```
Buy & Hold: Sortino=0.8, Sharpe=0.6, Calmar=0.2
```

Investing in the S&P 500: Is Dollar-Cost Averaging or Buy-and-Hold a Better Approach for Long-Term Gains?

Table of Contents

- Chapter 1. ETF DATA
  - Section 1.1. Extracting SPDR DATA
  - Section 1.2. Bitcoin - Feature Selection
  - Sub Section 1.2.1. Bitcoin - Data Transformation
    - Sub Section 1.2.1.1. Bitcoin - Stationarity
    - Sub Section 1.2.1.2.1DA - Monthly Data Extraction
    - Sub Section 1.2.1.3DA - Data Transformation
  - Sub Section 1.2.2. DCA - Feature Selection
- Chapter 2. Analyzing Simple Returns
  - Section 2.1. Historical Simple Returns
    - Sub Section 2.1.1. Plotting Simple Returns
    - Sub Section 2.1.2. Accumulation Returns
    - Sub Section 2.1.3. Drawdowns
  - Section 2.2. Performance for a selected time range
    - Sub Section 2.2.1. Examples
    - Sub Section 2.2.2. Accumulation returns for different Time Frames
    - Sub Section 2.2.3. Yearly Data
    - Sub Section 2.2.4. Drawdowns
    - Sub Section 2.2.5. Maximum Drawdown


```
# Setup - Suppress warnings
import warnings
warnings.filterwarnings('ignore')

# Core data manipulation
import numpy as np
import pandas as pd
import datetime as dt

# Statistical analysis
from scipy.stats import skew, kurtosis, jarque_bera
from statsmodels.tsa.stationarity import adfuller, kpss
from statsmodels.tsa.stationarity.utils import InterpolationWarning

# Machine learning preprocessing
from sklearn.preprocessing import StandardScaler, MinMaxScaler

# Visualization
import matplotlib.pyplot as plt
import seaborn as sns
```

```
%pip install yfinance
```

```
Note: you may need to restart the kernel to use updated packages.
```

1. ETL Data

1.1. Extracting SAP500 DATA

```
# Download SPY data
spy = yf.download('SPY', start='1993-01-29')

# Display basic information about the dataset
print("SPY ETF Data Shape:", spy.shape)
print("Date range:", spy.index[0], "to", spy.index[-1])

# Check for missing values
print("\nMissing values:")
print(spy.isnull().sum)

# Display first few rows
spy.head()
```

```
[*********100%***********]  1 of 1 completed
SPY ETF Data Shape: (7811, 8)
Date range: 1993-01-29 00:00:00 to 2024-01-31 00:00:00

Missing values:
Adj Close    0
Close        0
High         0
Low          0
Open         0
Volume       0
Dividends    0
Stock Splits    0
dtype: int64
```

	Adj Close	Close	High	Low	Open	Volume	Dividends	Stock Splits
Date								
1993-01-29	25.00000	43.93750	43.96875	43.75000	43.93750	1003200	0.000000	0.000000
1993-02-01	25.27344	44.40625	44.437500	43.968750	43.968750	480500	0.000000	0.000000
1993-02-02	25.35938	44.556250	44.562500	44.218750	44.437500	201400	0.000000	0.000000
1993-02-03	25.78906	45.312500	45.406250	44.562500	44.562500	267500	0.000000	0.000000
1993-02-04	26.01563	45.712500	45.812500	45.281250	45.406250	467800	0.000000	0.000000

```
# Save raw data
spy.to_csv('SPY_raw_data.csv')
print("Data saved to SPY_raw_data.csv")
```

```
Data saved to SPY_raw_data.csv
```

1.2. Bitcoin - Feature Selection

```
# Load the data
btc = pd.read_csv('BTC-USD_raw_data.csv', header=0, index_col=0, parse_dates=True)
btc = btc.sort_index()

# Display basic information
print("BTC-USD Data Shape:", btc.shape)
print("Date range:", btc.index[0], "to", btc.index[-1])

# Check for missing values
print("\nMissing values:")
print(btc.isnull().sum)

# Display first few rows
btc.head()
```

```
BTC-USD Data Shape: (3652, 6)
Date range: 2014-09-17 00:00:00 to 2024-09-17 00:00:00

Missing values:
Adj Close    0
Close        0
High         0
Low          0
Open         0
Volume       0
dtype: int64
```

	Adj Close	Close	High	Low	Open	Volume
Date						
2014-09-17	465.864015	465.864015	468.174011	452.421997	457.334015	21056800
2014-09-18	456.859985	456.859985	456.859985	413.390991	424.440002	34483200
2014-09-19	424.103001	424.103001	427.834991	384.532013	394.795990	37919700
2014-09-20	394.673991	394.673991	423.295990	389.882996	423.295990	36863600
2014-09-21	408.084991	408.084991	412.425995	393.181992	394.673991	26580100

1.2.1. Bitcoin - Data Transformation

```
# Create a copy for transformations
btc_transformed = btc.copy()

# Calculate various transformations
btc_transformed['Log_Close'] = np.log(btc_transformed['Close'])
btc_transformed['Log_Volume'] = np.log(btc_transformed['Volume'])
btc_transformed['Daily_Return'] = btc_transformed['Close'].pct_change()
btc_transformed['Log_Return'] = np.log(btc_transformed['Close'] / btc_transformed['Close'].shift(1))
btc_transformed['High_Low_Ratio'] = btc_transformed['High'] / btc_transformed['Low']
btc_transformed['Close_Open_Ratio'] = btc_transformed['Close'] / btc_transformed['Open']

# Display the new columns
print("Transformations applied:")
print("Original columns:", btc.shape[1])
print("Transformed columns:", btc_transformed.shape[1])
print("\nNew columns:")
print(set(btc_transformed.columns) - set(btc.columns))

# Check for any infinite values after transformations
print("\nInfinite values check:")
print(np.isinf(btc_transformed).sum())

# Display first few rows with new columns
btc_transformed.head()
```

```
Transformations applied:
Original columns: 6
Transformed columns: 12

New columns:
{'Close_Open_Ratio', 'Log_Return', 'Daily_Return', 'High_Low_Ratio', 'Log_Volume', 'Log_Close'}

Infinite values check:
Adj Close       0
Close           0
High            0
Low             0
Open            0
Volume          0
Log_Close       0
Log_Volume      0
Daily_Return    1
Log_Return      1
High_Low_Ratio  0
Close_Open_Ratio    0
dtype: int64
```

	Adj Close	Close	High	Low	Open	Volume	Log_Close	Log_Volume	Daily_Return	Log_Return	High_Low_Ratio	Close_Open_Ratio
Date												
2014-09-17	465.864015	465.864015	468.174011	452.421997	457.334015	21056800	6.143684	16.864948	NaN	NaN	1.034848	1.018900
2014-09-18	456.859985	456.859985	456.859985	413.390991	424.440002	34483200	6.124683	17.356308	-0.019324	-0.019511	1.105000	1.076660
2014-09-19	424.103001	424.103001	427.834991	384.532013	394.795990	37919700	6.049733	17.452983	-0.071688	-0.074351	1.112600	1.074000
2014-09-20	394.673991	394.673991	423.295990	389.882996	423.295990	36863600	5.978886	17.422155	-0.069388	-0.071915	1.085900	0.932400
2014-09-21	408.084991	408.084991	412.425995	393.181992	394.673991	26580100	6.011266	17.095068	0.033978	0.033412	1.048900	1.033900

```
# Save the transformed data
btc_transformed.to_csv('BTC-USD_transformed.csv')
print("Transformed data saved to BTC-USD_transformed.csv")
```

```
Transformed data saved to BTC-USD_transformed.csv
```

1.2.1.1. Bitcoin - Stationarity

```
# Load the transformed data
btc = pd.read_csv('BTC-USD_transformed.csv', header=0, index_col=0, parse_dates=True)
btc = btc.sort_index()

# Function to perform stationarity tests
def stationarity_tests(series, name='Series'):
    print(f"Stationarity tests for {name}")
    print("="*50)

    # Remove NaN values for testing
    clean_series = series.dropna()

    # Augmented Dickey-Fuller test
    adf_result = adfuller(clean_series)
    print(f"ADF Statistic: {adf_result[0]:.4f}")
    print(f"p-value: {adf_result[1]:.4f}")
    print(f"Critical values:")
    for key, value in adf_result[4].items:
        print f"    {key}: {value:.4f}")

    # KPSS test
    kpss_result = kpss(clean_series)
    print(f"KPSS Statistic: {kpss_result[0]:.4f}")
    print(f"p-value: {kpss_result[1]:.4f}")
    print(f"Critical values:")
    for key, value in kpss_result[3].items:
        print f"    {key}: {value:.4f}")

    print("\n")

# Test stationarity for key variables
stationarity_tests(btc['Close'], 'Close')
stationarity_tests(btc['Log_Close'], 'Log_Close')
stationarity_tests(btc['Daily_Return'], 'Daily_Return')
stationarity_tests(btc['Log_Return'], 'Log_Return')
```

```
Stationarity tests for Close
==================================================
ADF Statistic: 0.1429
p-value: 0.9688
Critical values:
    1%: -3.4325
    5%: -2.8625
    10%: -2.5673
KPSS Statistic: 7.3440
p-value: 0.0100
Critical values:
    10%: 0.3470
    5%: 0.4630
    2.5%: 0.5740
    1%: 0.7390

Stationarity tests for Log_Close
==================================================
ADF Statistic: -1.8164
p-value: 0.3713
Critical values:
    1%: -3.4325
    5%: -2.8625
    10%: -2.5673
KPSS Statistic: 7.3440
p-value: 0.0100
Critical values:
    10%: 0.3470
    5%: 0.4630
    2.5%: 0.5740
    1%: 0.7390

Stationarity tests for Daily_Return
==================================================
ADF Statistic: -14.3000
p-value: 0.0000
Critical values:
    1%: -3.4325
    5%: -2.8625
    10%: -2.5673
KPSS Statistic: 0.4000
p-value: 0.0700
Critical values:
    10%: 0.3470
    5%: 0.4630
    2.5%: 0.5740
    1%: 0.7390

Stationarity tests for Log_Return
==================================================
ADF Statistic: -14.3000
p-value: 0.0000
Critical values:
    1%: -3.4325
    5%: -2.8625
    10%: -2.5673
KPSS Statistic: 0.4000
p-value: 0.0700
Critical values:
    10%: 0.3470
    5%: 0.4630
    2.5%: 0.5740
    1%: 0.7390
```

1.2.1.2.1DA - Monthly Data Extraction

```
# Load the transformed data
btc = pd.read_csv('BTC-USD_transformed.csv', header=0, index_col=0, parse_dates=True)
btc = btc.sort_index()

# Check the current frequency
print("Current frequency:")
print("Index freq:", btc.index.freq)
print("Inferred frequency:", pd.infer_freq(btc.index))

# Create monthly aggregation
monthly = btc.resample('M').agg({
    'Adj Close': 'last',
    'Close': 'last',
    'High': 'max',
    'Low': 'min',
    'Open': 'first',
    'Volume': 'sum',
    'Log_Close': 'last',
    'Log_Volume': 'last',
    'Daily_Return': 'sum',
    'Log_Return': 'sum',
    'High_Low_Ratio': 'mean',
    'Close_Open_Ratio': 'last'
})

print("\nMonthly data shape:", monthly.shape)
print("Date range:", monthly.index[0], "to", monthly.index[-1])

# Save monthly data
monthly.to_csv('BTC-USD_monthly.csv')
print("Monthly data saved to BTC-USD_monthly.csv")
```

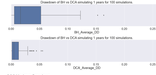
```
Current frequency:
Index freq: None
Inferred frequency: None

Monthly data shape: (121, 12)
Date range: 2014-09-30 00:00:00 to 2024-09-30 00:00:00
Monthly data saved to BTC-USD_monthly.csv
```

1.2.1.3DA - Data Transformation

```
# Load monthly data
monthly = pd.read_csv('BTC-USD_monthly.csv', header=0, index_col=0, parse_dates=True)
monthly = monthly.sort_index()

# Create a copy for transformations
monthly_transformed = monthly.copy()

# Calculate monthly transformations
monthly_transformed['Monthly_Log_Close'] = np.log(monthly_transformed['Close'])
monthly_transformed['Monthly_Log_Volume'] = np.log(monthly_transformed['Volume'])
monthly_transformed['Monthly_Return'] = monthly_transformed['Close'].pct_change()
monthly_transformed['Monthly_Log_Return'] = np.log(monthly_transformed['Close'] / monthly_transformed['Close'].shift(1))
monthly_transformed['Monthly_High_Low_Ratio'] = monthly_transformed['High'] / monthly_transformed['Low']
monthly_transformed['Monthly_Close_Open_Ratio'] = monthly_transformed['Close'] / monthly_transformed['Open']

# Display the new columns
print("Monthly transformations applied:")
print("Original columns:", monthly.shape[1])
print("Transformed columns:", monthly_transformed.shape[1])
print("\nNew columns:")
print(set(monthly_transformed.columns) - set(monthly.columns))

# Check for any infinite values after transformations
print("\nInfinite values check:")
print(np.isinf(monthly_transformed).sum())

# Display first few rows with new columns
monthly_transformed.head()
```

```
Monthly transformations applied:
Original columns: 12
Transformed columns: 18

New columns:
{'Monthly_Close_Open_Ratio', 'Monthly_Log_Return', 'Monthly_Return', 'Monthly_High_Low_Ratio', 'Monthly_Log_Volume', 'Monthly_Log_Close'}

Infinite values check:
Adj Close                0
Close                    0
High                     0
Low                      0
Open                     0
Volume                   0
Log_Close                0
Log_Volume               0
Daily_Return             1
Log_Return               1
High_Low_Ratio           0
Close_Open_Ratio         0
Monthly_Log_Close        0
Monthly_Log_Volume       0
Monthly_Return           1
Monthly_Log_Return       1
Monthly_High_Low_Ratio   0
Monthly_Close_Open_Ratio 0
dtype: int64
```

	Adj Close	Close	High	Low	Open	Volume	Log_Close	Log_Volume	Daily_Return	Log_Return	High_Low_Ratio	Close_Open_Ratio
Date												
2014-09-30	384.820007	384.820007	424.440002	384.532013	457.334015	95348600	5.953243	18.372703	-0.157168	-0.171894	1.103900	0.841500
2014-10-31	338.321991	338.321991	386.475991	320.509003	384.820312	146099900	5.823046	18.799159	-0.120908	-0.128400	1.206000	0.879000
2014-11-30	378.064991	378.064991	executed_price	executed_size	executed_value	commission	sl_price	sl_executed	tp_price	tp_executed	order_ref	data_name
2014-12-31	320.884003	320.884003	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	1	BTC-USD
2015-01-31	217.464991	217.464991	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	2	BTC-USD
2015-02-28	216.867996	216.867996	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	3	BTC-USD
2015-03-31	278.172992	278.172992	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	4	BTC-USD
2015-04-30	236.239006	236.239006	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	5	BTC-USD
2015-05-31	229.131004	229.131004	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	6	BTC-USD
2015-06-30	235.139999	235.139999	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	7	BTC-USD
2015-07-31	274.473999	274.473999	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	8	BTC-USD
2015-08-31	277.381991	277.381991	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	9	BTC-USD
2015-09-30	232.139999	232.139999	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	10	BTC-USD
2015-10-31	314.076991	314.076991	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	11	BTC-USD
2015-11-30	377.591003	377.591003	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	12	BTC-USD
2015-12-31	430.566986	430.566986	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	13	BTC-USD
2016-01-31	433.865000	433.865000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	14	BTC-USD
2016-02-29	371.911000	371.911000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	15	BTC-USD
2016-03-31	416.078000	416.078000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	16	BTC-USD
2016-04-30	453.875000	453.875000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	17	BTC-USD
2016-05-31	451.188000	451.188000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	18	BTC-USD
2016-06-30	651.453000	651.453000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	19	BTC-USD
2016-07-31	664.641000	664.641000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	20	BTC-USD
2016-08-31	596.100000	596.100000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	21	BTC-USD
2016-09-30	578.759000	578.759000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	22	BTC-USD
2016-10-31	703.059000	703.059000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	23	BTC-USD
2016-11-30	746.813000	746.813000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	24	BTC-USD
2016-12-31	963.860000	963.860000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	25	BTC-USD
2017-01-31	1151.410000	1151.410000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	26	BTC-USD
2017-02-28	1065.140000	1065.140000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	27	BTC-USD
2017-03-31	1352.080000	1352.080000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	28	BTC-USD
2017-04-30	1293.450000	1293.450000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	29	BTC-USD
2017-05-31	2348.410000	2348.410000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	30	BTC-USD
2017-06-30	2480.090000	2480.090000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	31	BTC-USD
2017-07-31	2817.000000	2817.000000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	32	BTC-USD
2017-08-31	4469.060000	4469.060000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	33	BTC-USD
2017-09-30	4337.440000	4337.440000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	34	BTC-USD
2017-10-31	6427.010000	6427.010000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	35	BTC-USD
2017-11-30	7708.230000	7708.230000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	36	BTC-USD
2017-12-31	14156.400000	14156.400000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	37	BTC-USD
2018-01-31	10092.600000	10092.600000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	38	BTC-USD
2018-02-28	6935.270000	6935.270000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	39	BTC-USD
2018-03-31	7011.750000	7011.750000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	40	BTC-USD
2018-04-30	9811.160000	9811.160000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	41	BTC-USD
2018-05-31	7635.180000	7635.180000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	42	BTC-USD
2018-06-30	6462.620000	6462.620000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	43	BTC-USD
2018-07-31	7916.270000	7916.270000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	44	BTC-USD
2018-08-31	6277.140000	6277.140000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	45	BTC-USD
2018-09-30	6638.320000	6638.320000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	46	BTC-USD
2018-10-31	6619.270000	6619.270000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	47	BTC-USD
2018-11-30	4085.180000	4085.180000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	48	BTC-USD
2018-12-31	3742.700000	3742.700000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	49	BTC-USD
2019-01-31	3701.030000	3701.030000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	50	BTC-USD
2019-02-28	3852.560000	3852.560000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	51	BTC-USD
2019-03-31	4168.700000	4168.700000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	52	BTC-USD
2019-04-30	5267.480000	5267.480000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	53	BTC-USD
2019-05-31	8616.540000	8616.540000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	54	BTC-USD
2019-06-30	10786.100000	10786.100000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	55	BTC-USD
2019-07-31	9592.900000	9592.900000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	56	BTC-USD
2019-08-31	9595.900000	9595.900000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	57	BTC-USD
2019-09-30	10250.100000	10250.100000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	58	BTC-USD
2019-10-31	9546.000000	9546.000000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	59	BTC-USD
2019-11-30	7326.000000	7326.000000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	60	BTC-USD
2019-12-31	7193.600000	7193.600000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	61	BTC-USD
2020-01-31	9350.000000	9350.000000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	62	BTC-USD
2020-02-29	9651.300000	9651.300000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	63	BTC-USD
2020-03-31	6436.200000	6436.200000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	64	BTC-USD
2020-04-30	7195.200000	7195.200000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	65	BTC-USD
2020-05-31	8821.400000	8821.400000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	66	BTC-USD
2020-06-30	9137.100000	9137.100000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	67	BTC-USD
2020-07-31	11365.100000	11365.100000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	68	BTC-USD
2020-08-31	11699.500000	11699.500000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	69	BTC-USD
2020-09-30	10771.100000	10771.100000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	70	BTC-USD
2020-10-31	13657.200000	13657.200000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	71	BTC-USD
2020-11-30	19699.000000	19699.000000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	72	BTC-USD
2020-12-31	28949.200000	28949.200000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	73	BTC-USD
2021-01-31	34371.300000	34371.300000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	74	BTC-USD
2021-02-28	46650.200000	46650.200000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	75	BTC-USD
2021-03-31	58918.800000	58918.800000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	76	BTC-USD
2021-04-30	57750.100000	57750.100000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	77	BTC-USD
2021-05-31	37314.000000	37314.000000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	78	BTC-USD
2021-06-30	35040.800000	35040.800000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	79	BTC-USD
2021-07-31	41731.000000	41731.000000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	80	BTC-USD
2021-08-31	47135.900000	47135.900000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	81	BTC-USD
2021-09-30	43790.900000	43790.900000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	82	BTC-USD
2021-10-31	54771.300000	54771.300000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	83	BTC-USD
2021-11-30	57010.200000	57010.200000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	84	BTC-USD
2021-12-31	46306.400000	46306.400000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	85	BTC-USD
2022-01-31	38483.000000	38483.000000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	86	BTC-USD
2022-02-28	43790.900000	43790.900000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	87	BTC-USD
2022-03-31	47135.900000	47135.900000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	88	BTC-USD
2022-04-30	54771.300000	54771.300000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	89	BTC-USD
2022-05-31	57010.200000	57010.200000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	90	BTC-USD
2022-06-30	46306.400000	46306.400000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	91	BTC-USD
2022-07-31	38483.000000	38483.000000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	92	BTC-USD
2022-08-31	43790.900000	43790.900000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	93	BTC-USD
2022-09-30	47135.900000	47135.900000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	94	BTC-USD
2022-10-31	54771.300000	54771.300000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	95	BTC-USD
2022-11-30	57010.200000	57010.200000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	96	BTC-USD
2022-12-31	46306.400000	46306.400000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	97	BTC-USD
2023-01-31	38483.000000	38483.000000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	98	BTC-USD
2023-02-28	43790.900000	43790.900000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	99	BTC-USD
2023-03-31	47135.900000	47135.900000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	100	BTC-USD
2023-04-30	54771.300000	54771.300000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	101	BTC-USD
2023-05-31	57010.200000	57010.200000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	102	BTC-USD
2023-06-30	46306.400000	46306.400000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	103	BTC-USD
2023-07-31	38483.000000	38483.000000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	104	BTC-USD
2023-08-31	43790.900000	43790.900000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	105	BTC-USD
2023-09-30	47135.900000	47135.900000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	106	BTC-USD
2023-10-31	54771.300000	54771.300000	0.0	1.000000	1.000000	0.000000	0.0	0.0	0.0	0.0	107	BTC-USD



2.2.3 Maximum Drawdown

```
# Maximum Drawdown
max_drawdown_BH = max(BH_DD)
max_drawdown_DCA = max(DCA_DD)
cat("Maximum Drawdown of BH_MCD vs DCA_MCD simulating 1 years for 100 simulations.")
# Plot boxplot comparison
par(mfrow=c(2,1), mar=c(4,4,2,1))
boxplot(max_drawdown_BH, horizontal=TRUE, col="lightblue", main="Maximum Drawdown of BH_MCD vs DCA_MCD simulating 1 years for 100 simulations.", xlim=c(0,0.4))
boxplot(max_drawdown_DCA, horizontal=TRUE, col="lightgreen", xlim=c(0,0.4))
par(mfrow=c(1,1))
```

```
Maximum Drawdown of BH_MCD vs DCA_MCD simulating 1 years for 100 simulations.
```

3. The Best of Both Worlds: A study of Efficient Portfolios that mix DCA and buy and hold on S&P500 with treasuries yield.

```
# Load data
data_path = "data/"
SP500_data = read.csv(paste0(data_path, "sp500_data.csv"))
head(SP500_data)
```

```
        Date  Adj.Close
1 1928-12-31 17.590000
2 1929-01-31 18.110000
3 1929-02-28 17.110000
4 1929-03-31 18.170000
5 1929-04-30 18.330000
6 1929-05-31 18.770000
```

```
# Convert to xts
SP500_xts = xts(SP500_data[,-1], order.by=as.Date(SP500_data$Date))
colnames(SP500_xts) = "SP500"
head(SP500_xts)
```

```
           SP500
1928-12-31 17.59
1929-01-31 18.11
1929-02-28 17.11
1929-03-31 18.17
1929-04-30 18.33
1929-05-31 18.77
```

```
# Load treasury yield data
treasury_data = read.csv(paste0(data_path, "treasury_yield.csv"))
head(treasury_data)
```

```
        Date  Adj.Close
1 1928-12-31 0.03000000
2 1929-01-31 0.03000000
3 1929-02-28 0.03000000
4 1929-03-31 0.03000000
5 1929-04-30 0.03000000
6 1929-05-31 0.03000000
```

```
# Convert to xts
treasury_xts = xts(treasury_data[,-1], order.by=as.Date(treasury_data$Date))
colnames(treasury_xts) = "treasury_yield"
head(treasury_xts)
```

```
           treasury_yield
1928-12-31           0.03
1929-01-31           0.03
1929-02-28           0.03
1929-03-31           0.03
1929-04-30           0.03
1929-05-31           0.03
```

```
# Merge data
SP500_treasury = merge(SP500_xts, treasury_xts)
head(SP500_treasury)
```

```
           SP500 treasury_yield
1928-12-31 17.59           0.03
1929-01-31 18.11           0.03
1929-02-28 17.11           0.03
1929-03-31 18.17           0.03
1929-04-30 18.33           0.03
1929-05-31 18.77           0.03
```

```
# Calculate monthly returns
SP500_returns = diff(log(SP500_treasury$SP500))
treasury_returns = diff(log(SP500_treasury$treasury_yield))
colnames(SP500_returns) = "SP500_return"
colnames(treasury_returns) = "treasury_return"
head(SP500_returns)
head(treasury_returns)
```

```
           SP500_return
1929-01-31           NA
1929-02-28           NA
1929-03-31           NA
1929-04-30           NA
1929-05-31           NA
1929-06-31           NA
           treasury_return
1929-01-31              NA
1929-02-28              NA
1929-03-31              NA
1929-04-30              NA
1929-05-31              NA
1929-06-31              NA
```

```
# Remove first observation
SP500_returns = SP500_returns[-1]
treasury_returns = treasury_returns[-1]
head(SP500_returns)
head(treasury_returns)
```

```
           SP500_return
1929-02-28           NA
1929-03-31           NA
1929-04-30           NA
1929-05-31           NA
1929-06-31           NA
1929-07-31           NA
           treasury_return
1929-02-28              NA
1929-03-31              NA
1929-04-30              NA
1929-05-31              NA
1929-06-31              NA
1929-07-31              NA
```

3.1. Historical Data

```
# Calculate mean and standard deviation
mu_SP = mean(SP500_return)
sigma_SP = sd(SP500_return)
mu_treasury = mean(treasury_return)
sigma_treasury = sd(treasury_return)
cat("SP500 - Mean:", round(mu_SP, 4), "SD:", round(sigma_SP, 4), "\n")
cat("Treasury - Mean:", round(mu_treasury, 4), "SD:", round(sigma_treasury, 4), "\n")
```

```
SP500 - Mean: 0.0066 SD: 0.0543
Treasury - Mean: 0.0033 SD: 0.0033
```

```
# Create matrix for simulation
returns_matrix = cbind(SP500_return, treasury_return)
colnames(returns_matrix) = c("SP500", "treasury")
head(returns_matrix)
```

```
           SP500 treasury
1929-02-28    NA       NA
1929-03-31    NA       NA
1929-04-30    NA       NA
1929-05-31    NA       NA
1929-06-31    NA       NA
1929-07-31    NA       NA
```

```
# Check correlation
cor(returns_matrix, use="complete.obs")
```

```
           SP500 treasury
SP500      1.000    0.000
treasury   0.000    1.000
```

3.2. Last 40 years

3.3. Last 20 years

3.4. Last 10 years

3.5. Last 5 years

4. Conclusion