In [12]:
```
import pandas as pd
import numpy as np

import statsmodels.api as sm
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.ar_model import AutoReg


pd.set_option('display.max_columns', None)


# #Test on BTC data
BTC = pd.read_csv('data/BTC_daily.csv', index_col=0, parse_dates=True)
ETH = pd.read_csv('data/ETH_daily.csv', index_col=0, parse_dates=True)
```

In [13]:
```
#Concact 2 close price
price_data = [BTC["Close"], ETH["Close"]]
headers = ["BTC", "ETH"]
price_df = pd.concat(price_data, axis=1, keys=headers).dropna()
display(price_df)
```

date	BTC	ETH
2022-09-01	20133.65	1586.23
2022-09-02	19953.74	1575.69
2022-09-03	19835.47	1557.70
2022-09-04	20004.73	1579.04
2022-09-05	19794.58	1618.01
...	...	...
2024-07-28	68244.30	3269.98
2024-07-29	66771.45	3317.55
2024-07-30	66169.68	3278.27
2024-07-31	64609.62	3231.81
2024-08-01	65288.18	3200.54

701 rows × 2 columns

In [14]:
```
# Regression
x = price_df['BTC']
Y = price_df['ETH']
x = sm.add_constant(x)
model = sm.OLS(Y, x)
result = model.fit()
print(result.summary())
```

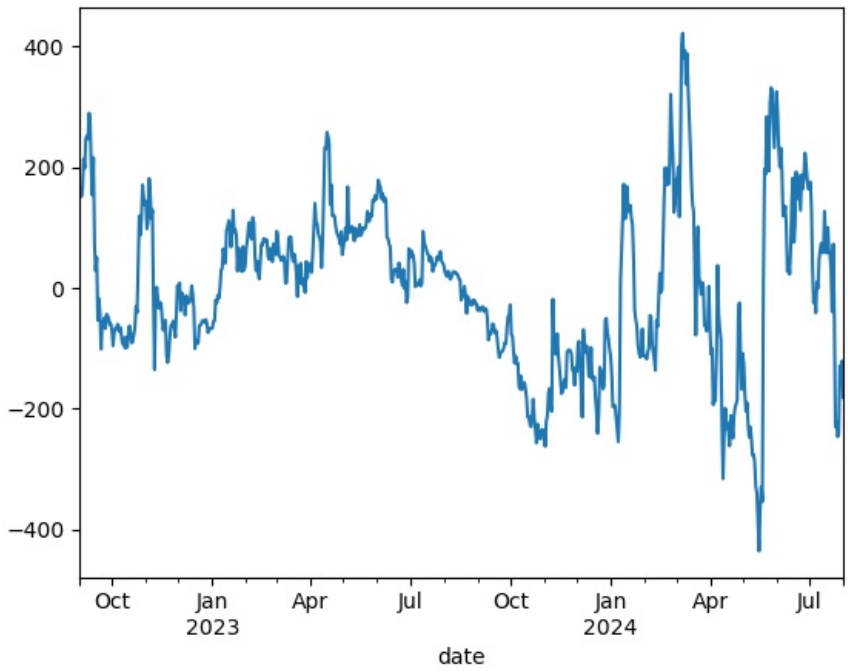
```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    ETH   R-squared:                       0.969
Model:                            OLS   Adj. R-squared:                  0.969
Method:                 Least Squares   F-statistic:                 2.218e+04
Date:                Thu, 15 Aug 2024   Prob (F-statistic):               0.00
Time:                        22:34:12   Log-Likelihood:                -4430.6
No. Observations:                 701   AIC:                             8865.
Df Residuals:                     699   BIC:                             8874.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        544.4281     11.830     46.021      0.000     521.202     567.655
BTC            0.0435      0.000    148.916      0.000       0.043       0.044
==============================================================================
Omnibus:                        0.795   Durbin-Watson:                   0.098
Prob(Omnibus):                  0.672   Jarque-Bera (JB):                0.636
Skew:                           0.026   Prob(JB):                        0.728
Kurtosis:                       3.138   Cond. No.                     9.43e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 9.43e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

In [15]:
```
#Obtain hedging numbers
hedge_ratio = result.params[1]
hedge_const = result.params[0]
print(f'hedge ratio: {hedge_ratio}, hedge constant: {hedge_const}')

#Obtain residuals
residuals = result.resid

#Augumented Dickey-Fuller Test to check for staionarity of the spread
adf_test = adfuller(residuals)
print(f'CADF:{adf_test}')
if(adf_test[1] < 0.01):
    print(f'the p-value of {adf_test[1]} is less than 1%')
    print('Proceed, stationary')
else:
    print('Stop! non-stationary')

residuals.plot()
```

```
hedge ratio: 0.043461421516468736, hedge constant: 544.4281125466536
CADF:(-4.663626608260299, 9.843851449718441e-05, 5, 695, {'1%': -3.439794053189972, '5%': -2.8657075899001314, '
10%': -2.56898934061384}, 6971.878254780801)
the p-value of 9.843851449718441e-05 is less than 1%
Proceed, stationary
```

```
/var/folders/cd/f8k7t0cj4ys82598qkc0bznr0000gn/T/ipykernel_7550/3978218189.py:2: FutureWarning: Series.__getitem
__ treating keys as positions is deprecated. In a future version, integer keys will always be treated as labels
(consistent with DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  hedge_ratio = result.params[1]
/var/folders/cd/f8k7t0cj4ys82598qkc0bznr0000gn/T/ipykernel_7550/3978218189.py:3: FutureWarning: Series.__getitem
__ treating keys as positions is deprecated. In a future version, integer keys will always be treated as labels
(consistent with DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  hedge_const = result.params[0]
```

Out[15]: <Axes: xlabel='date'>



In [16]:
```
#Fitting the spread to OU model
# e_t = C + Be_t-1 + error
OU_model = AutoReg(residuals, lags=1)
result = OU_model.fit()
print(result.summary())
```

```
                            AutoReg Model Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  701
Model:                     AutoReg(1)   Log Likelihood               -3604.348
Method:               Conditional MLE   S.D. of innovations             41.685
Date:                Thu, 15 Aug 2024   AIC                           7214.696
Time:                        22:34:20   BIC                           7228.349
Sample:                    09-02-2022   HQIC                          7219.973
                         - 08-01-2024
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.4847      1.576     -0.308      0.758      -3.573       2.603
y.L1           0.9509      0.012     81.118      0.000       0.928       0.974
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1            1.0516           +0.0000j            1.0516            0.0000
-----------------------------------------------------------------------------
```

```
/opt/anaconda3/lib/python3.12/site-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency in
formation was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
```

In [17]:
```
B = result.params['y.L1']
C = result.params['const']

# Compute OU parameters
tau = 1/365 # Since we are analyzing daily data for crypto (market opens 365 days a year)
theta = -np.log(B)/tau
half_life = np.log(2)/theta


mu = C/(1-B)
sigma_eq = result.resid.std() * np.sqrt(1/(1-np.exp(-2 * theta * tau)))

print("tau:",tau)
print("theta:", theta)
print("half_life:", half_life)
print("number of days to revert:", half_life/tau)
print("mean:", mu)
print("sigma_equalibrium:", sigma_eq)
```

```
tau: 0.0027397260273972603
theta: 18.376099878201146
half_life: 0.037720037720419604
number of days to revert: 13.767813767953156
mean: -9.871082407722996
sigma_equalibrium: 134.78160525377615
```

In [18]:
```
print(f'-------Conclusion------\nTrade ETH against BTC: hedge ratio = {hedge_ratio}, hedge constant ={hedge_cons
```

```
-------Conclusion------
Trade ETH against BTC: hedge ratio = 0.043461421516468736, hedge constant =544.4281125466536, residuals mean = -
9.871082407722996, sigma_eq = 134.78160525377615
```

In [ ]:
```

```

In [74]:
```
from openbb import obb

# Drop unnecessary columns from openbb and set columns to correctly match with the backtesting framework
def cleandata(df):
    if(df.columns.isin(['open','high','low','close','volume']).any()):
        df = df[['open','high','low','close','volume']]
    #     df.drop(["transactions","vwap"],axis=1,inplace=True, errors='ignore')
        df.columns = ["Open", "High", "Low", "Close", "Volume"]
    return df
```

In [75]:
```
#Getting historical daily price data for both btc and eth
df = obb.crypto.price.historical(["BTC-USD","ETH-USD"],
                                 start_date='2022-09-01',
                                 end_date = '2024-08-01').to_df()
```

In [77]:
```
# split by crypto
df_BTC = df[df["symbol"]=="BTCUSD"].drop("symbol", axis=1)
df_ETH = df[df["symbol"]=="ETHUSD"].drop("symbol", axis=1)

# Write to disk
# Improvements: Use of database
df_BTC = cleandata(df_BTC)
df_ETH = cleandata(df_ETH)

print(df_BTC)
print(df_ETH)
df_BTC.to_csv('data/BTC_daily.csv')
df_ETH.to_csv('data/ETH_daily.csv')
```

```
                Open      High       Low     Close        Volume
date
2022-09-01  20048.27  20205.53  19561.01  20133.65  3.018203e+10
2022-09-02  20132.67  20444.00  19755.00  19953.74  2.912400e+10
2022-09-03  19953.40  20053.90  19655.00  19835.47  2.361305e+10
2022-09-04  19833.59  20026.20  19588.27  20004.73  2.524586e+10
2022-09-05  20003.85  20060.00  19633.11  19794.58  2.881346e+10
...              ...       ...       ...       ...           ...
2024-07-28  67901.81  68308.82  67021.80  68244.30  2.392337e+08
2024-07-29  68244.30  70000.00  66400.01  66771.45  9.108617e+08
2024-07-30  66767.14  66994.15  65283.68  66169.68  5.778836e+08
2024-07-31  66169.68  66837.40  64500.00  64609.62  5.806063e+08
2024-08-01  64609.61  65593.56  62212.81  65288.18  1.020873e+09

[701 rows x 5 columns]
               Open     High      Low    Close        Volume
date
2022-09-01  1554.25  1599.51  1512.83  1586.23  1.643428e+10
2022-09-02  1586.10  1650.00  1546.20  1575.69  1.770848e+10
2022-09-03  1575.81  1582.69  1534.54  1557.70  9.516826e+09
2022-09-04  1557.55  1583.52  1540.59  1579.04  8.884145e+09
2022-09-05  1578.97  1631.44  1557.04  1618.01  1.306054e+10
...             ...      ...      ...      ...           ...
2024-07-28  3249.24  3283.98  3198.23  3269.98  9.972618e+07
2024-07-29  3270.15  3396.61  3256.68  3317.55  3.700714e+08
2024-07-30  3317.42  3365.68  3232.55  3278.27  2.832700e+08
2024-07-31  3278.42  3348.00  3213.00  3231.81  2.835073e+08
2024-08-01  3231.81  3242.14  3077.17  3200.54  3.686659e+08

[701 rows x 5 columns]
```

In [79]:
```
#Getting historical daily price data for both btc and eth
df_ETH_BTC = obb.crypto.price.historical("ETH-BTC",
                                 start_date='2022-09-01',
                                 end_date = '2024-08-01', provider='polygon').to_df()

df_ETH_BTC = cleandata(df_ETH_BTC)
display(df_ETH_BTC)
df_ETH_BTC.to_csv('data/ETH_BTC_daily.csv')
```

```
                Open      High       Low     Close        Volume
date
2022-09-01  0.07749  0.07923  0.07682  0.07879   8829.630282
2022-09-02  0.07877  0.08079  0.07810  0.07895   5352.616359
2022-09-03  0.07890  0.07895  0.07766  0.07855   2349.062462
2022-09-04  0.07846  0.07939  0.07836  0.07896   4563.738969
2022-09-05  0.07904  0.08205  0.07897  0.08172  20628.705177
...             ...      ...      ...      ...           ...
2024-07-28  0.04786  0.04838  0.04764  0.04796   3425.647761
2024-07-29  0.04792  0.04986  0.04766  0.04970   6898.227132
2024-07-30  0.04970  0.05036  0.04938  0.04952   4003.047977
2024-07-31  0.04955  0.05028  0.04940  0.04999   3476.845473
2024-08-01  0.05000  0.05013  0.04875  0.04902   6347.414128

[701 rows x 5 columns]
```

In [80]:
```
# Please make sure polygon_api_key is stored in your ~/openbb_platform/user_settings.json
# Otherwise remove the provider input
def get_crypto_historical_from_polygon(ticker):
    df = obb.crypto.price.historical(ticker,
                                     start_date='2024-05-01',
                                     end_date='2024-08-01',
                                     interval='15m',
                                     provider='polygon').to_df()
    df = cleandata(df)
    return df
```

In [81]:
```
btc_15m = get_crypto_historical_from_polygon('BTC-USD')
print(btc_15m)
```

```
                               Open     High       Low     Close      Volume
date
2024-05-01 00:00:00+00:00  60610.00  60991.0  60541.35  60541.35  160.993979
2024-05-01 00:15:00+00:00  60743.00  60875.0  60322.44  60370.84  116.953860
2024-05-01 00:30:00+00:00  60367.51  60581.0  60015.99  60291.27  192.209682
2024-05-01 00:45:00+00:00  60294.50  60550.0  60115.63  60173.03  195.513673
2024-05-01 01:00:00+00:00  60175.44  60475.0  59803.38  59828.94  261.251889
...                             ...      ...       ...       ...         ...
2024-08-01 22:45:00+00:00  65323.12  65516.0  65076.70  65116.22  316.639894
2024-08-01 23:00:00+00:00  65116.60  65346.0  64920.93  65030.00  277.887537
2024-08-01 23:15:00+00:00  65030.00  65194.0  64934.98  65041.07   98.133052
2024-08-01 23:30:00+00:00  65039.83  65365.0  64962.60  65211.00   79.573882
2024-08-01 23:45:00+00:00  65211.00  65549.0  65170.25  65288.18   97.306846

[8928 rows x 5 columns]
                               Open     High       Low     Close      Volume
date
2024-05-01 00:00:00+00:00  60610.00  60991.0  60541.35  60541.35  160.993979
2024-05-01 00:15:00+00:00  60743.00  60875.0  60322.44  60370.84  116.953860
2024-05-01 00:30:00+00:00  60367.51  60581.0  60015.99  60291.27  192.209682
2024-05-01 00:45:00+00:00  60294.50  60550.0  60115.63  60173.03  195.513673
2024-05-01 01:00:00+00:00  60175.44  60475.0  59803.38  59828.94  261.251889
...                             ...      ...       ...       ...         ...
2024-08-01 22:45:00+00:00  65323.12  65516.0  65076.70  65116.22  316.639894
2024-08-01 23:00:00+00:00  65116.60  65346.0  64920.93  65030.00  277.887537
2024-08-01 23:15:00+00:00  65030.00  65194.0  64934.98  65041.07   98.133052
2024-08-01 23:30:00+00:00  65039.83  65365.0  64962.60  65211.00   79.573882
2024-08-01 23:45:00+00:00  65211.00  65549.0  65170.25  65288.18   97.306846

[8928 rows x 5 columns]
```

In [84]:
```
sol_15m = get_crypto_historical_from_polygon('SOL-USD')
print(sol_15m)
```

```
                             Open    High     Low   Close        Volume
date
2024-05-01 00:00:00+00:00  126.63  127.54  126.43  126.43   9480.350219
2024-05-01 00:15:00+00:00  126.47  127.36  125.68  125.92  18365.765105
2024-05-01 00:30:00+00:00  125.92  126.13  124.44  125.00  31545.380961
2024-05-01 00:45:00+00:00  124.99  125.88  124.75  125.04   8946.268212
2024-05-01 01:00:00+00:00  125.02  125.78  124.38  124.55  12497.727483
...                           ...     ...     ...     ...           ...
2024-08-01 22:45:00+00:00  167.59  168.13  166.95  167.31   8411.162567
2024-08-01 23:00:00+00:00  167.31  168.13  166.97  167.26   8852.029155
2024-08-01 23:15:00+00:00  167.27  167.56  166.82  167.11   8739.188324
2024-08-01 23:30:00+00:00  167.11  168.28  166.80  167.90   7170.402839
2024-08-01 23:45:00+00:00  167.89  168.44  167.21  167.22   9141.329439

[8928 rows x 5 columns]
```

In [85]:
```
btc_15m.to_csv('data/BTC_15m.csv')
sol_15m.to_csv('data/SOL_15m.csv')
```

In [2]:
```
from backtesting import Backtest, Strategy
from backtesting.lib import crossover
import pandas as pd
import numpy as np
```

```
/opt/anaconda3/lib/python3.12/site-packages/backtesting/_plotting.py:50: UserWarning: Jupyter Notebook detected.
Setting Bokeh output to notebook. This may not work in Jupyter clients without JavaScript support (e.g. PyCharm,
Spyder IDE). Reset with `backtesting.set_bokeh_output(notebook=False)`.
  warnings.warn('Jupyter Notebook detected. '
```

Loading BokehJS ...

In [3]:
```
#Trend-Following Strategy
#Defining indicators
def EMA(array, n=20):
    """
    Expotential Moving Average
    from n previous periods
    """
    return pd.Series(array).ewm(span=n, adjust=False).mean()

def MACD(array):
    """MACD Indicator"""
    EMA12 = pd.Series(array).ewm(span=12, adjust=False).mean()
    EMA26 = pd.Series(array).ewm(span=26, adjust=False).mean()
    macd = EMA12 - EMA26
    signal = macd.ewm(span=9, adjust=False).mean()
    return macd, signal

def ADX(timeSeriesData, n=14):
    """
    Find ADX of time series data
    Assuming input timeSeriesData is a pandas dataframe containing columns High, Low, Close
    with number of periods default to 14
    """

    if (not ('High' in timeSeriesData.columns)
        and ('Low' in timeSeriesData.columns)
        and ('Close' in timeSeriesData.columns)):
        raise ValueError("Input does not contain necessary data")

    #Find Average True Range
    df = timeSeriesData.copy()
    df['TR'] = np.maximum(df['High'] - df['Low'],
                          np.abs(df['High'] - df['Close'].shift(1)),
                          np.abs(df['Low'] - df['Close'].shift(1)))

    df['ATR'] = df['TR'].ewm(alpha=1/n, adjust=False).mean()

    #Find Directional Movement
    df['H-pH'] = df['High'] - df['High'].shift(1)
    df['pL-L'] = df['Low'].shift(1) - df['Low']

    df['+DM'] = np.where((df['H-pH'] > df['pL-L']) & (df['H-pH'] > 0), df['H-pH'], 0.0)
    df['-DM'] = np.where((df['pL-L'] > df['H-pH']) & (df['pL-L'] > 0), df['pL-L'], 0.0)

    # #Find Directional Index
    df['+DI'] = df['+DM'].ewm(alpha=1/n, adjust=False).mean()/df['ATR'] * 100
    df['-DI'] = df['-DM'].ewm(alpha=1/n, adjust=False).mean()/df['ATR'] * 100

    # #Find ADX
    df['DX'] = np.abs(df['+DI'] - df['-DI'])/(df['+DI'] + df['-DI']) * 100
    df['ADX'] = df['DX'].ewm(alpha=1/n, adjust=False).mean()

    return df['ADX']

class TrendFollowing(Strategy):

    #Parameters need to be optimized
    adx_threshold = 25
    trade_size = 0.98
    tp_ratio = 0.05
    sl_ratio = 0.02

    def init(self):
        self.macd, self.macd_sig = self.I(MACD, self.data.Close)
        self.adx = self.I(ADX, self.data.df)
        self.ema30 = self.I(EMA, self.data.Close, 30)

    def next(self):
        price = self.data.Close[-1]
```

```
        if (self.macd > 0 and
            crossover(self.macd, self.macd_sig) and
            self.adx > self.adx_threshold):

            #Close any short position and buy
            if self.position.is_short:
                self.position.close()
            self.buy(tp=(1+self.tp_ratio)*price, sl=(1-self.sl_ratio)*price, size=self.trade_size)

        elif (self.macd < 0 and
              crossover(self.macd_sig, self.macd) and
              self.adx > self.adx_threshold):

            #Close any long position and sell
            if self.position.is_long:
                self.position.close()
            self.sell(tp=(1-self.tp_ratio)*price, sl=(1+self.sl_ratio)*price, size=self.trade_size)

        #Closing position logic
        if self.position.is_long:
            if (price < 0.99 * self.ema30[-1] and
                self.macd < 0):
                self.position.close()
        if self.position.is_short:
            if (price > 1.01 * self.ema30[-1] and
                self.macd > 0):
                self.position.close()
```

In [3]:
```
#Backtest on BTC 15 minutes data for the past 3 month
symbol = "BTC"
interval = "15m"
df = pd.read_csv(f'data/{symbol}_{interval}.csv', index_col=0, parse_dates=True)
display(df)
```

date	Open	High	Low	Close	Volume
2024-05-01 00:00:00+00:00	60610.00	60991.0	60541.35	60541.35	160.993979
2024-05-01 00:15:00+00:00	60743.00	60875.0	60322.44	60370.84	116.953860
2024-05-01 00:30:00+00:00	60367.51	60581.0	60015.99	60291.27	192.209682
2024-05-01 00:45:00+00:00	60294.50	60550.0	60115.63	60173.03	195.513673
2024-05-01 01:00:00+00:00	60175.44	60475.0	59803.38	59828.94	261.251889
...	...	...	...	...	...
2024-08-01 22:45:00+00:00	65323.12	65516.0	65076.70	65116.22	316.639894
2024-08-01 23:00:00+00:00	65116.60	65346.0	64920.93	65030.00	277.887537
2024-08-01 23:15:00+00:00	65030.00	65194.0	64934.98	65041.07	98.133052
2024-08-01 23:30:00+00:00	65039.83	65365.0	64962.60	65211.00	79.573882
2024-08-01 23:45:00+00:00	65211.00	65549.0	65170.25	65288.18	97.306846

8928 rows × 5 columns

In [129…
```
bt = Backtest(df, TrendFollowing, cash=1000000, commission=0.001)
stats = bt.optimize(
    method='grid',
    adx_threshold = range(25, 40, 3),
    trade_size = [0.33, 0.49, 0.99],
    tp_ratio = [0.04, 0.05, 0.06, 0.1],
    sl_ratio = [0.01, 0.02, 0.03, 0.1],
    maximize='Return [%]'
)
print(stats)
print(stats._strategy)
print(stats._trades)
bt.plot(filename=f'backtest_results/{symbol}_{interval}_plot.html')
```

```
/opt/anaconda3/lib/python3.12/site-packages/backtesting/backtesting.py:1375: UserWarning: For multiprocessing su
pport in `Backtest.optimize()` set multiprocessing start method to 'fork'.
  warnings.warn("For multiprocessing support in `Backtest.optimize()` "
  0%|          | 0/8 [00:00<?, ?it/s]
Start                     2024-05-01 00:00...
End                       2024-08-01 23:45...
Duration                     92 days 23:45:00
Exposure Time [%]                   23.790323
```

```
Equity Final [$]                1232502.45673
Equity Peak [$]                 1232502.45673
Return [%]                         23.250246
Buy & Hold Return [%]               7.840641
Return (Ann.) [%]                 111.728514
Volatility (Ann.) [%]              59.85909
Sharpe Ratio                        1.866525
Sortino Ratio                      11.496677
Calmar Ratio                       15.250826
Max. Drawdown [%]                  -7.326063
Avg. Drawdown [%]                  -1.519498
Max. Drawdown Duration       44 days 02:30:00
Avg. Drawdown Duration        3 days 07:28:00
# Trades                                  34
Win Rate [%]                       29.411765
Best Trade [%]                      6.105556
Worst Trade [%]                    -1.153549
Avg. Trade [%]                      0.410625
Max. Trade Duration           3 days 09:30:00
Avg. Trade Duration           0 days 16:56:00
Profit Factor                       1.572669
Expectancy [%]                      0.445432
SQN                                 1.426755
_strategy                 TrendFollowing(a...
_equity_curve                             ...
_trades                      Size  EntryB...
dtype: object
TrendFollowing(adx_threshold=34,trade_size=0.99,tp_ratio=0.06,sl_ratio=0.01)
    Size  EntryBar  ExitBar   EntryPrice   ExitPrice          PnL  ReturnPct  \
0    -16        25       75  59792.76738  58662.9000  18077.87808   0.018896
1     16       274      532  62944.88200  63581.2000  10181.08800   0.010109
2    -16       665      830  62761.07610  62169.1600   9470.65760   0.009431
3    -16      1273     1306  61669.63863  62341.1087 -10743.52112  -0.010888
4      1      1420     1507  65874.80900  65142.0000   -732.80900  -0.011124
5     15      1395     1508  63732.21855  65117.5000  20779.22175   0.021736
6     15      1578     1904  66497.75132  70417.1992  58791.71820   0.058941
7     15      1918     1925  71495.73431  70710.0669 -11785.01115  -0.010989
8    -16      2185     2194  67023.56934  67766.3237 -11884.06976  -0.011082
9    -15      2748     2801  67516.01640  68258.0321 -11130.23550  -0.010990
10    -1      3982     4015  66797.20593  67532.7208   -735.51487  -0.011011
11   -15      3976     4071  66976.17678  67748.7800 -11589.04830  -0.011535
12   -16      4668     4693  64551.18420  65259.3724 -11331.01120  -0.010971
13    15      4844     4853  66260.49430  65523.1500 -11060.16450  -0.011128
14   -16      4943     4955  63640.69560  64320.0219 -10869.22080  -0.010674
15    -1      5223     5239  60979.71924  61666.2570   -686.53776  -0.011258
16   -16      5196     5263  62916.02100  59074.6478  61461.97120   0.061056
17   -17      5263     5278  59753.67651  60423.7045 -11390.47583  -0.011213
18    -1      5251     5292  60452.76672  61117.6452   -664.87848  -0.010998
19    16      5850     6009  62904.34150  62212.9860 -11061.68800  -0.010991
20    -1      6100     6102  59867.02305  60527.9769   -660.95385  -0.011040
21   -17      6037     6152  61755.52266  58106.6640  62030.59722   0.059086
22    -1      6198     6199  56841.90120  57465.0711   -623.16990  -0.010963
23   -19      6175     6257  58383.82773  54949.3920  65254.27887   0.058825
24    20      6430     6450  58435.93756  57794.2101 -12834.54920  -0.010982
25   -20      6797     6809  57319.54308  57950.6993 -12623.12440  -0.011011
26   -20      6918     6921  56556.70668  57174.1608 -12349.08240  -0.010917
27    18      7204     7294  61311.35010  64914.3682  64854.32580   0.058766
28    17      7663     7681  67293.41619  66553.9380 -12571.12923  -0.010989
29   -18      8165     8226  64491.49395  65193.3891 -12634.11270  -0.010884
30    17      8311     8314  67650.34276  66912.5556 -12542.38172  -0.010906
31    16      8570     8599  69873.51371  69111.8406 -12186.76976  -0.010901
32    -1      8841     8879  64097.40843  64803.1857   -705.77727  -0.011011
33   -17      8639     8894  66728.77443  62787.8358  66995.95671   0.059059

                  EntryTime                  ExitTime        Duration
0  2024-05-01 06:15:00+00:00 2024-05-01 18:45:00+00:00 0 days 12:30:00
1  2024-05-03 20:30:00+00:00 2024-05-06 13:00:00+00:00 2 days 16:30:00
2  2024-05-07 22:15:00+00:00 2024-05-09 15:30:00+00:00 1 days 17:15:00
3  2024-05-14 06:15:00+00:00 2024-05-14 14:30:00+00:00 0 days 08:15:00
4  2024-05-15 19:00:00+00:00 2024-05-16 16:45:00+00:00 0 days 21:45:00
5  2024-05-15 12:45:00+00:00 2024-05-16 17:00:00+00:00 1 days 04:15:00
6  2024-05-17 10:30:00+00:00 2024-05-20 20:00:00+00:00 3 days 09:30:00
7  2024-05-20 23:30:00+00:00 2024-05-21 01:15:00+00:00 0 days 01:45:00
8  2024-05-23 18:15:00+00:00 2024-05-23 20:30:00+00:00 0 days 02:15:00
9  2024-05-29 15:00:00+00:00 2024-05-30 04:15:00+00:00 0 days 13:15:00
10 2024-06-11 11:30:00+00:00 2024-06-11 19:45:00+00:00 0 days 08:15:00
11 2024-06-11 10:00:00+00:00 2024-06-12 09:45:00+00:00 0 days 23:45:00
12 2024-06-18 15:00:00+00:00 2024-06-18 21:15:00+00:00 0 days 06:15:00
13 2024-06-20 11:00:00+00:00 2024-06-20 13:15:00+00:00 0 days 02:15:00
14 2024-06-21 11:45:00+00:00 2024-06-21 14:45:00+00:00 0 days 03:00:00
15 2024-06-24 09:45:00+00:00 2024-06-24 13:45:00+00:00 0 days 04:00:00
16 2024-06-24 03:00:00+00:00 2024-06-24 19:45:00+00:00 0 days 16:45:00
17 2024-06-24 19:45:00+00:00 2024-06-24 23:30:00+00:00 0 days 03:45:00
```

```
18 2024-06-24 16:45:00+00:00 2024-06-25 03:00:00+00:00 0 days 10:15:00
19 2024-06-30 22:30:00+00:00 2024-07-02 14:15:00+00:00 1 days 15:45:00
20 2024-07-03 13:00:00+00:00 2024-07-03 13:30:00+00:00 0 days 00:30:00
21 2024-07-02 21:15:00+00:00 2024-07-04 02:00:00+00:00 1 days 04:45:00
22 2024-07-04 13:30:00+00:00 2024-07-04 13:45:00+00:00 0 days 00:15:00
23 2024-07-04 07:45:00+00:00 2024-07-05 04:15:00+00:00 0 days 20:30:00
24 2024-07-06 23:30:00+00:00 2024-07-07 04:30:00+00:00 0 days 05:00:00
25 2024-07-10 19:15:00+00:00 2024-07-10 22:15:00+00:00 0 days 03:00:00
26 2024-07-12 01:30:00+00:00 2024-07-12 02:15:00+00:00 0 days 00:45:00
27 2024-07-15 01:00:00+00:00 2024-07-15 23:30:00+00:00 0 days 22:30:00
28 2024-07-19 19:45:00+00:00 2024-07-20 00:15:00+00:00 0 days 04:30:00
29 2024-07-25 01:15:00+00:00 2024-07-25 16:30:00+00:00 0 days 15:15:00
30 2024-07-26 13:45:00+00:00 2024-07-26 14:30:00+00:00 0 days 00:45:00
31 2024-07-29 06:30:00+00:00 2024-07-29 13:45:00+00:00 0 days 07:15:00
32 2024-08-01 02:15:00+00:00 2024-08-01 11:45:00+00:00 0 days 09:30:00
33 2024-07-29 23:45:00+00:00 2024-08-01 15:30:00+00:00 2 days 15:45:00
```

```
BokehDeprecationWarning: Passing lists of formats for DatetimeTickFormatter scales was deprecated in Bokeh 3.0. Configure a single string format for each scale
/opt/anaconda3/lib/python3.12/site-packages/backtesting/_plotting.py:250: UserWarning: DatetimeFormatter scales now only accept a single format. Using the first provided: '%d %b'
  formatter=DatetimeTickFormatter(days=['%d %b', '%a %d'],
BokehDeprecationWarning: Passing lists of formats for DatetimeTickFormatter scales was deprecated in Bokeh 3.0. Configure a single string format for each scale
/opt/anaconda3/lib/python3.12/site-packages/backtesting/_plotting.py:250: UserWarning: DatetimeFormatter scales now only accept a single format. Using the first provided: '%m/%Y'
  formatter=DatetimeTickFormatter(days=['%d %b', '%a %d'],
/opt/anaconda3/lib/python3.12/site-packages/backtesting/_plotting.py:456: FutureWarning: 'H' is deprecated and will be removed in a future version, please use 'h' instead.
  .resample(resample_rule, label='left')
/opt/anaconda3/lib/python3.12/site-packages/backtesting/_plotting.py:659: UserWarning: found multiple competing values for 'toolbar.active_drag' property; using the latest value
  fig = gridplot(
/opt/anaconda3/lib/python3.12/site-packages/backtesting/_plotting.py:659: UserWarning: found multiple competing values for 'toolbar.active_scroll' property; using the latest value
  fig = gridplot(
```

Out[129… **GridPlot**(id = 'p19653', …)

In [4]:
```python
#Backtest on SOL 15 minutes data for the past 3 month
symbol = "SOL"
interval = "15m"
df = pd.read_csv(f'data/{symbol}_{interval}.csv', index_col=0, parse_dates=True)
display(df)
```

date	Open	High	Low	Close	Volume
2024-05-01 00:00:00+00:00	126.63	127.54	126.43	126.43	9480.350219
2024-05-01 00:15:00+00:00	126.47	127.36	125.68	125.92	18365.765105
2024-05-01 00:30:00+00:00	125.92	126.13	124.44	125.00	31545.380961
2024-05-01 00:45:00+00:00	124.99	125.88	124.75	125.04	8946.268212
2024-05-01 01:00:00+00:00	125.02	125.78	124.38	124.55	12497.727483
…	…	…	…	…	…
2024-08-01 22:45:00+00:00	167.59	168.13	166.95	167.31	8411.162567
2024-08-01 23:00:00+00:00	167.31	168.13	166.97	167.26	8852.029155
2024-08-01 23:15:00+00:00	167.27	167.56	166.82	167.11	8739.188324
2024-08-01 23:30:00+00:00	167.11	168.28	166.80	167.90	7170.402839
2024-08-01 23:45:00+00:00	167.89	168.44	167.21	167.22	9141.329439

8928 rows × 5 columns

In [131…
```python
bt = Backtest(df, TrendFollowing, cash=1000000, commission=0.001)
stats = bt.optimize(
    method='grid',
    adx_threshold = range(25, 40, 3),
    trade_size = [0.33, 0.49, 0.99],
    tp_ratio = [0.04, 0.05, 0.06, 0.1],
    sl_ratio = [0.01, 0.02, 0.03, 0.1],
    maximize='Return [%]'
)
print(stats)
print(stats._strategy)
print(stats._trades)
bt.plot(filename=f'backtest_results/{symbol}_{interval}_plot.html')
```

```
/opt/anaconda3/lib/python3.12/site-packages/backtesting/backtesting.py:1375: UserWarning: For multiprocessing su
pport in `Backtest.optimize()` set multiprocessing start method to 'fork'.
  warnings.warn("For multiprocessing support in `Backtest.optimize()` "
  0%|          | 0/8 [00:00<?, ?it/s]
Start                     2024-05-01 00:00...
End                       2024-08-01 23:45...
Duration                     92 days 23:45:00
Exposure Time [%]                   29.569892
Equity Final [$]                 1387493.16089
Equity Peak [$]                  1387493.16089
Return [%]                          38.749316
Buy & Hold Return [%]               32.262912
Return (Ann.) [%]                  211.502457
Volatility (Ann.) [%]              137.262595
Sharpe Ratio                         1.54086
Sortino Ratio                       10.081473
Calmar Ratio                        18.371026
Max. Drawdown [%]                  -11.512828
Avg. Drawdown [%]                   -1.882531
Max. Drawdown Duration       15 days 16:00:00
Avg. Drawdown Duration        1 days 18:24:00
# Trades                                   83
Win Rate [%]                        48.192771
Best Trade [%]                       4.251718
Worst Trade [%]                     -3.170304
Avg. Trade [%]                       0.660425
Max. Trade Duration           1 days 20:45:00
Avg. Trade Duration           0 days 12:09:00
Profit Factor                        1.676647
Expectancy [%]                       0.703949
SQN                                  1.515604
_strategy                 TrendFollowing(a...
_equity_curve                            ...
_trades                      Size  EntryB...
dtype: object
TrendFollowing(adx_threshold=25,trade_size=0.99,tp_ratio=0.04,sl_ratio=0.03)
    Size  EntryBar  ExitBar  EntryPrice  ExitPrice          PnL  ReturnPct  \
0  -7990         9       29   123.89598   119.0496  38722.57620   0.039117
1   7342       157      223   140.04991   137.3100 -20116.41922  -0.019564
2   -164       710      731   145.39446   148.3100   -478.14856  -0.020053
3   -207       692      731   147.18267   148.3100   -233.35731  -0.007659
4   -180       668      731   149.84001   148.3100    275.40180   0.010211
..   ...       ...      ...         ...        ...          ...        ...
78 -7309      8433     8436   179.64018   185.1940 -40592.87038  -0.030916
79 -6922      8631     8712   183.86595   176.6880  49685.76990   0.039039
80  -292      8648     8744   180.18963   181.3800   -347.58804  -0.006606
81   -99      8848     8894   168.47136   161.7792    662.52384   0.039723
82 -7833      8845     8894   168.72111   162.1056  51819.28983   0.039210

                  EntryTime                  ExitTime        Duration
0  2024-05-01 02:15:00+00:00 2024-05-01 07:15:00+00:00 0 days 05:00:00
1  2024-05-02 15:15:00+00:00 2024-05-03 07:45:00+00:00 0 days 16:30:00
2  2024-05-08 09:30:00+00:00 2024-05-08 14:45:00+00:00 0 days 05:15:00
3  2024-05-08 05:00:00+00:00 2024-05-08 14:45:00+00:00 0 days 09:45:00
4  2024-05-07 23:00:00+00:00 2024-05-08 14:45:00+00:00 0 days 15:45:00
..                       ...                       ...             ...
78 2024-07-27 20:15:00+00:00 2024-07-27 21:00:00+00:00 0 days 00:45:00
79 2024-07-29 21:45:00+00:00 2024-07-30 18:00:00+00:00 0 days 20:15:00
80 2024-07-30 02:00:00+00:00 2024-07-31 02:00:00+00:00 1 days 00:00:00
81 2024-08-01 04:00:00+00:00 2024-08-01 15:30:00+00:00 0 days 11:30:00
82 2024-08-01 03:15:00+00:00 2024-08-01 15:30:00+00:00 0 days 12:15:00

[83 rows x 10 columns]
BokehDeprecationWarning: Passing lists of formats for DatetimeTickFormatter scales was deprecated in Bokeh 3.0.
Configure a single string format for each scale
/opt/anaconda3/lib/python3.12/site-packages/backtesting/_plotting.py:250: UserWarning: DatetimeFormatter scales
now only accept a single format. Using the first provided: '%d %b'
  formatter=DatetimeTickFormatter(days=['%d %b', '%a %d'],
BokehDeprecationWarning: Passing lists of formats for DatetimeTickFormatter scales was deprecated in Bokeh 3.0.
Configure a single string format for each scale
/opt/anaconda3/lib/python3.12/site-packages/backtesting/_plotting.py:250: UserWarning: DatetimeFormatter scales
now only accept a single format. Using the first provided: '%m/%Y'
  formatter=DatetimeTickFormatter(days=['%d %b', '%a %d'],
/opt/anaconda3/lib/python3.12/site-packages/backtesting/_plotting.py:456: FutureWarning: 'H' is deprecated and w
ill be removed in a future version, please use 'h' instead.
  .resample(resample_rule, label='left')
/opt/anaconda3/lib/python3.12/site-packages/backtesting/_plotting.py:659: UserWarning: found multiple competing
values for 'toolbar.active_drag' property; using the latest value
  fig = gridplot(
/opt/anaconda3/lib/python3.12/site-packages/backtesting/_plotting.py:659: UserWarning: found multiple competing
values for 'toolbar.active_scroll' property; using the latest value
  fig = gridplot(
```

Out[131… **GridPlot**(id = 'p20103', …)

In [6]:
```
#Mean Rversion Strategy

def resid(df1, df2):
    hedge_ratio = 0.04346
    hedge_const = 544.50
    #Values are obtained directly from engle-granger procedure
    #See engle-granger.ipynb
    #Trade ETH against BTC: hedge ratio = 0.04346, hedge const = 544.50,

    return df1 - (df2 * hedge_ratio + hedge_const)

class MeanReversion(Strategy):

    Z_entry = 0.5
    Z_exit = 0.1

    def init(self):
        self.resid = self.I(resid, self.data.ETH, self.data.BTC)

    def next(self):
        sigma_eq = 134.79
        mu = -9.91
        #Values are obtained directly from engle-granger procedure
        #See engle-granger.ipynb
        #residuals mean = -9.91, sigma_eq = 134.79

        if not self.position:
            if self.resid > mu + self.Z_entry * sigma_eq:
                self.sell()
            elif self.resid < mu - self.Z_entry * sigma_eq:
                self.buy()
        elif (self.resid < mu + self.Z_exit * sigma_eq and
            self.resid > mu - self.Z_exit * sigma_eq):
            self.position.close()
```

In [7]:
```
#Mean Reversion backtesting
interval = "daily"
btc_daily = pd.read_csv(f'data/BTC_{interval}.csv', index_col=0, parse_dates=True)
eth_daily = pd.read_csv(f'data/ETH_{interval}.csv', index_col=0, parse_dates=True)
eth_btc_daily= pd.read_csv(f'data/ETH_BTC_{interval}.csv', index_col=0, parse_dates=True)

#ETH_BTC is used for trading purpose
#ETH and BTC appended individually to the dataframe to caluclate residules for signal generation
df = eth_btc_daily.assign(ETH=eth_daily['Close'], BTC=btc_daily['Close'])

display(df.head())

bt = Backtest(df, MeanReversion, cash=10000, commission=0.001, exclusive_orders=True)
stats = bt.optimize(
    method='grid',
    Z_entry = [x / 100.0 for x in range(30, 150, 10)],
    Z_exit = [x / 100.0 for x in range(1, 30, 3)],
    maximize='Return [%]'
)
print(stats)
print(stats._trades)
print(stats._strategy)
bt.plot(filename=f'backtest_results/mean_reversion_plot.html')
```

date	Open	High	Low	Close	Volume	ETH	BTC
2022-09-01	0.07749	0.07923	0.07682	0.07879	8829.630282	1586.23	20133.65
2022-09-02	0.07877	0.08079	0.07810	0.07895	5352.616359	1575.69	19953.74
2022-09-03	0.07890	0.07895	0.07766	0.07855	2349.062462	1557.70	19835.47
2022-09-04	0.07846	0.07939	0.07836	0.07896	4563.738969	1579.04	20004.73
2022-09-05	0.07904	0.08205	0.07897	0.08172	20628.705177	1618.01	19794.58

```
/opt/anaconda3/lib/python3.12/site-packages/backtesting/backtesting.py:1375: UserWarning: For multiprocessing su
pport in `Backtest.optimize()` set multiprocessing start method to 'fork'.
  warnings.warn("For multiprocessing support in `Backtest.optimize()` "
  0%|          | 0/8 [00:00<?, ?it/s]
Start                     2022-09-01 00:00:00
End                       2024-08-01 00:00:00
Duration                    700 days 00:00:00
Exposure Time [%]                   72.610556
Equity Final [$]                27162.366044
```

```
Equity Peak [$]                  28213.571994
Return [%]                          171.62366
Buy & Hold Return [%]              -37.783983
Return (Ann.) [%]                   68.252027
Volatility (Ann.) [%]               48.265302
Sharpe Ratio                         1.414101
Sortino Ratio                          4.5005
Calmar Ratio                         5.211141
Max. Drawdown [%]                  -13.097329
Avg. Drawdown [%]                   -2.727702
Max. Drawdown Duration      171 days 00:00:00
Avg. Drawdown Duration       14 days 00:00:00
# Trades                                   28
Win Rate [%]                        92.857143
Best Trade [%]                      11.184875
Worst Trade [%]                     -2.600422
Avg. Trade [%]                       3.633192
Max. Trade Duration          71 days 00:00:00
Avg. Trade Duration          18 days 00:00:00
Profit Factor                       21.075947
Expectancy [%]                       3.688756
SQN                                  5.658952
_strategy                MeanReversion(Z_...
_equity_curve                            ...
_trades                        Size  Entr...
dtype: object
      Size  EntryBar  ExitBar  EntryPrice  ExitPrice          PnL  ReturnPct  \
0  -126869         2       19    0.078821    0.07049  1056.958326   0.105696
1   163740        21       26    0.067527    0.06950   322.983700   0.029211
2   169680        32       53    0.067067    0.06970   446.767440   0.039259
3  -157051        56       69    0.075305    0.07194   528.416936   0.044680
4   178363        70       71    0.069269    0.07382   811.694340   0.065697
5   187400        81       90    0.070260    0.07397   695.218394   0.052801
6   197408       107      126    0.070220    0.07456   856.721109   0.061803
7  -199289       136      146    0.073856    0.06876  1015.590694   0.069000
8  -225226       155      165    0.069860    0.06952    76.592606   0.004868
9  -229839       168      190    0.068791    0.07058  -411.149794  -0.026004
10 -214816       193      200    0.071688    0.06367  1722.446244   0.111849
11 -258123       216      287    0.066334    0.06570   163.546733   0.009552
12 -271463       316      325    0.063676    0.06266   275.876988   0.015960
13  293281       385      394    0.059880    0.06198   615.942891   0.035073
14  306436       399      435    0.059319    0.05781  -462.491597  -0.025443
15  320025       437      497    0.055355    0.05538     7.904618   0.000446
16 -300792       499      510    0.058921    0.05621   815.453128   0.046011
17  341633       513      524    0.054264    0.05507   275.284455   0.014849
18  351899       526      532    0.053463    0.05358    41.027904   0.002181
19 -331873       537      567    0.056813    0.05184  1650.447572   0.087535
20 -384882       568      569    0.053277    0.05231   372.053883   0.018144
21  416795       580      587    0.050090    0.05070   254.228278   0.012177
22  437793       590      606    0.048268    0.05169  1498.031332   0.070891
23  448728       607      628    0.050430    0.05126   372.273723   0.016451
24 -426149       629      650    0.053976    0.05194   867.626580   0.037720
25 -443372       654      674    0.053836    0.05269   508.153083   0.021289
26 -450720       681      691    0.054086    0.05092  1426.916419   0.058534
27 -488933       692      700    0.052777    0.05000  1357.850060   0.052621

    EntryTime   ExitTime Duration
0  2022-09-03 2022-09-20  17 days
1  2022-09-22 2022-09-27   5 days
2  2022-10-03 2022-10-24  21 days
3  2022-10-27 2022-11-09  13 days
4  2022-11-10 2022-11-11   1 days
5  2022-11-21 2022-11-30   9 days
6  2022-12-17 2023-01-05  19 days
7  2023-01-15 2023-01-25  10 days
8  2023-02-03 2023-02-13  10 days
9  2023-02-16 2023-03-10  22 days
10 2023-03-13 2023-03-20   7 days
11 2023-04-05 2023-06-15  71 days
12 2023-07-14 2023-07-23   9 days
13 2023-09-21 2023-09-30   9 days
14 2023-10-05 2023-11-10  36 days
15 2023-11-12 2024-01-11  60 days
16 2024-01-13 2024-01-24  11 days
17 2024-01-27 2024-02-07  11 days
18 2024-02-09 2024-02-15   6 days
19 2024-02-20 2024-03-21  30 days
20 2024-03-22 2024-03-23   1 days
21 2024-04-03 2024-04-10   7 days
22 2024-04-13 2024-04-29  16 days
23 2024-04-30 2024-05-21  21 days
24 2024-05-22 2024-06-12  21 days
25 2024-06-16 2024-07-06  20 days
```

```
26 2024-07-13 2024-07-23  10 days
27 2024-07-24 2024-08-01   8 days
MeanReversion(Z_entry=0.6,Z_exit=0.28)
```

```
BokehDeprecationWarning: Passing lists of formats for DatetimeTickFormatter scales was deprecated in Bokeh 3.0. Configure a single string format for each scale
/opt/anaconda3/lib/python3.12/site-packages/backtesting/_plotting.py:250: UserWarning: DatetimeFormatter scales now only accept a single format. Using the first provided: '%d %b'
  formatter=DatetimeTickFormatter(days=['%d %b', '%a %d'],
BokehDeprecationWarning: Passing lists of formats for DatetimeTickFormatter scales was deprecated in Bokeh 3.0. Configure a single string format for each scale
/opt/anaconda3/lib/python3.12/site-packages/backtesting/_plotting.py:250: UserWarning: DatetimeFormatter scales now only accept a single format. Using the first provided: '%m/%Y'
  formatter=DatetimeTickFormatter(days=['%d %b', '%a %d'],
/opt/anaconda3/lib/python3.12/site-packages/backtesting/_plotting.py:456: FutureWarning: 'M' is deprecated and will be removed in a future version, please use 'ME' instead.
  .resample(resample_rule, label='left')
/opt/anaconda3/lib/python3.12/site-packages/backtesting/_plotting.py:659: UserWarning: found multiple competing values for 'toolbar.active_drag' property; using the latest value
  fig = gridplot(
/opt/anaconda3/lib/python3.12/site-packages/backtesting/_plotting.py:659: UserWarning: found multiple competing values for 'toolbar.active_scroll' property; using the latest value
  fig = gridplot(
```

Out[7]: **GridPlot**(id = 'p1351', …)

In [ ]: