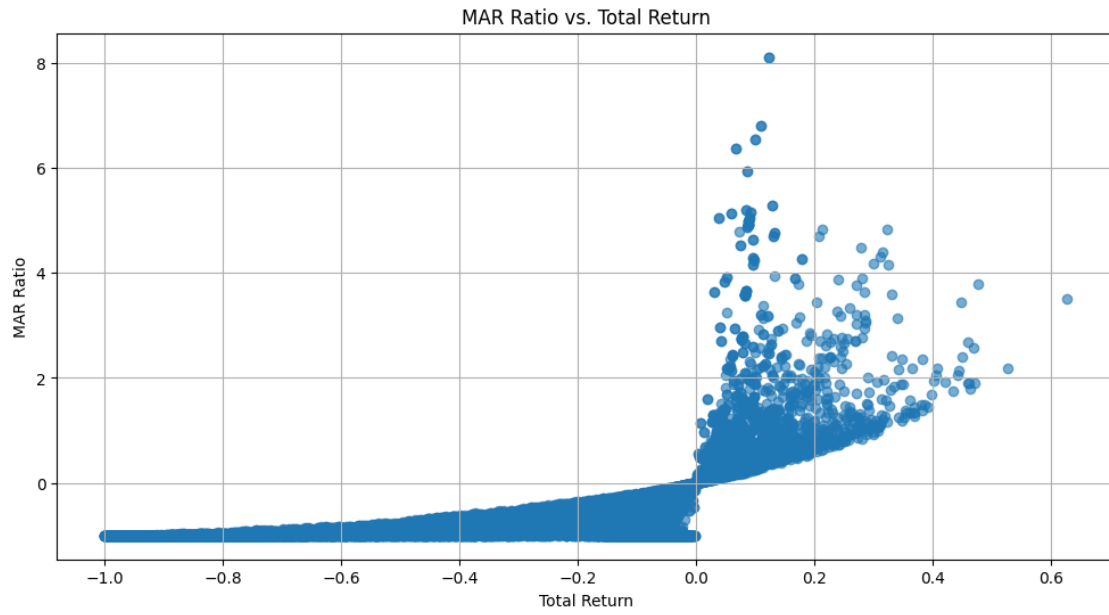


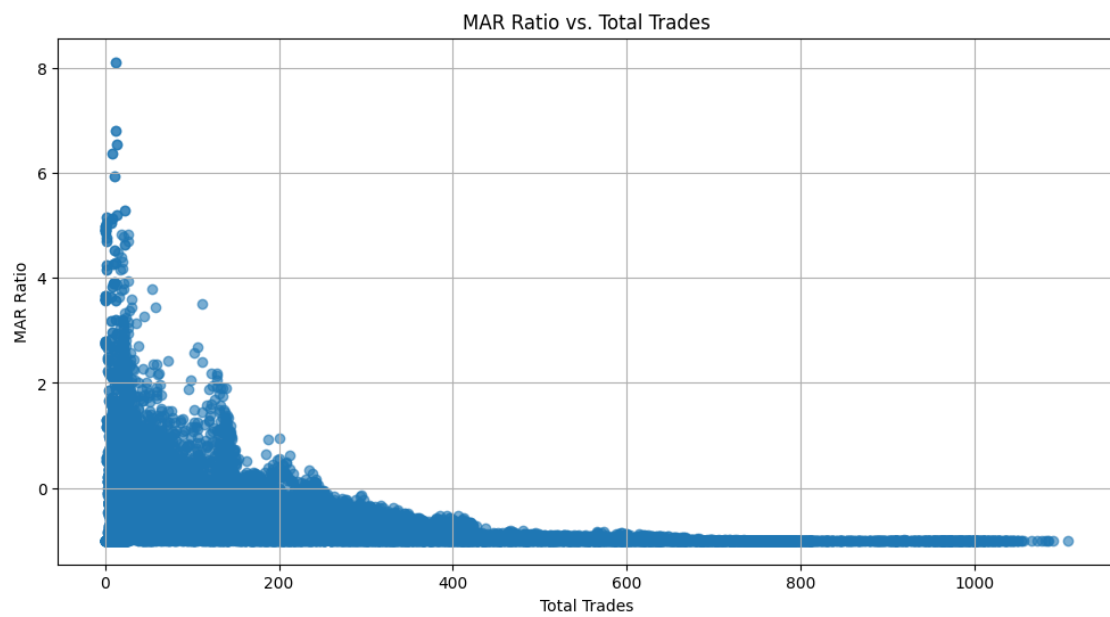
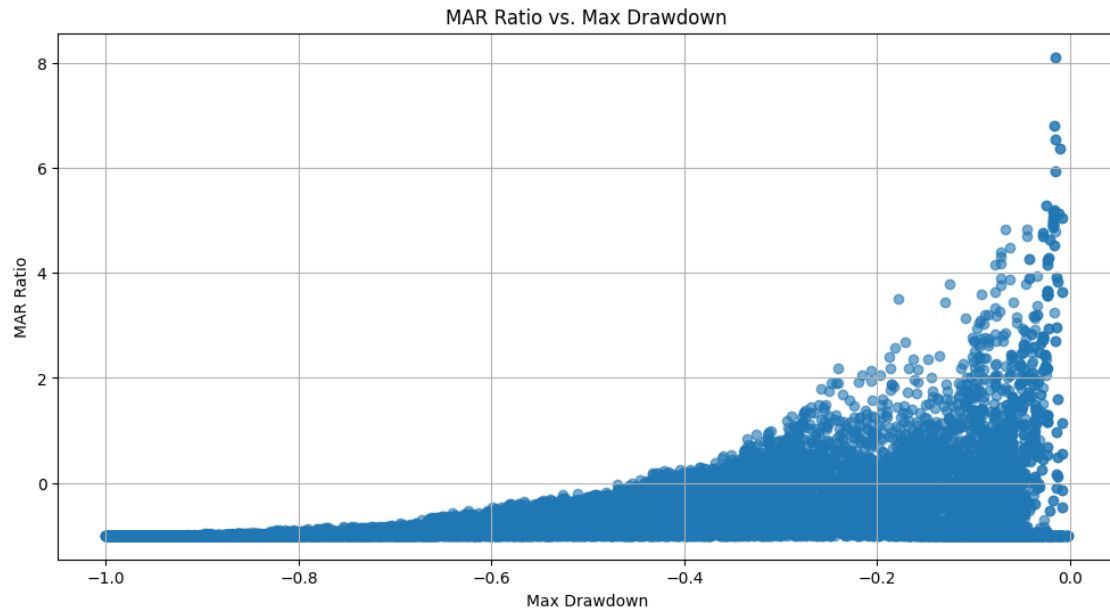


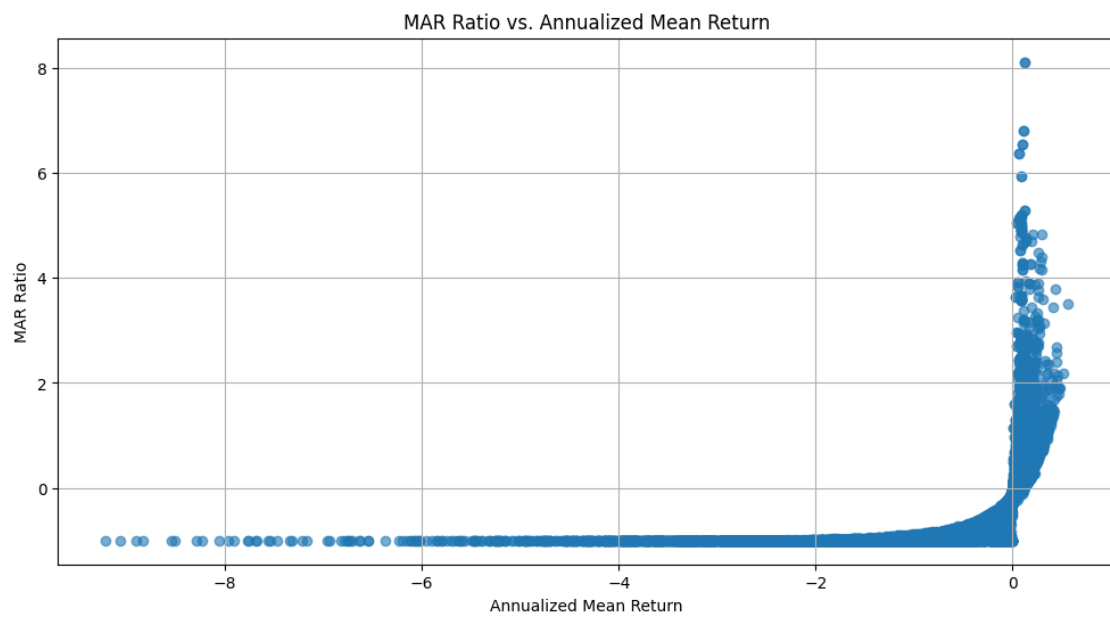
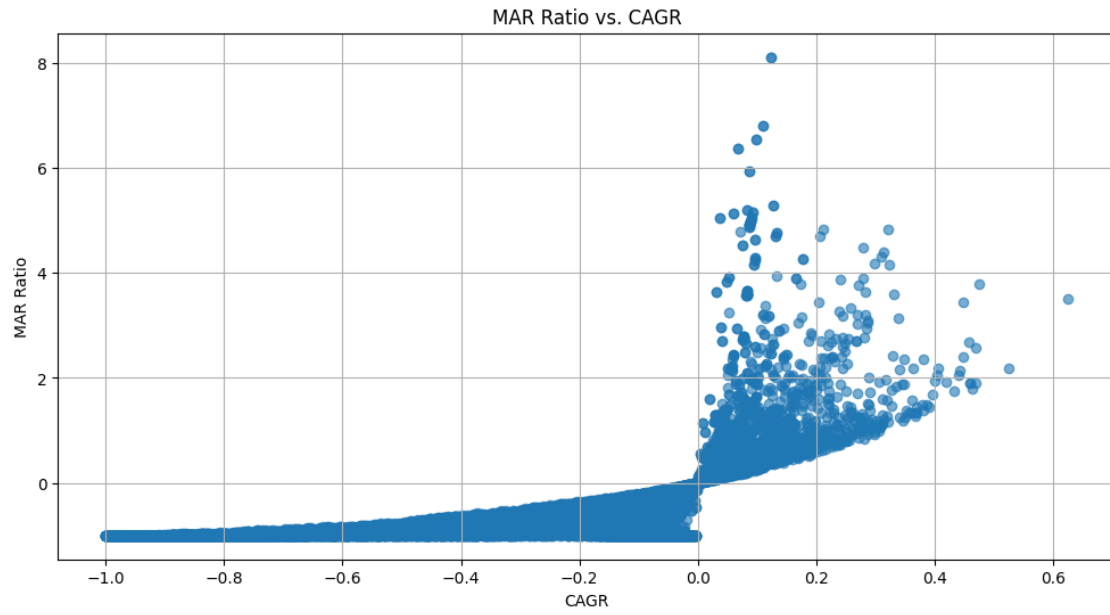


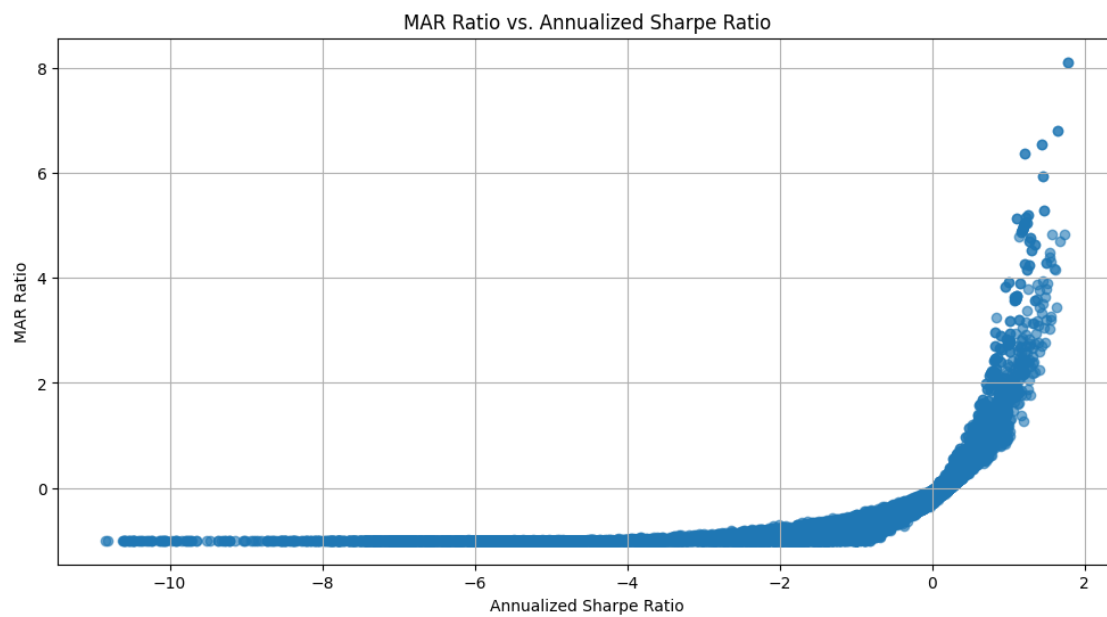
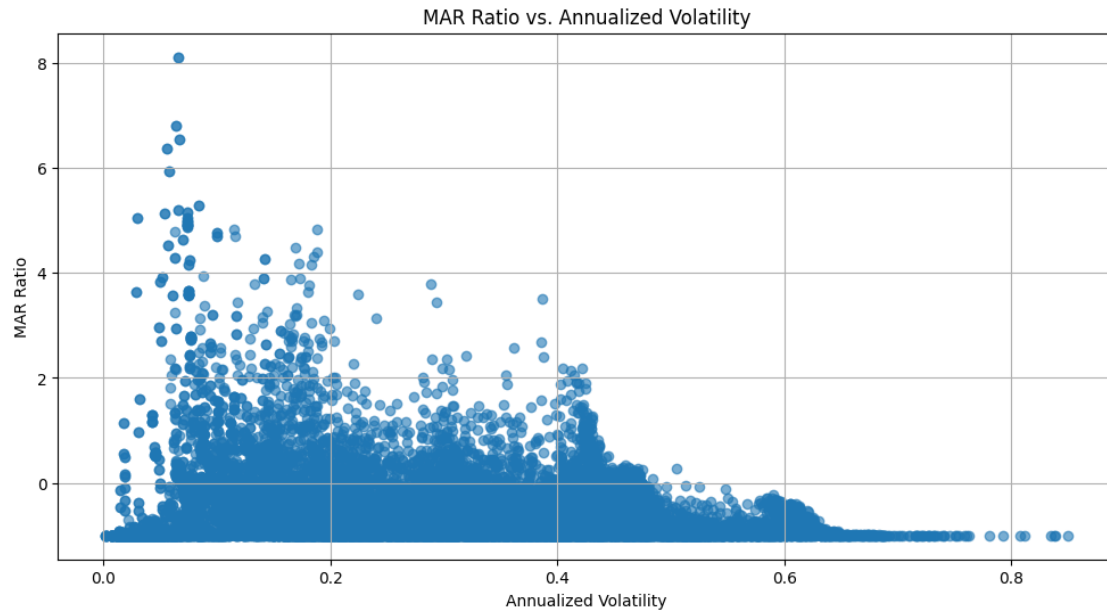












```
[29]: import math
import pandas as pd
from typing import Iterable, Mapping, Optional, Tuple

def filter_top_percent(
    df: pd.DataFrame,
```

```

cols: Iterable[str],
top_pct: float,
how: str = "all", # "all" = rows must be in top pct for every column;
↳ "any" = for at least one
ascending: Optional[Mapping[str, bool]] = None, # True = smaller is better;
↳ False = larger is better
coerce_numeric: bool = True, # coerce non-numeric to NaN for the listed
↳ columns
) -> Tuple[pd.DataFrame, pd.Series]:
    """
    Filter rows to the top X% for a set of columns.

    Parameters
    -----
    df : pd.DataFrame
        Source data.
    cols : Iterable[str]
        Columns to evaluate.
    top_pct : float
        Percentage (e.g., 10 or 0.10 for top 10%).
    how : {"all", "any"}, default "all"
        "all": keep rows that are in the top X% for every column
↳ (intersection).
        "any": keep rows that are in the top X% for at least one column
↳ (union).
    ascending : dict[str, bool] | None
        Per-column direction. For each col: True => smaller is better (use
↳ lower tail),
        False => larger is better (use upper tail). If None, defaults to False
↳ for all.
        Example: {"Max Drawdown": True, "MAR Ratio": False}
    coerce_numeric : bool
        If True, tries to convert the specified cols to numeric.

    Returns
    -----
    filtered_df : pd.DataFrame
        Rows meeting the criterion.
    thresholds : pd.Series
        Threshold value used for each column (the cutoff included in the keep).
    """
    if 0 < top_pct <= 1:
        frac = top_pct
    elif 1 < top_pct <= 100:
        frac = top_pct / 100.0
    else:

```

```

        raise ValueError("top_pct must be in (0,1] or (1,100].")

cols = list(cols)
if coerce_numeric:
    for c in cols:
        if c in df.columns:
            df[c] = pd.to_numeric(df[c], errors="coerce")

if ascending is None:
    ascending = {c: False for c in cols} # larger is better by default
else:
    # Any unspecified column defaults to larger-is-better
    ascending = {**{c: False for c in cols}, **ascending}

n = len(df)
# Use quantile cutoff; include ties on the boundary
thresholds = {}
masks = {}

for c in cols:
    if c not in df.columns:
        raise KeyError(f"Column '{c}' not found in DataFrame.")

    # If smaller is better, top X% = bottom X% (lower tail). Else upper
    ↪tail.
    q = frac if ascending[c] else (1 - frac)
    # Guard against all-NaN columns
    if df[c].notna().any():
        cutoff = df[c].quantile(q)
    else:
        cutoff = float("nan")
    thresholds[c] = cutoff

    if ascending[c]:
        # keep values <= cutoff (lower tail)
        masks[c] = df[c] <= cutoff
    else:
        # keep values >= cutoff (upper tail)
        masks[c] = df[c] >= cutoff

thresholds = pd.Series(thresholds)

if how == "all":
    keep_mask = pd.concat(masks, axis=1).all(axis=1)
elif how == "any":
    keep_mask = pd.concat(masks, axis=1).any(axis=1)
else:

```

```

        raise ValueError("how must be 'all' or 'any'.")

    filtered_df = df[keep_mask].copy()
    return filtered_df, thresholds

```

```

[30]: top25_mix, thresh_mix = filter_top_percent(
        df=combined_results_df,
        cols=["MAR Ratio"],
        top_pct=25,
        how="all",
        ascending={} # Set True for lowest value
    )

```

```

[31]: top10_mar, thresh_mix = filter_top_percent(
        df=combined_results_df,
        cols=["MAR Ratio", "CAGR", "Total Return"],
        top_pct=10,
        how="all",
        ascending={} # Set True for lowest value
    )

```

```

[32]: top10_mar = top10_mar.sort_values(by=["Total Return"], ascending=False)
    top10_mar

```

```

[32]:
    TICKERS  MA_DAYS  INITIAL_CAPITAL  RSI_PERIOD  RSI_THRESHOLD  \
2018  BTC-USD      7.00             10000          6             12
2298  BTC-USD      7.00             10000          8             20
1678  BTC-USD      NaN             10000         22             18
2778  BTC-USD      7.00             10000         12             28
2478  BTC-USD      7.00             10000         10             18
...      ...      ...      ...      ...      ...
3432  BTC-USD      7.00             10000         20             14
3433  BTC-USD      7.00             10000         20             14
7222  BTC-USD     21.00             10000         18             14
12614  BTC-USD     42.00             10000         12             12
2807  BTC-USD      7.00             10000         14             12

    TRAILING_STOP_PCT  START_DATE  END_DATE  Total Trades  Win Rate  ...  \
2018                0.03  2024-01-01  2024-12-31      112.00    0.43  ...
2298                0.03  2024-01-01  2024-12-31      129.00    0.43  ...
1678                0.03  2024-01-01  2024-12-31       54.00    0.46  ...
2778                0.03  2024-01-01  2024-12-31      132.00    0.43  ...
2478                0.03  2024-01-01  2024-12-31      102.00    0.40  ...
...      ...      ...      ...      ...      ...
3432                0.03  2024-01-01  2024-12-31        4.00    0.25  ...
3433                0.03  2024-01-01  2024-12-31        4.00    0.25  ...
7222                0.01  2024-01-01  2024-12-31       17.00    0.41  ...

```

12614	0.03	2024-01-01	2024-12-31	45.00	0.33	...
2807	0.01	2024-01-01	2024-12-31	17.00	0.47	...

	Runtime EMA (s)	Success-only	Runtime EMA (s)	Success	Error	\
2018	3.31		3.31	True	NaN	
2298	3.65		3.65	True	NaN	
1678	2.49		2.49	True	NaN	
2778	3.76		3.76	True	NaN	
2478	3.17		3.17	True	NaN	
...		
3432	2.44		2.44	True	NaN	
3433	2.43		2.43	True	NaN	
7222	2.17		2.17	True	NaN	
12614	2.59		2.59	True	NaN	
2807	2.50		2.50	True	NaN	

	Order Entry	BB Rule	BB Window	BB Num Std	Trade Taker Fee	\
2018	limit	touch_lower	20.00	2.00	0.00	
2298	limit	touch_lower	20.00	2.00	0.00	
1678	limit	touch_lower	20.00	2.00	0.00	
2778	limit	touch_lower	20.00	2.00	0.00	
2478	limit	touch_lower	20.00	2.00	0.00	
...	
3432	market	touch_lower	20.00	2.00	0.00	
3433	market	NaN	NaN	NaN	0.00	
7222	limit	touch_lower	20.00	2.00	0.00	
12614	limit	touch_lower	20.00	2.00	0.00	
2807	limit	NaN	NaN	NaN	0.00	

	Trade Maker Fee
2018	0.00
2298	0.00
1678	0.00
2778	0.00
2478	0.00
...	...
3432	0.00
3433	0.00
7222	0.00
12614	0.00
2807	0.00

[1395 rows x 45 columns]

```
[33]: top10_mar[['TICKERS', 'MA_DAYS', 'RSI_PERIOD', 'RSI_THRESHOLD',
               'TRAILING_STOP_PCT', 'Total Trades',
               'Win Rate', 'Total Return', 'Average Return Per Trade',
```

```
'Max Trade Gain (%)', 'Max Trade Loss (%)', 'Total PnL',
'Average PnL Per Trade', 'Max Trade Gain ($)', 'Max Trade Loss ($)',
'Max Drawdown', 'Annualized Mean Return', 'Annualized Volatility',
'Annualized Sharpe Ratio', 'CAGR',
'MAR Ratio', 'BB Rule']]
```

```
[33]: TICKERS MA_DAYS RSI_PERIOD RSI_THRESHOLD TRAILING_STOP_PCT \
2018 BTC-USD 7.00 6 12 0.03
2298 BTC-USD 7.00 8 20 0.03
1678 BTC-USD NaN 22 18 0.03
2778 BTC-USD 7.00 12 28 0.03
2478 BTC-USD 7.00 10 18 0.03
... ... ... ... ...
3432 BTC-USD 7.00 20 14 0.03
3433 BTC-USD 7.00 20 14 0.03
7222 BTC-USD 21.00 18 14 0.01
12614 BTC-USD 42.00 12 12 0.03
2807 BTC-USD 7.00 14 12 0.01

Total Trades Win Rate Total Return Average Return Per Trade \
2018 112.00 0.43 0.63 0.01
2298 129.00 0.43 0.53 0.00
1678 54.00 0.46 0.48 0.01
2778 132.00 0.43 0.47 0.00
2478 102.00 0.40 0.47 0.00
... ... ... ... ...
3432 4.00 0.25 0.04 0.01
3433 4.00 0.25 0.04 0.01
7222 17.00 0.41 0.04 0.00
12614 45.00 0.33 0.04 0.00
2807 17.00 0.47 0.03 0.00

Max Trade Gain (%) ... Average PnL Per Trade Max Trade Gain ($) \
2018 0.20 ... 61.06 2560.50
2298 0.20 ... 45.30 2516.12
1678 0.14 ... 88.31 1494.60
2778 0.20 ... 39.71 2384.95
2478 0.20 ... 46.13 2441.05
... ... ... ... ...
3432 0.08 ... 88.59 794.78
3433 0.08 ... 88.59 794.78
7222 0.05 ... 20.77 453.05
12614 0.09 ... 7.79 888.31
2807 0.04 ... 20.54 414.40

Max Trade Loss ($) Max Drawdown Annualized Mean Return \
2018 -561.85 -0.18 0.56
```

2298	-564.09	-0.24	0.51
1678	-473.36	-0.13	0.43
2778	-522.34	-0.25	0.47
2478	-504.15	-0.18	0.45
...
3432	-235.02	-0.04	0.04
3433	-235.02	-0.04	0.04
7222	-130.56	-0.04	0.04
12614	-299.13	-0.15	0.05
2807	-161.43	-0.03	0.04

	Annualized Volatility	Annualized Sharpe Ratio	CAGR	MAR Ratio \
2018	0.39	1.45	0.63	3.51
2298	0.42	1.21	0.53	2.19
1678	0.29	1.49	0.48	3.79
2778	0.43	1.11	0.47	1.90
2478	0.36	1.24	0.47	2.58
...
3432	0.08	0.47	0.04	0.98
3433	0.08	0.47	0.04	0.98
7222	0.06	0.57	0.04	0.94
12614	0.19	0.27	0.03	0.23
2807	0.06	0.58	0.03	1.22

	BB Rule
2018	touch_lower
2298	touch_lower
1678	touch_lower
2778	touch_lower
2478	touch_lower
...	...
3432	touch_lower
3433	NaN
7222	touch_lower
12614	touch_lower
2807	NaN

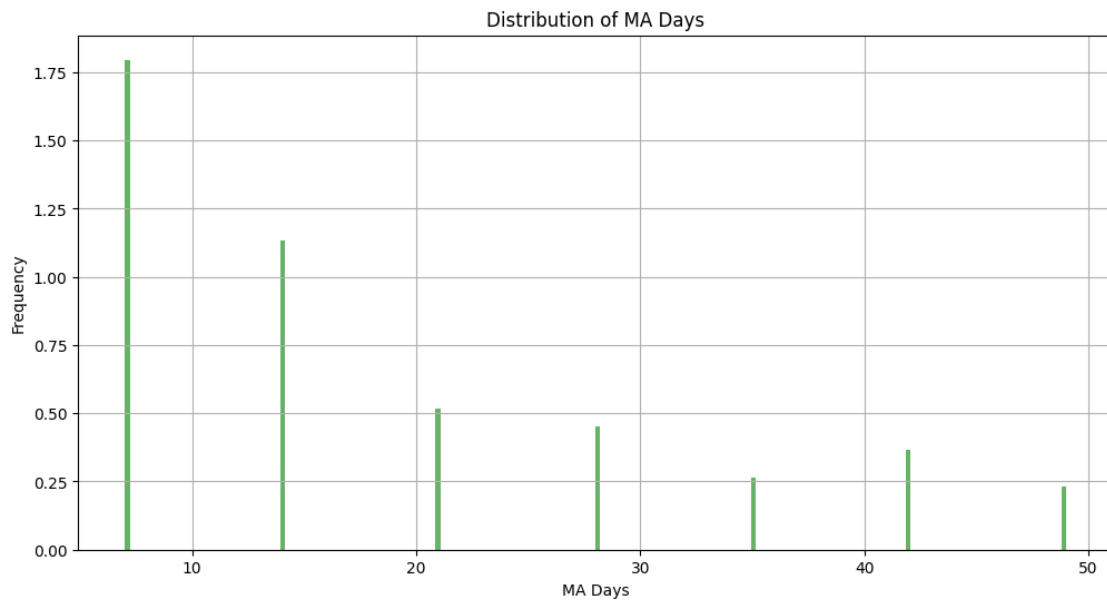
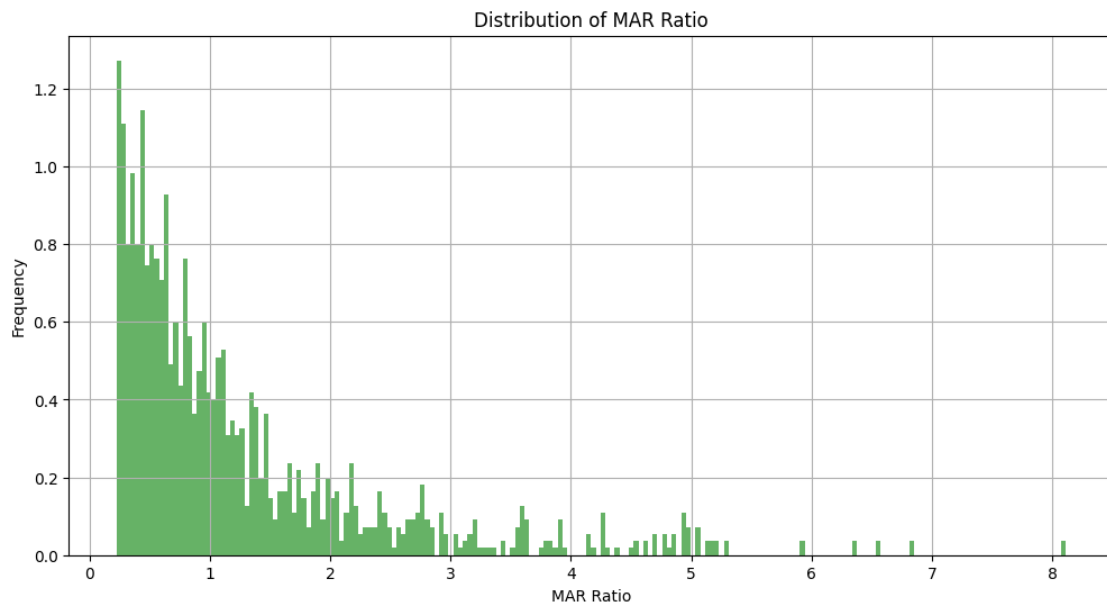
[1395 rows x 22 columns]

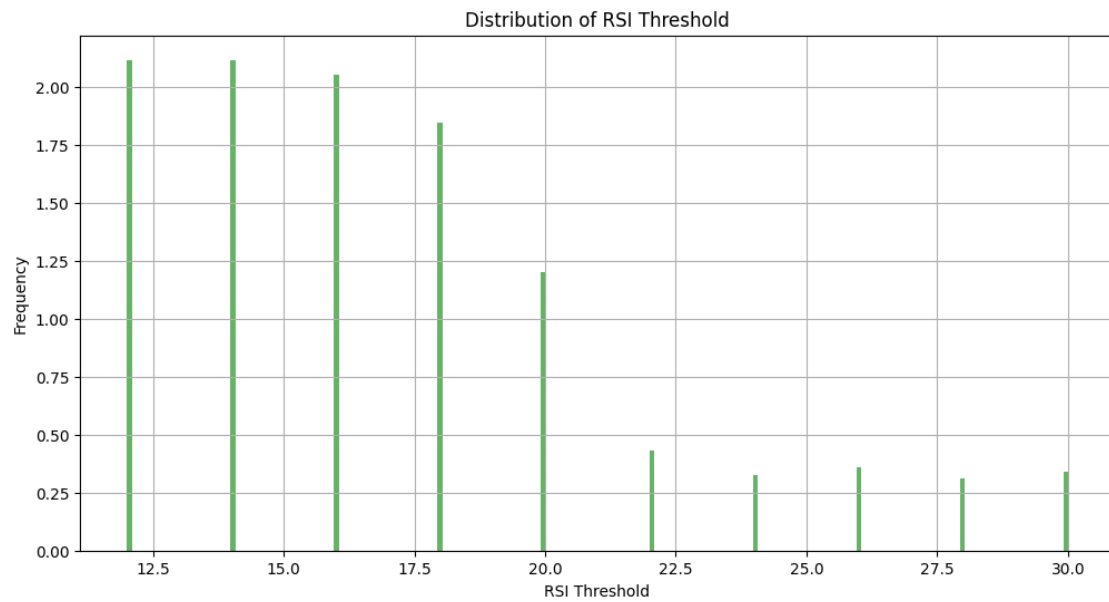
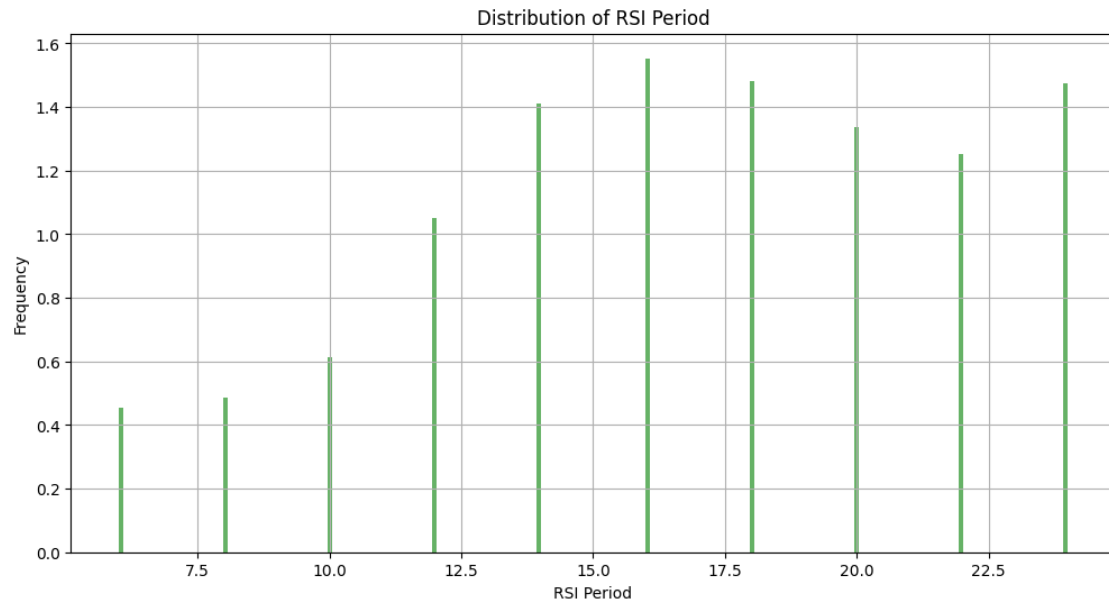
```
[34]: thresh_mix
```

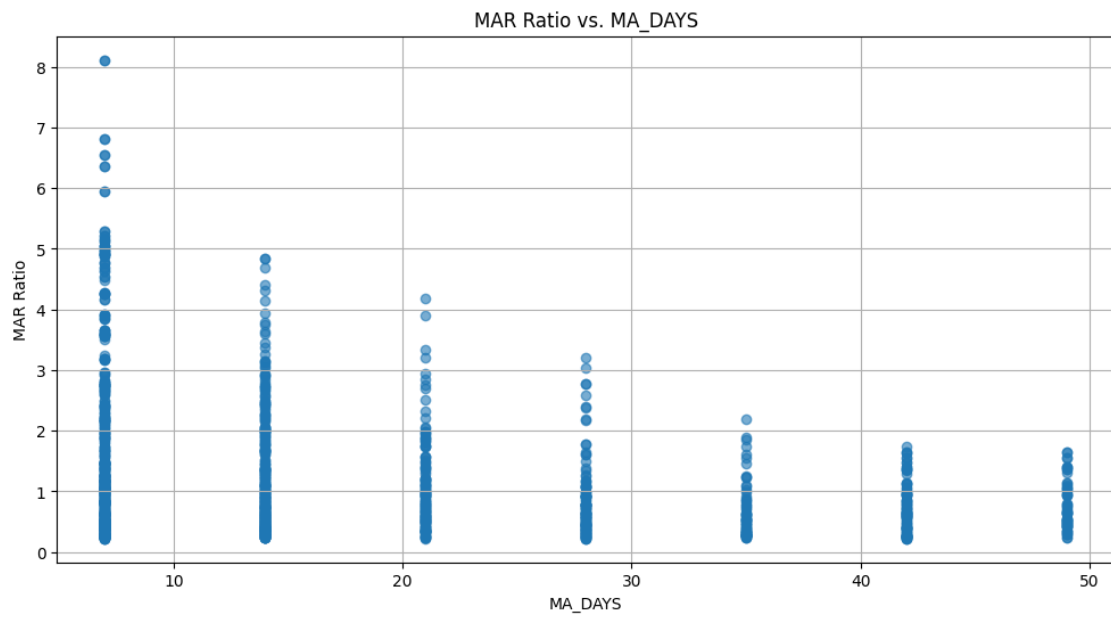
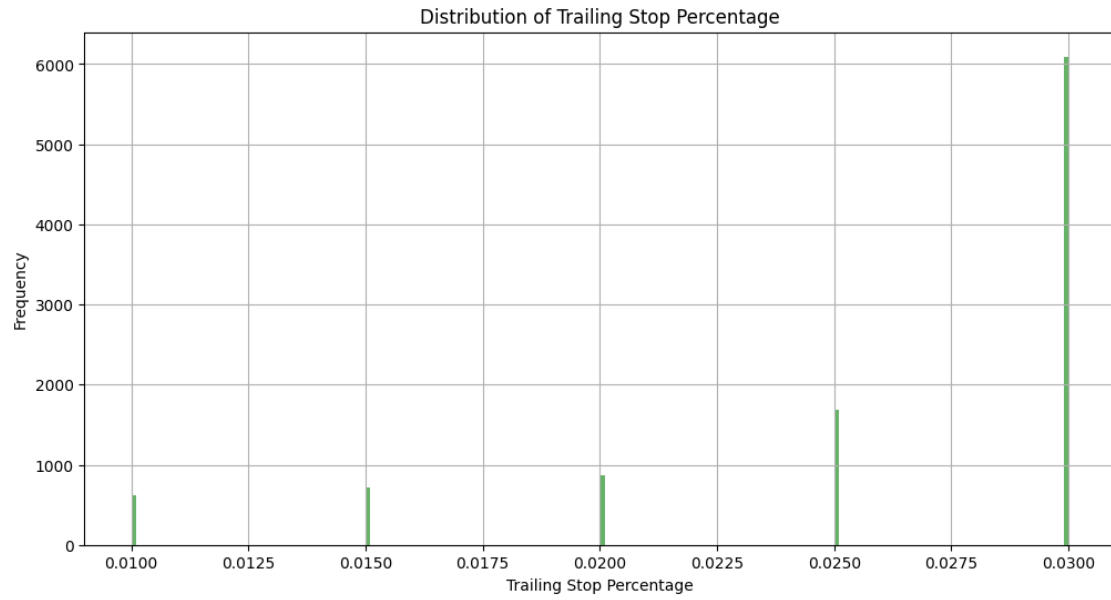
```
[34]: MAR Ratio      0.22
      CAGR           0.03
      Total Return   0.03
      dtype: float64
```

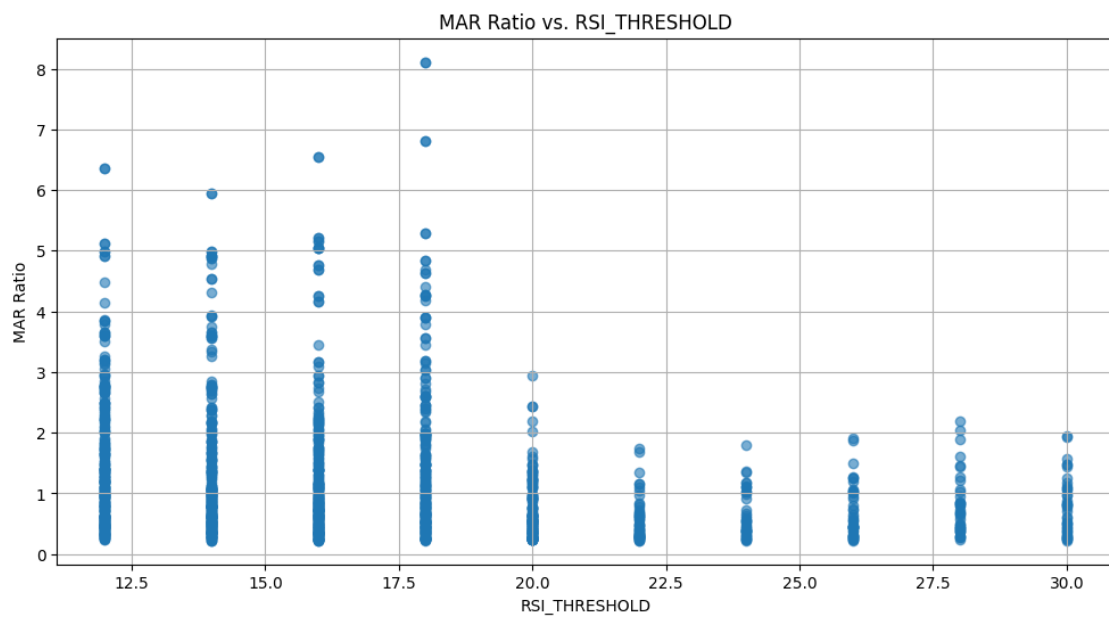
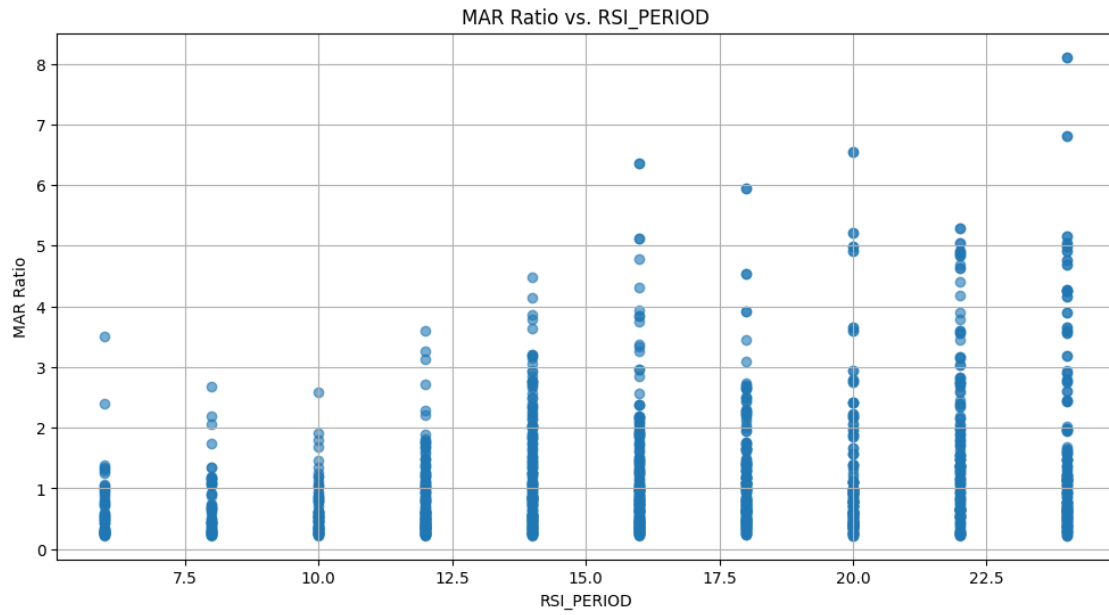


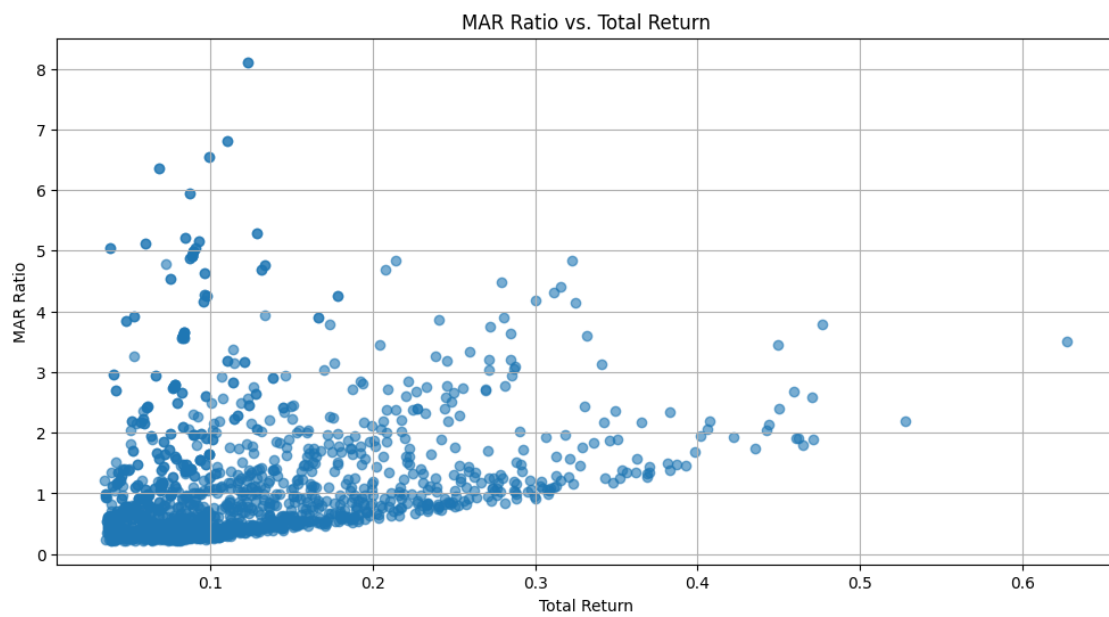
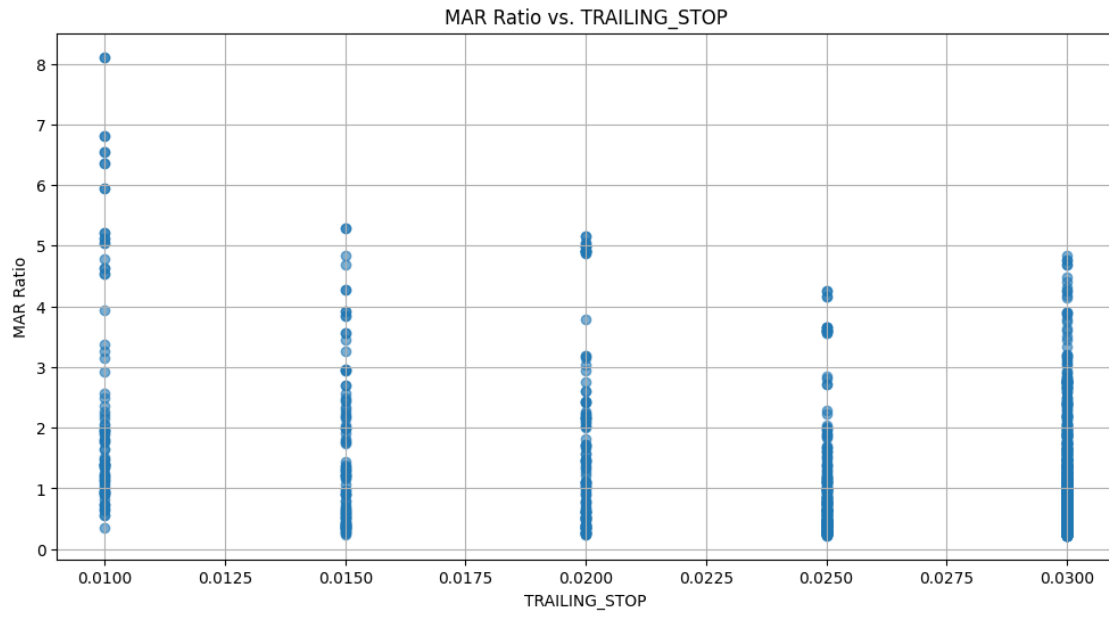
```
[35]: plot_iteration_results(  
      results_df=top10_mar,  
      )
```

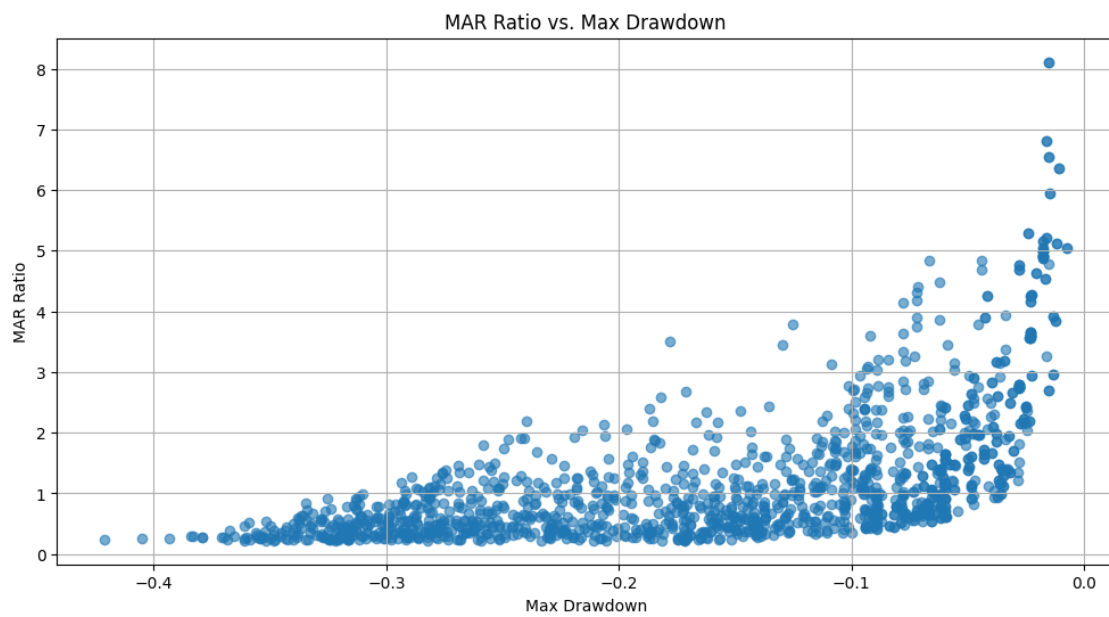
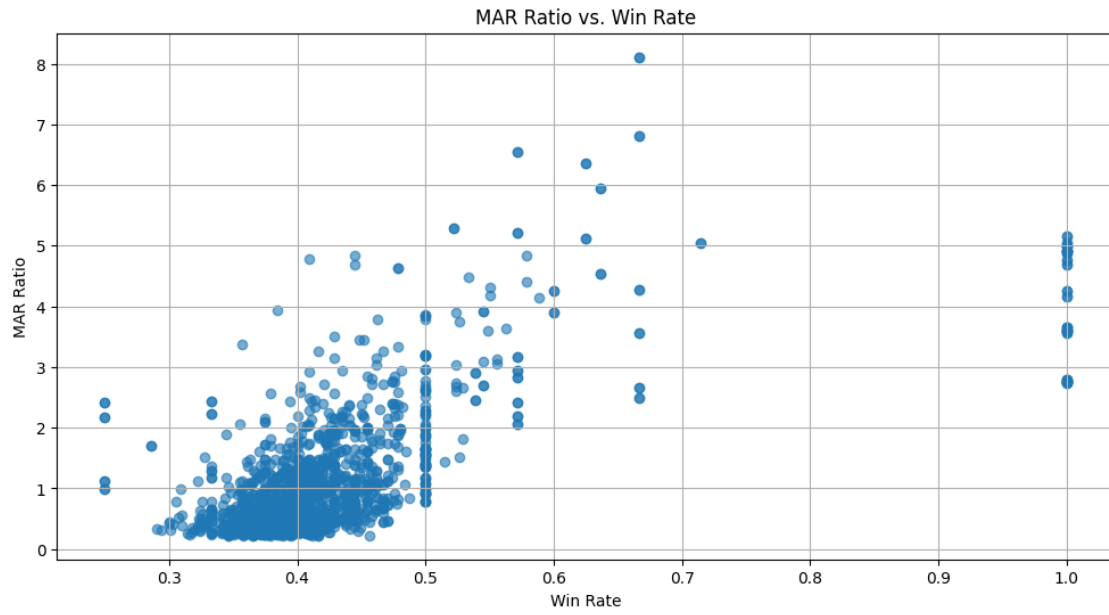


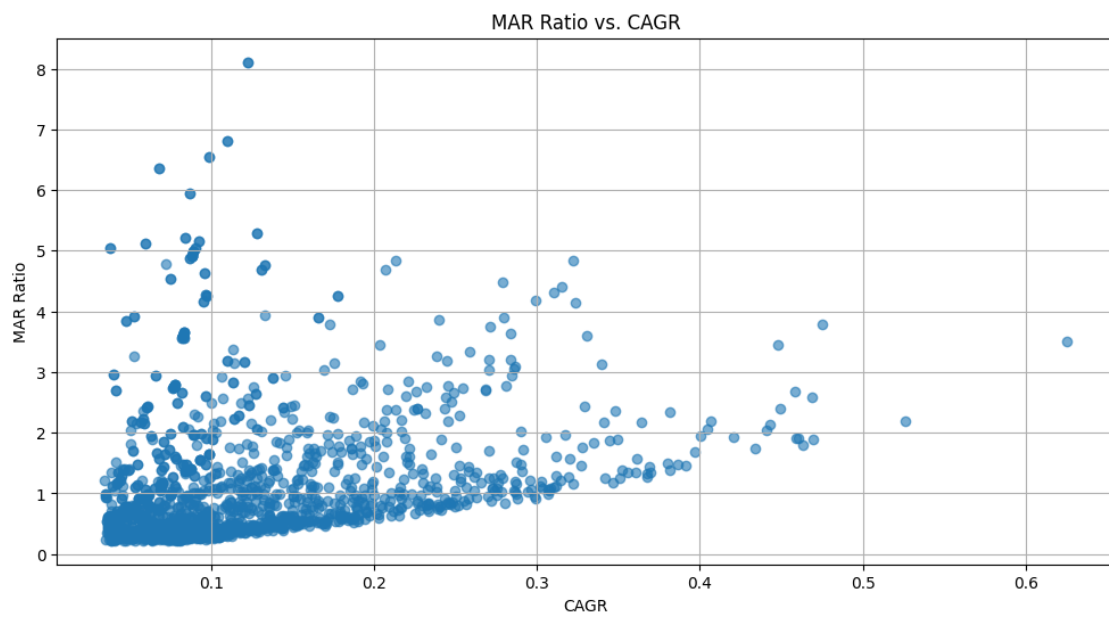
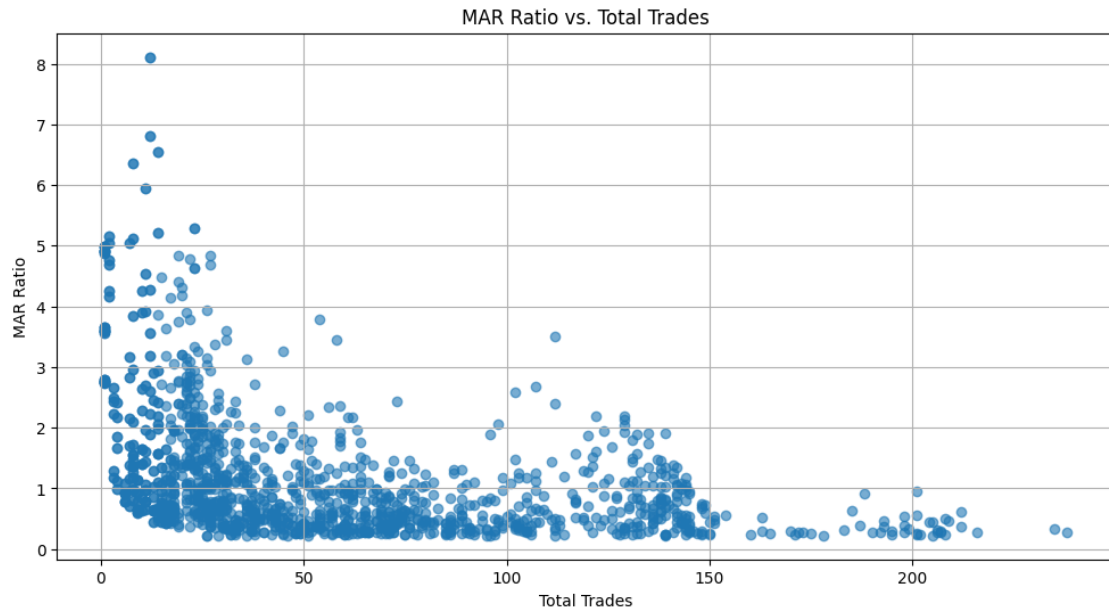


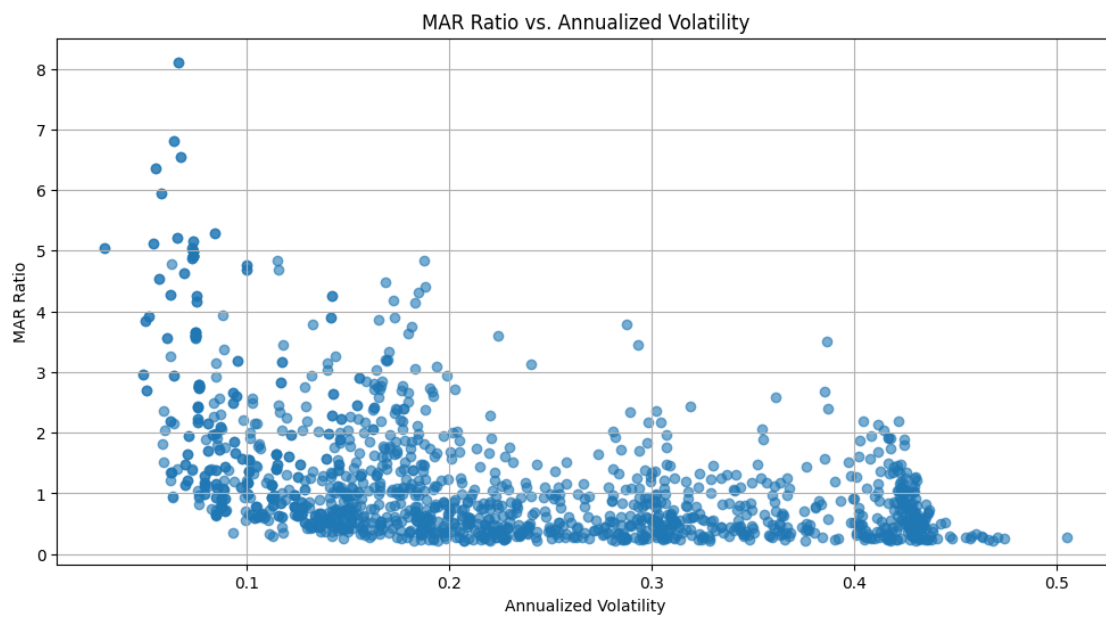
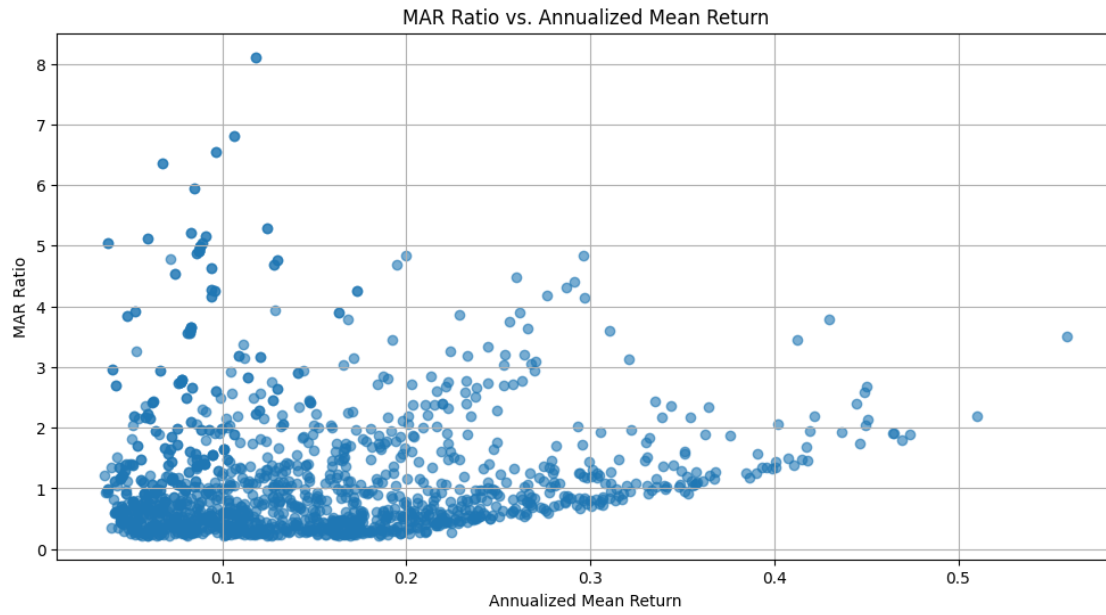


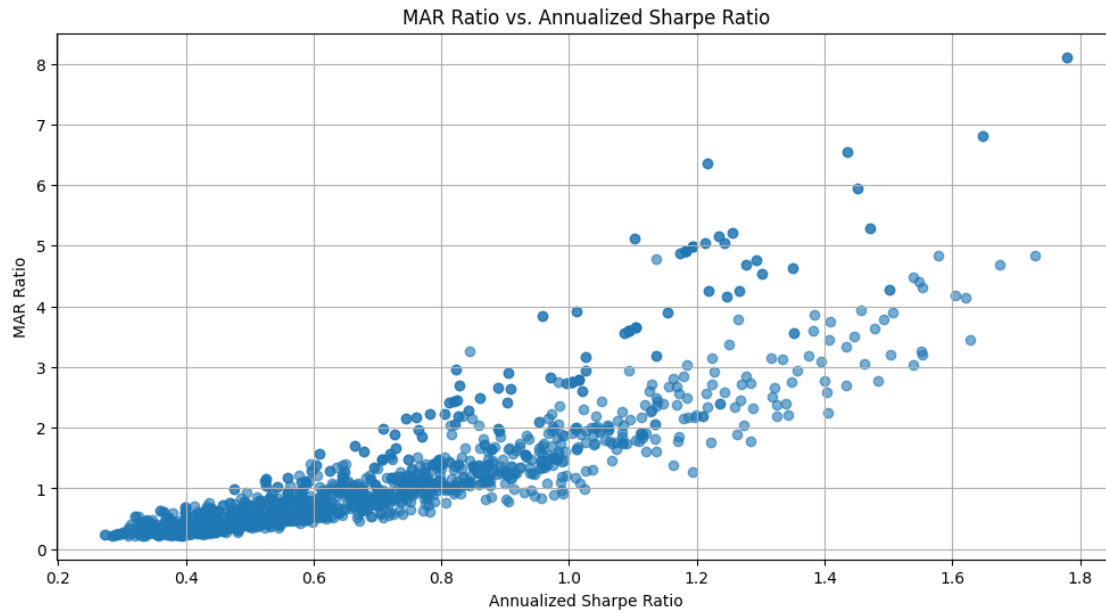












2.12 Delete existing ZIP files

```
[36]: # import os
# from pathlib import Path
# from glob import escape
# from typing import Iterable, Tuple, List

# ParamSet = Tuple[list, int, int] # (ma_days, rsi_period, rsi_threshold)

# def delete_zip_variants_batch(
#     base_dir: Path,
#     param_sets: Iterable[ParamSet],
#     dry_run: bool = False,
# ) -> List[Path]:
#     """
#     Delete all zip file variants whose filenames contain
#     ..._{ma_days}_{rsi_period}_{rsi_threshold}... .zip

#     - Matches recursively under `base_dir`.
#     - Escapes '[' and ']' in ma_days so glob works.
#     - If dry_run=True, only prints what would be deleted.

#     Returns a list of deleted (or would-delete if dry_run) Paths.
#     """
#     deleted: List[Path] = []
```

```

#     for ma_days, rsi_period, rsi_threshold in param_sets:
#         # replicate naming exactly as your exporter: str(list) e.g. "[7, 14]"
#         ma_days_str = str(ma_days)
#         # escape for glob ([], etc.)
#         ma_days_glob = escape(ma_days_str)

#         pattern = f"*_{ma_days_glob}_{rsi_period}_{rsi_threshold}*.zip"

#         for zip_file in base_dir.rglob(pattern):
#             if dry_run:
#                 print(f"[DRY-RUN] Would delete: {zip_file}")
#             else:
#                 try:
#                     os.remove(zip_file)
#                     print(f"Deleted: {zip_file}")
#                 except OSError as e:
#                     print(f"Failed to delete {zip_file}: {e}")
#                     continue
#                 deleted.append(zip_file)

#     if not deleted:
#         print("No matching zip files found for the provided parameter sets.")

#     return deleted

```

```

[37]: # from pathlib import Path
# from itertools import product
# from typing import Iterable, List, Tuple

# # Reuse the existing function you already have:
# # delete_zip_variants_batch(base_dir: Path, param_sets: Iterable[Tuple[list,
↳int, int]], dry_run: bool = False)

# def build_param_sets(
#     ma_days_options: Iterable[list],
#     rsi_period_options: Iterable[int],
#     rsi_threshold_options: Iterable[int],
# ) -> List[Tuple[list, int, int]]:
#     """
#     Cartesian product of the provided options:
#     returns [(ma_days, rsi_period, rsi_threshold), ...]
#     """
#     return [(md, rp, rt) for md, rp, rt in product(ma_days_options,
↳rsi_period_options, rsi_threshold_options)]

```

```

[38]: # # Base directory that contains year subfolders with ZIPs
# base = Path(current_directory) / "Iterations"

```

```

# # Provide every value you want included; the helper will generate all combos.
# ma_days_options = [[], [7], [14], [21], [28], [35], [42]]

# # rsi_period_options = [6, 8, 10, 12, 14, 16, 18, 20, 22, 24]
# rsi_period_options = [5, 7, 9, 11, 13, 15, 17, 19]

# rsi_threshold_options = [15, 17, 19, 21, 23, 25, 27, 29, 35]
# # rsi_threshold_options = [15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]

# param_sets = build_param_sets(
#     ma_days_options=ma_days_options,
#     rsi_period_options=rsi_period_options,
#     rsi_threshold_options=rsi_threshold_options,
# )

# # Preview first
# delete_zip_variants_batch(base, param_sets, dry_run=False)

```

2.13 Rename ZIP Files

```

[39]: # import os
# import re

# # Directory where your zip files live
# zip_dir = Path(current_directory) / "Iterations" / "2021"

# # Regex pattern - order part is optional
# pattern = re.compile(
#     r"^(?P<start>\d{4}-\d{2}-\d{2})_"
#     r"^(?P<end>\d{4}-\d{2}-\d{2})_"
#     r"^(?P<tickers>\[.*?\])_"
#     r"^(?P<ma>\[.*?\])_"
#     r"^(?P<rp>\d+)"
#     r"^(?P<rt>\d+)"
#     r"^(?P<ts>[\d.]+"
#     r"^(?:_(?P<order>.+))?" # optional order
#     r"\.zip$"
# )

# for fname in os.listdir(zip_dir):
#     if not fname.endswith(".zip"):
#         continue

#     match = pattern.match(fname)
#     if not match:

```

```

#         print(f"Skipping {fname} (does not match pattern)")
#         continue

#     # Extract fields
#     start = match.group("start")
#     end = match.group("end")
#     tickers = match.group("tickers")
#     ma = match.group("ma")
#     rp = match.group("rp")
#     rt = match.group("rt")
#     ts = match.group("ts")
#     order = match.group("order") or "market" # default if missing

#     # Replace [] with [0]
#     if ma == "[]":
#         ma = "[0]"

#     # Force ts to 3 decimal places
#     ts = f"{float(ts):.3f}"

#     # Build new filename
#     new_fname =
↪ f"{start}_{end}_{tickers}_MA-{ma}_RP-{rp}_RT-{rt}_TS-{ts}_{order}.zip"

#     # Paths
#     old_path = os.path.join(zip_dir, fname)
#     new_path = os.path.join(zip_dir, new_fname)

#     # Skip if already exists
#     if os.path.exists(new_path):
#         print(f"Skipping {fname} -> {new_fname} (already exists)")
#         continue

#     # Rename file
#     os.rename(old_path, new_path)
#     print(f"Renamed: {fname} -> {new_fname}")

```

```

[40]: # import os
# import re

# # Directory where your zip files live
# zip_dir = Path(current_directory) / "Iterations" / "2021"

# # Regex to capture the TS portion in the already-renamed format
# pattern = re.compile(
#     r"^(?P<prefix>.*_TS-)(?P<ts>[\d.]+)(?P<suffix>_+\.zip)$"
# )

```

```

# for fname in os.listdir(zip_dir):
#     if not fname.endswith(".zip"):
#         continue

#     match = pattern.match(fname)
#     if not match:
#         print(f"Skipping {fname} (does not match TS pattern)")
#         continue

#     prefix = match.group("prefix")
#     ts = match.group("ts")
#     suffix = match.group("suffix")

#     # Reformat TS to 3 decimal places
#     ts_new = f"{float(ts):.3f}"

#     new_fname = f"{prefix}{ts_new}{suffix}"

#     old_path = os.path.join(zip_dir, fname)
#     new_path = os.path.join(zip_dir, new_fname)

#     # Skip if already exists
#     if os.path.exists(new_path):
#         print(f"Skipping {fname} -> {new_fname} (already exists)")
#         continue

#     os.rename(old_path, new_path)
#     print(f"Renamed: {fname} -> {new_fname}")

```

[]:

[]:

[41]: # # Copy this <!-- INSERT_01_VIX_DF_Info_HERE --> to index_temp.md
export_track_md_deps(dep_file=dep_file, md_filename="01_VIX_DF_Info.md",
↪content=df_info_markdown(vix))

[42]: # # Copy this <!-- INSERT_01_VIX_Stats_HERE --> to index_temp.md
export_track_md_deps(dep_file=dep_file, md_filename="01_VIX_Stats.md",
↪content=vix_stats.to_markdown(floatfmt=".2f"))