In [1]:
```
import yfinance as yf
import datetime
import numpy as np
import matplotlib.pyplot as plt
import hvplot.pandas
import pandas as pd
import quantstats as qs
import talib as ta
```

In [2]:
```
df = yf.download("BTC-USD")
```

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

In [3]:
```
df.tail(15)
```

Out[3]:

| Date | Open | High | Low | Close | Adj Close | Volume |
|---|---|---|---|---|---|---|
| **2022-05-27** | 29251.140625 | 29346.943359 | 28326.613281 | 28627.574219 | 28627.574219 | 36582005748 |
| **2022-05-28** | 28622.625000 | 28814.900391 | 28554.566406 | 28814.900391 | 28814.900391 | 35519577634 |
| **2022-05-29** | 29019.867188 | 29498.009766 | 28841.107422 | 29445.957031 | 29445.957031 | 18093886409 |
| **2022-05-30** | 29443.365234 | 31949.630859 | 29303.572266 | 31726.390625 | 31726.390625 | 39277993274 |
| **2022-05-31** | 31723.865234 | 32249.863281 | 31286.154297 | 31792.310547 | 31792.310547 | 33538210634 |
| **2022-06-01** | 31792.554688 | 31957.285156 | 29501.587891 | 29799.080078 | 29799.080078 | 41135817341 |
| **2022-06-02** | 29794.890625 | 30604.734375 | 29652.705078 | 30467.488281 | 30467.488281 | 29083562061 |
| **2022-06-03** | 30467.806641 | 30633.035156 | 29375.689453 | 29704.390625 | 29704.390625 | 26175547452 |
| **2022-06-04** | 29706.138672 | 29930.564453 | 29500.005859 | 29832.914062 | 29832.914062 | 16588370958 |
| **2022-06-05** | 29835.117188 | 30117.744141 | 29574.449219 | 29906.662109 | 29906.662109 | 17264085441 |
| **2022-06-06** | 29910.283203 | 31693.291016 | 29894.187500 | 31370.671875 | 31370.671875 | 31947336829 |

| Date | Open | High | Low | Close | Adj Close | Volume |
|---|---|---|---|---|---|---|
| **2022-06-07** | 31371.742188 | 31489.683594 | 29311.683594 | 31155.478516 | 31155.478516 | 40770974039 |
| **2022-06-08** | 31151.480469 | 31253.691406 | 29944.404297 | 30214.355469 | 30214.355469 | 30242059107 |
| **2022-06-09** | 30215.279297 | 30609.310547 | 30020.265625 | 30111.998047 | 30111.998047 | 21692004719 |
| **2022-06-10** | 30086.400391 | 30105.738281 | 29732.906250 | 29994.623047 | 29994.623047 | 23435472896 |

In [4]:
```
# Count nulls
df.isna().sum()
```

Out[4]:
```
Open         0
High         0
Low          0
Close        0
Adj Close    0
Volume       0
dtype: int64
```

In [5]:
```
# Calculate VWAP
df["VWAP"] = (df.Volume*(df.Close)).cumsum() / df.Volume.cumsum()
```

In [6]:
```
df.head()
```

Out[6]:

| Date | Open | High | Low | Close | Adj Close | Volume | VWAP |
|---|---|---|---|---|---|---|---|
| **2014-09-17** | 465.864014 | 468.174011 | 452.421997 | 457.334015 | 457.334015 | 21056800 | 457.334015 |
| **2014-09-18** | 456.859985 | 456.859985 | 413.104004 | 424.440002 | 424.440002 | 34483200 | 436.911062 |
| **2014-09-19** | 424.102997 | 427.834991 | 384.532013 | 394.795990 | 394.795990 | 37919700 | 419.823580 |
| **2014-09-20** | 394.673004 | 423.295990 | 389.882996 | 408.903992 | 408.903992 | 36863600 | 416.734836 |
| **2014-09-21** | 408.084991 | 412.425995 | 393.181000 | 398.821014 | 398.821014 | 26580100 | 413.700159 |

In [7]:
```
# Drop coulmns
df.drop(columns=["Adj Close", "Volume"])
```

Out[7]:

| Date | Open | High | Low | Close | VWAP |
|---|---|---|---|---|---|
| **2014-09-17** | 465.864014 | 468.174011 | 452.421997 | 457.334015 | 457.334015 |
| **2014-09-18** | 456.859985 | 456.859985 | 413.104004 | 424.440002 | 436.911062 |
| **2014-09-19** | 424.102997 | 427.834991 | 384.532013 | 394.795990 | 419.823580 |
| **2014-09-20** | 394.673004 | 423.295990 | 389.882996 | 408.903992 | 416.734836 |
| **2014-09-21** | 408.084991 | 412.425995 | 393.181000 | 398.821014 | 413.700159 |
| **...** | ... | ... | ... | ... | ... |
| **2022-06-06** | 29910.283203 | 31693.291016 | 29894.187500 | 31370.671875 | 28019.631692 |
| **2022-06-07** | 31371.742188 | 31489.683594 | 29311.683594 | 31155.478516 | 28022.598132 |
| **2022-06-08** | 31151.480469 | 31253.691406 | 29944.404297 | 30214.355469 | 28024.134972 |
| **2022-06-09** | 30215.279297 | 30609.310547 | 30020.265625 | 30111.998047 | 28025.184535 |
| **2022-06-10** | 30086.400391 | 30105.738281 | 29732.906250 | 29994.623047 | 28026.253558 |

2824 rows × 5 columns

In [8]:
```
df.info()
```

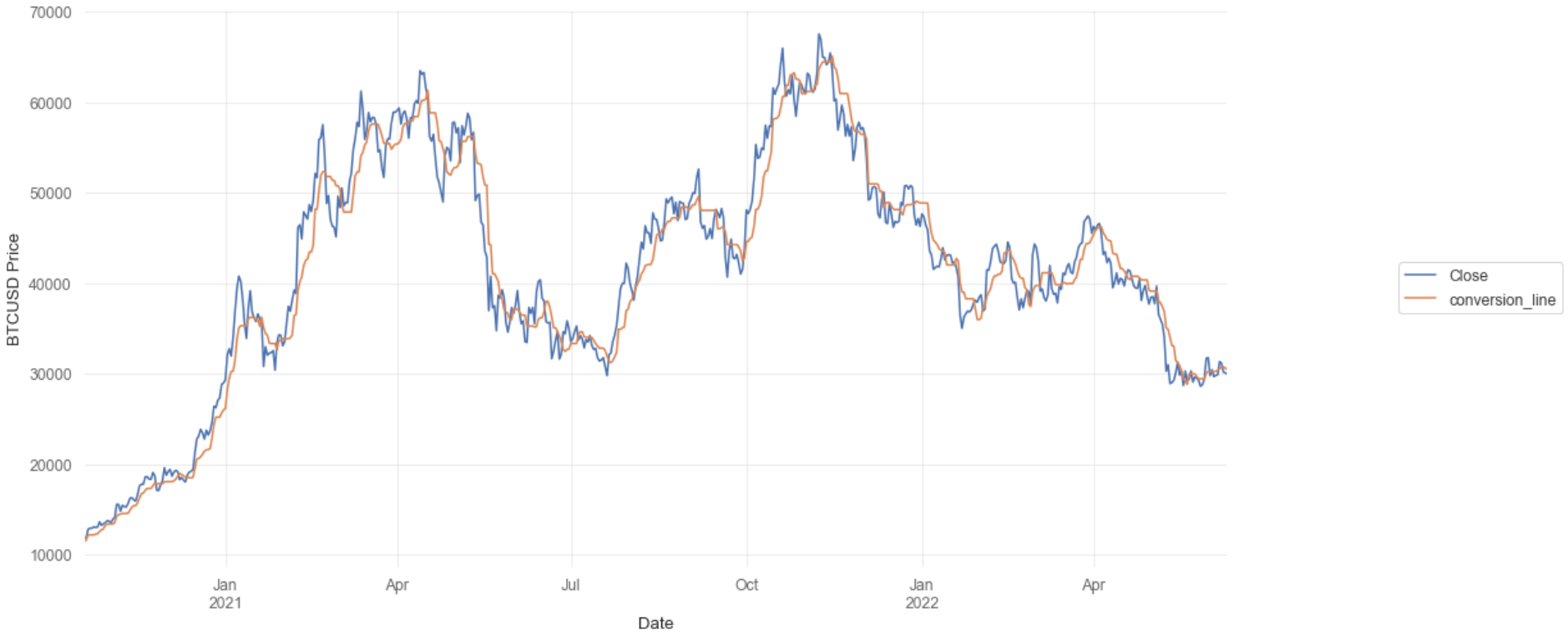
```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2824 entries, 2014-09-17 to 2022-06-10
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Open       2824 non-null   float64
 1   High       2824 non-null   float64
 2   Low        2824 non-null   float64
 3   Close      2824 non-null   float64
 4   Adj Close  2824 non-null   float64
```

```
 5   Volume     2824 non-null   int64
 6   VWAP       2824 non-null   float64
dtypes: float64(6), int64(1)
memory usage: 176.5 KB
```

In [9]:
```
# Convert to datetime index
df.index = pd.to_datetime(df.index)
```

In [10]:
```
df.Close.plot(figsize=(16, 8))
plt.ylabel("BTCUSD Price")
plt.show()
```

In [11]:
```
# Tenkansen (Conversion Line)
high_9 = df.High.rolling(9).max()
low_9 = df.Low.rolling(9).min()
df["conversion_line"] = (high_9 + low_9) / 2
conversion_line = df[["Close", "conversion_line"]]
conversion_line[-600:].plot(figsize=(16, 8))
plt.legend(loc="center right", bbox_to_anchor=(1.3, 0.5))
plt.ylabel("BTCUSD Price")
plt.show()
```




In [12]:
```
# Kijun-sen (Base Line)
high_26 = df.High.rolling(26).max()
low_26 = df.Low.rolling(26).min()
df["base_line"] = (high_26 + low_26) / 2

base_line = df[["Close", "conversion_line", "base_line"]]
base_line[-600:].plot(figsize=(16, 8))
```

```
plt.legend(loc="center right", bbox_to_anchor=(1.3, 0.5))
plt.ylabel("BTCUSD Price")
plt.show()
```


In [13]:

```
# Senkou Span A (Leading Span A)
df["leading_span_A"] = ((df.conversion_line + df.base_line) / 2).shift(30)

leading_span_A = df[["Close", "conversion_line",
                     "base_line", "leading_span_A"]]

leading_span_A[-600:].plot(figsize=(16, 8))
plt.legend(loc="center right", bbox_to_anchor=(1.3, 0.5))
plt.ylabel("BTCUSD Price")
plt.show()
```


In [14]:

```
# Senkou Span B (Leading Span B)
high_52 = df.High.rolling(52).max()
low_52 = df.Low.rolling(52).min()
df["leading_span_B"] = ((high_52 + low_52) / 2).shift(30)

leading_span_B = df[["Close", "conversion_line",
                     "base_line", "leading_span_A", "leading_span_B"]]
leading_span_B[-600:].plot(figsize=(16, 8))
plt.legend(loc="center right", bbox_to_anchor=(1.3, 0.5))
plt.ylabel("BTCUSD Price")
plt.show()
```


In [15]:

```
# Ichimoku Cloud
cloud = df[-600:].Close.plot(figsize=(16, 8))
cloud.fill_between(
    df[-600:].index, df[-600:].leading_span_A, df[-600:].leading_span_B, color="grey")
plt.legend(loc="center right", bbox_to_anchor=(1.3, 0.5))
plt.ylabel("BTCUSD Price")
plt.show()
```


In [16]:
```
# RSI indicator settings
df['RSI'] = ta.RSI(df.Close,14)
```

In [17]:
```
# Entry setup
df["signal"] = np.nan

# Prices are above the cloud
condition_1 = (df.Close > df.leading_span_A) & (df.Close > df.leading_span_B)

# leading Span A (senkou_span_A) is greater than leading span B (senkou_span_B)
condition_2 = (df.leading_span_A > df.leading_span_B)

# # Conversion Line (tenkan_sen) moves above Base Line (kijun_sen)
condition_3 = (df.conversion_line > df.base_line)

# RSI momentum
condition_4 = (df.RSI > 50)
```

```
# Combine the conditions and store in the signal column 1 when all the conditions are true
df.loc[condition_1 & condition_2 & condition_3 & condition_4, "signal"] = 1
```

In [18]:

```
# Exit (to cash)
# Price closes below the cloud
condition_1 = (df.Close < df.leading_span_A)

# Store condition in signal column 0 when true
df.loc[(condition_1), "signal"] = 0

# If signal NA foward fill with previous signal
df.signal.fillna(method="ffill", inplace=True)

df.iloc[320:360, :]
```

Out[18]:

| Date | Open | High | Low | Close | Adj Close | Volume | VWAP | conversion_line | base_line | leading_span_A | leading_span_B | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **2015-08-03** | 282.806000 | 285.471008 | 280.233002 | 281.226990 | 281.226990 | 21474100 | 286.633870 | 287.651489 | 290.740005 | 251.304001 | 244.581505 | 51.7 |
| **2015-08-04** | 281.225006 | 285.714996 | 281.225006 | 285.217987 | 285.217987 | 21908700 | 286.630108 | 287.651489 | 291.598007 | 255.034756 | 247.901009 | 54.9 |
| **2015-08-05** | 284.846985 | 285.501007 | 281.488007 | 281.881989 | 281.881989 | 20128000 | 286.618547 | 287.088989 | 293.218506 | 257.761250 | 249.359001 | 51.3 |
| **2015-08-06** | 281.906006 | 281.906006 | 278.403015 | 278.576996 | 278.576996 | 18792100 | 286.600308 | 286.032501 | 293.218506 | 258.158749 | 249.359001 | 47.9 |
| **2015-08-07** | 278.740997 | 280.391998 | 276.365997 | 279.584991 | 279.584991 | 42484800 | 286.564519 | 283.246002 | 291.495499 | 259.389751 | 249.359001 | 49.( |
| **2015-08-08** | 279.742004 | 279.928009 | 260.709991 | 260.997009 | 260.997009 | 58533000 | 286.386068 | 274.834503 | 279.241989 | 260.088249 | 249.359001 | 34.( |
| **2015-08-09** | 261.115997 | 267.002991 | 260.467987 | 265.083008 | 265.083008 | 23789600 | 286.325808 | 273.091492 | 279.120987 | 269.027252 | 257.943504 | 38.8 |
| **2015-08-10** | 265.477997 | 267.032013 | 262.596008 | 264.470001 | 264.470001 | 20979400 | 286.271423 | 273.091492 | 279.120987 | 271.659756 | 259.901009 | 38.4 |

| Date | Open | High | Low | Close | Adj Close | Volume | VWAP | conversion_line | base_line | leading_span_A | leading_span_B | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2015-08-11 | 264.342010 | 270.385986 | 264.093994 | 270.385986 | 270.385986 | 25433900 | 286.223646 | 273.091492 | 279.120987 | 280.838257 | 267.845009 | 44.4 |
| 2015-08-12 | 270.597992 | 270.673004 | 265.468994 | 266.376007 | 266.376007 | 26815400 | 286.160909 | 273.091492 | 279.120987 | 281.963505 | 267.845009 | 41.5 |
| 2015-08-13 | 266.183014 | 266.231995 | 262.841003 | 264.079987 | 264.079987 | 27685500 | 286.089082 | 272.984497 | 279.120987 | 283.384754 | 267.845009 | 39.8 |
| 2015-08-14 | 264.131989 | 267.466003 | 261.477997 | 265.679993 | 265.679993 | 27091200 | 286.024324 | 271.186996 | 279.120987 | 283.384754 | 267.845009 | 41.( |
| 2015-08-15 | 265.528992 | 266.666992 | 261.295990 | 261.550995 | 261.550995 | 19321100 | 285.969068 | 270.429993 | 279.120987 | 283.384754 | 267.845009 | 38.5 |
| 2015-08-16 | 261.865997 | 262.440002 | 257.040985 | 258.506989 | 258.506989 | 29717000 | 285.874031 | 268.484497 | 277.407486 | 284.059757 | 267.845009 | 36.4 |
| 2015-08-17 | 258.489990 | 260.505005 | 257.117004 | 257.976013 | 257.976013 | 21617900 | 285.803975 | 263.856995 | 277.407486 | 284.488758 | 267.845009 | 36.( |
| 2015-08-18 | 257.925995 | 257.993011 | 211.078995 | 211.078995 | 211.078995 | 42147200 | 285.439916 | 240.875999 | 254.426491 | 285.299007 | 267.845009 | 18.7 |
| 2015-08-19 | 225.671005 | 237.408997 | 222.766006 | 226.684006 | 226.684006 | 60869200 | 285.029388 | 240.875999 | 254.426491 | 285.299007 | 267.845009 | 30.5 |
| 2015-08-20 | 226.899002 | 237.365005 | 226.899002 | 235.350006 | 235.350006 | 32275000 | 284.846017 | 240.875999 | 254.426491 | 284.734501 | 267.845009 | 36.2 |
| 2015-08-21 | 235.354996 | 236.432007 | 231.723999 | 232.569000 | 232.569000 | 23173800 | 284.707837 | 239.272499 | 254.426491 | 281.425255 | 267.845009 | 35.2 |
| 2015-08-22 | 232.662003 | 234.957001 | 222.703995 | 230.389999 | 230.389999 | 23205900 | 284.564443 | 239.272499 | 253.863991 | 281.779751 | 268.406509 | 34.4 |
| 2015-08-23 | 230.376007 | 232.705002 | 225.580002 | 228.169006 | 228.169006 | 18406600 | 284.446602 | 238.872993 | 252.807503 | 281.550255 | 268.523003 | 33.( |
| 2015-08-24 | 228.112000 | 228.139008 | 210.442993 | 210.494995 | 210.494995 | 59220700 | 283.952755 | 236.441498 | 250.284500 | 282.668755 | 268.523003 | 27.7 |

| Date | Open | High | Low | Close | Adj Close | Volume | VWAP | conversion_line | base_line | leading_span_A | leading_span_B | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2015-08-25 | 210.067993 | 226.320999 | 199.567001 | 221.608994 | 221.608994 | 61089200 | 283.526227 | 230.036003 | 244.263008 | 283.366005 | 268.523003 | 35.3 |
| 2015-08-26 | 222.076004 | 231.182999 | 220.203995 | 225.830994 | 225.830994 | 31808000 | 283.321431 | 228.780006 | 242.640999 | 284.546501 | 268.523003 | 38.( |
| 2015-08-27 | 226.050003 | 228.643005 | 223.684006 | 224.768997 | 224.768997 | 21905400 | 283.178646 | 218.487999 | 242.640999 | 284.658249 | 268.523003 | 37.( |
| 2015-08-28 | 224.701004 | 235.218994 | 220.925995 | 231.395996 | 231.395996 | 31336600 | 282.998631 | 218.466003 | 242.640999 | 285.187252 | 268.616508 | 41.9 |
| 2015-08-29 | 231.548996 | 233.222000 | 227.330002 | 229.779999 | 229.779999 | 17142500 | 282.897616 | 217.999504 | 242.640999 | 286.312500 | 271.161507 | 41.1 |
| 2015-08-30 | 229.895004 | 232.067993 | 226.246994 | 228.761002 | 228.761002 | 19412600 | 282.781501 | 217.392998 | 242.534004 | 287.951996 | 271.202003 | 40.( |
| 2015-08-31 | 229.113998 | 231.955994 | 225.914993 | 230.056000 | 230.056000 | 20710700 | 282.661125 | 217.392998 | 240.736504 | 287.951996 | 271.580505 | 41.( |
| 2015-09-01 | 230.255997 | 231.216003 | 226.860001 | 228.121002 | 228.121002 | 20575200 | 282.537701 | 217.392998 | 239.979500 | 288.520744 | 271.802010 | 40.5 |
| 2015-09-02 | 228.026993 | 230.576996 | 226.475006 | 229.283997 | 229.283997 | 18760400 | 282.428045 | 217.392998 | 239.747505 | 289.195747 | 271.802010 | 41.5 |
| 2015-09-03 | 229.324005 | 229.604996 | 226.667007 | 227.182999 | 227.182999 | 17482000 | 282.322242 | 227.711494 | 235.120003 | 289.624748 | 273.199005 | 40.2 |
| 2015-09-04 | 227.214996 | 230.899994 | 227.050995 | 230.298004 | 230.298004 | 20962400 | 282.203046 | 228.072495 | 235.120003 | 290.153748 | 273.908005 | 43.( |
| 2015-09-05 | 230.199005 | 236.143005 | 229.442993 | 235.018997 | 235.018997 | 20671400 | 282.096681 | 228.534500 | 235.120003 | 289.625504 | 275.258003 | 47.( |
| 2015-09-06 | 234.869995 | 242.912003 | 234.681000 | 239.839996 | 239.839996 | 25473700 | 281.979619 | 234.413498 | 235.120003 | 287.370750 | 277.379509 | 50.8 |
| 2015-09-07 | 239.934006 | 242.106003 | 238.722000 | 239.847000 | 239.847000 | 21192200 | 281.882741 | 234.413498 | 233.516502 | 277.038246 | 277.379509 | 50.8 |

| Date | Open | High | Low | Close | Adj Close | Volume | VWAP | conversion_line | base_line | leading_span_A | leading_span_B | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **2015-09-08** | 239.845993 | 245.781006 | 239.677994 | 243.606995 | 243.606995 | 26879200 | 281.771438 | 235.848000 | 233.516502 | 276.106239 | 277.379509 | 53.8 |
| **2015-09-09** | 243.414993 | 244.416000 | 237.820999 | 238.167999 | 238.167999 | 23635700 | 281.660228 | 236.128006 | 233.116997 | 276.106239 | 277.379509 | 49.1 |
| **2015-09-10** | 238.335999 | 241.292999 | 235.791000 | 238.477005 | 238.477005 | 21215500 | 281.561593 | 236.128006 | 231.003502 | 276.106239 | 277.379509 | 49.4 |
| **2015-09-11** | 238.328995 | 241.169006 | 238.328995 | 240.106995 | 240.106995 | 19224700 | 281.475968 | 236.224007 | 230.036003 | 276.106239 | 277.379509 | 50.9 |

In [19]:
```
df["signal_change"] = df.signal.diff()
df["signal_change"].value_counts()
```

Out[19]:
```
 0.0    2709
 1.0      25
-1.0      25
Name: signal_change, dtype: int64
```

In [20]:
```
# Visualize entry position relative to close price
entry = df[df["signal_change"] == 1.0]["Close"].hvplot.scatter(
    color="green",
    marker="^",
    size=200,
    legend=False,
    ylabel="Price in $",
    width=1000,
    height=400
)

# Visualize exit position relative to close price
exit = df[df["signal_change"] == -1.0]["Close"].hvplot.scatter(
    color="red",
    marker="v",
    size=200,
    legend=False,
```

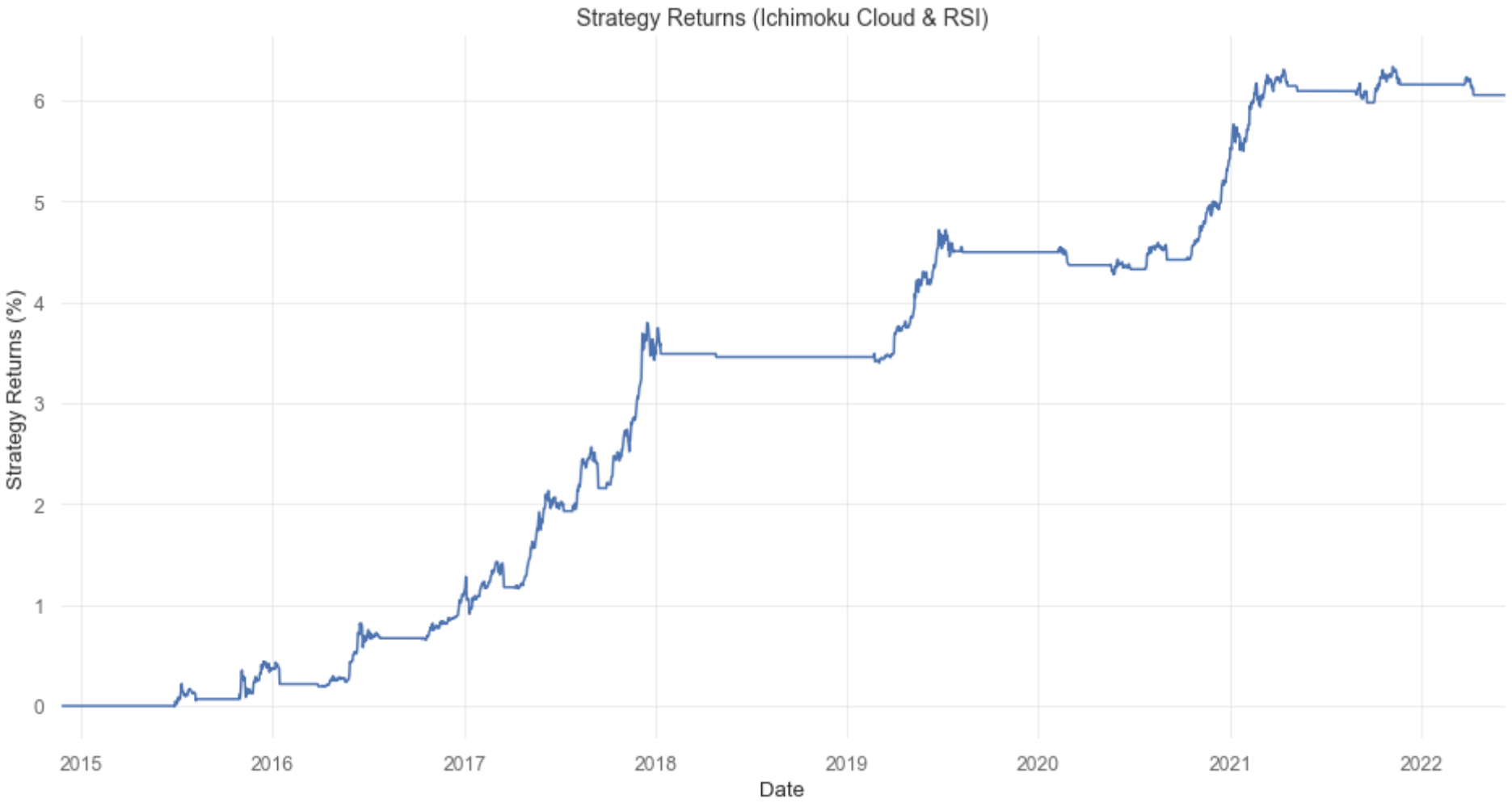
```
    ylabel="Price in $",
    width=1000,
    height=400
)


# Visualize close price for the investment
security_close = df[["Close"]].hvplot(
    line_color="lightgray",
    ylabel="Price in $",
    width=1000,
    height=400
)

# Plot Ichimoku indicators
ichi = df[["Close", "conversion_line",
                    "base_line", "leading_span_A", "leading_span_B"]].hvplot(
    ylabel="Price in $",
    width=1000,
    height=400
)


# # Overlay plots
ichiplot = security_close * ichi * entry * exit
ichiplot
```

Out[20]:

In [21]:

```
# Calculate daily returns
daily_returns = df.Close.pct_change()

# Calculate strategy returns
strategy_returns = daily_returns * df.signal.shift(1)
strategy_returns.dropna(inplace=True)

# Plot strategy returns
strategy_returns.cumsum().plot(figsize=(16, 8))
plt.xlabel("Date")
plt.ylabel("Strategy Returns (%)")
```

```
plt.title("Strategy Returns (Ichimoku Cloud & RSI)", fontsize=14)
plt.show()
```



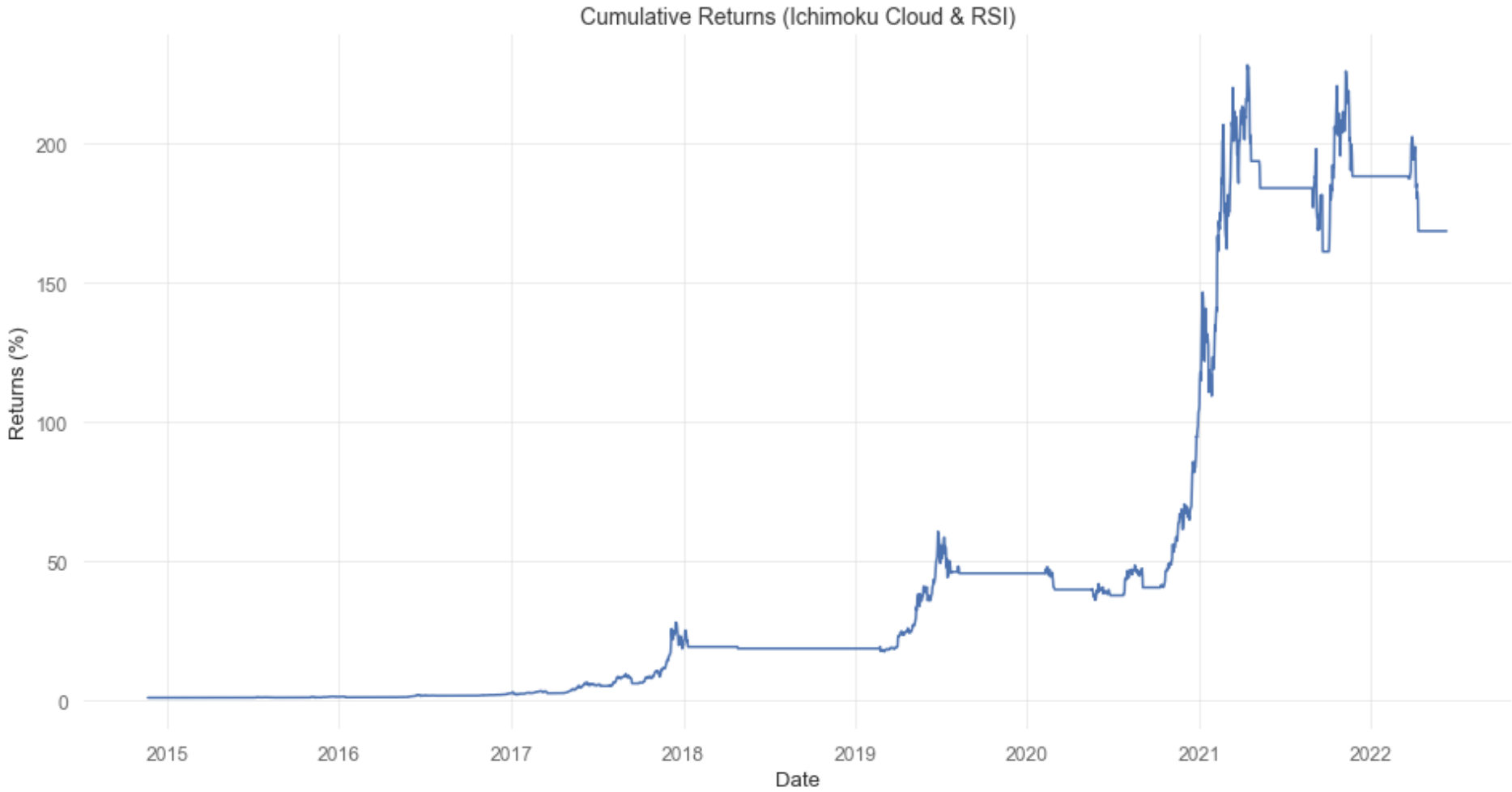

In [22]:

```
# Check Sharpe ratio calculation
def annualized_sharpe_ratio(returns, N=252):
    return ((N) * returns.mean()) / (returns.std() * np.sqrt(N))

# Sharpe ratio
excess_daily_strategy_return = strategy_returns
sharpe = annualized_sharpe_ratio(excess_daily_strategy_return)
print("The Sharpe ratio of strategy is %.2f" % sharpe)
```

```
The Sharpe ratio of strategy is 1.34
```

In [23]:

```
# Calculate the cumulative returns
df["cumulative_returns"] = (strategy_returns+1).cumprod()

# Plot the cumulative returns
plt.figure(figsize=(16, 8))
plt.plot(df["cumulative_returns"])
plt.title("Cumulative Returns (Ichimoku Cloud & RSI)", fontsize=14)
plt.xlabel("Date")
plt.ylabel("Returns (%)")
plt.show()
```

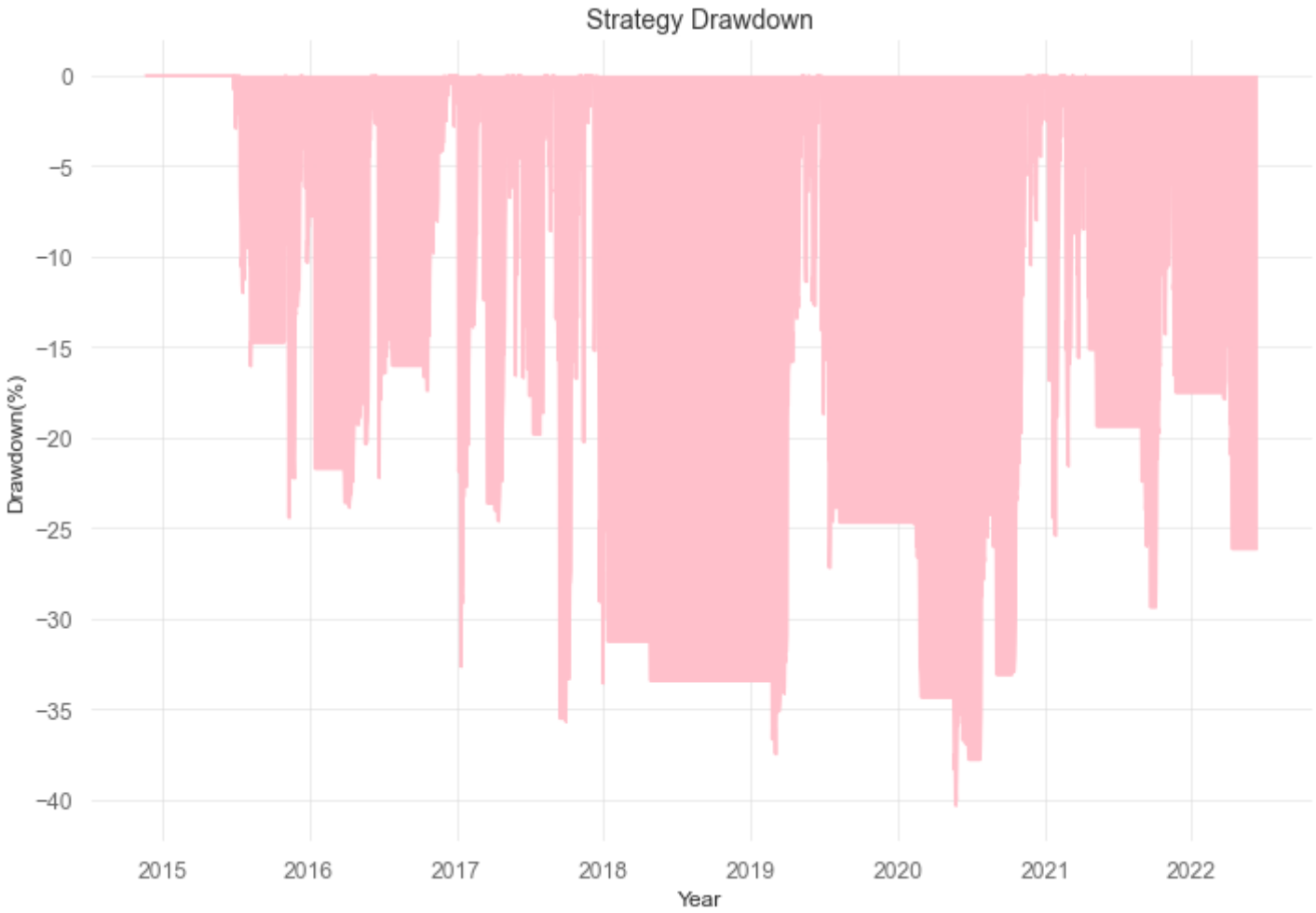



In [24]:
```
# strategy_returns.value_counts()
```

In [25]:
```
# Calculate the running maximum
running_max = np.maximum.accumulate(df["cumulative_returns"].dropna())
# Ensure the value never drops below 1
running_max[running_max < 1] = 1
# Calculate the percentage drawdown
drawdown = ((df["cumulative_returns"])/running_max - 1) * 100

# Calculate the maximum drawdown
```

```
print("Maximum drawdown of the strategy is {0:.2f}%".format(drawdown.min()))

fig = plt.figure(figsize=(10, 7))

# Plot max drawdown
plt.plot(drawdown, color="pink")
# Fill in-between the drawdown
plt.fill_between(drawdown.index, drawdown.values, color="pink")
plt.title("Strategy Drawdown", fontsize=14)
plt.ylabel("Drawdown(%)", fontsize=12)
plt.xlabel("Year", fontsize=12)

plt.tight_layout()
plt.show()
```

```
Maximum drawdown of the strategy is -40.32%
```




In [27]:
```
# Extend pandas functionality with metrics
qs.extend_pandas()
```

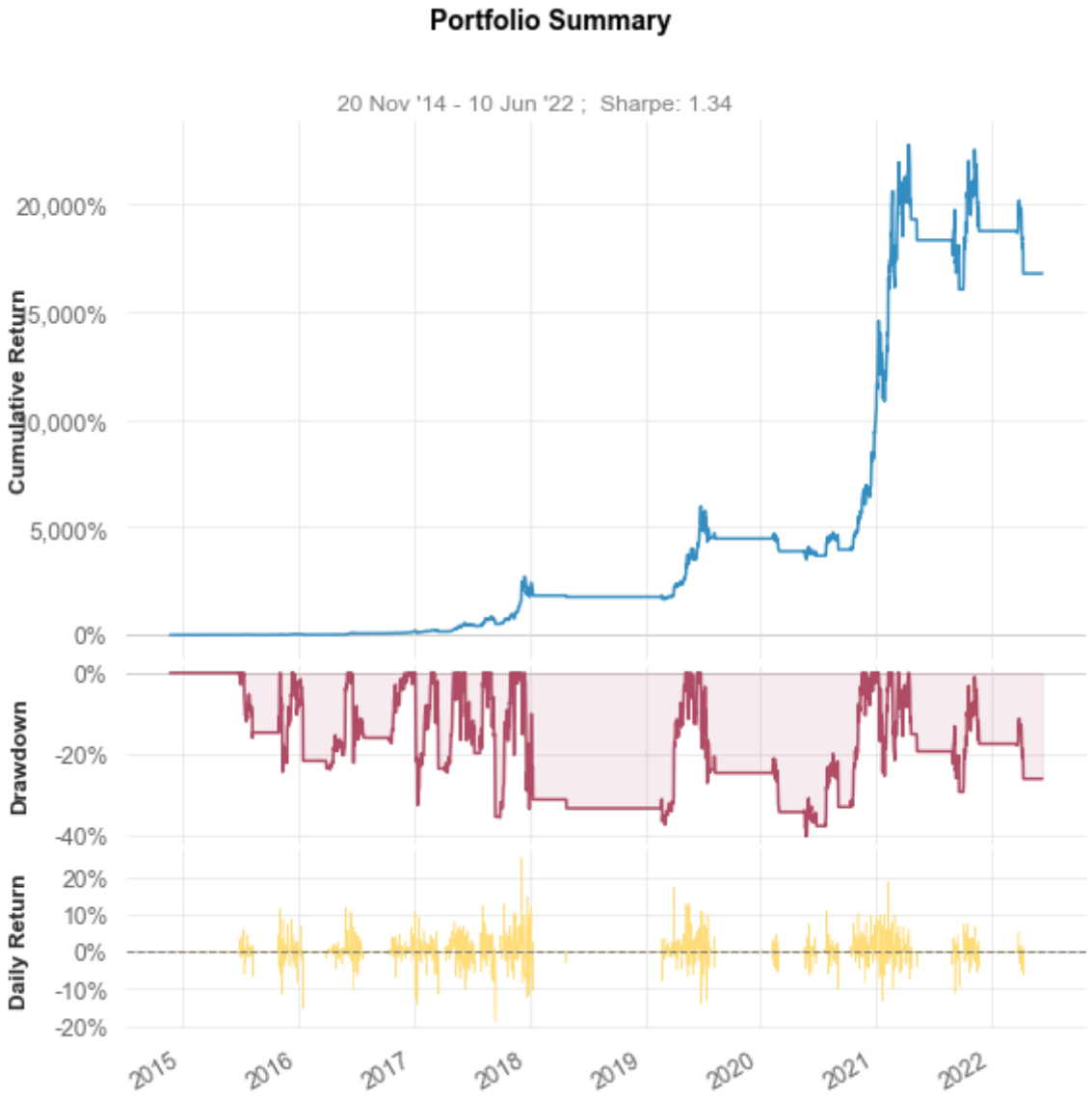
In [28]:
```
# View full perfomance metrics
qs.reports.basic(strategy_returns)
```

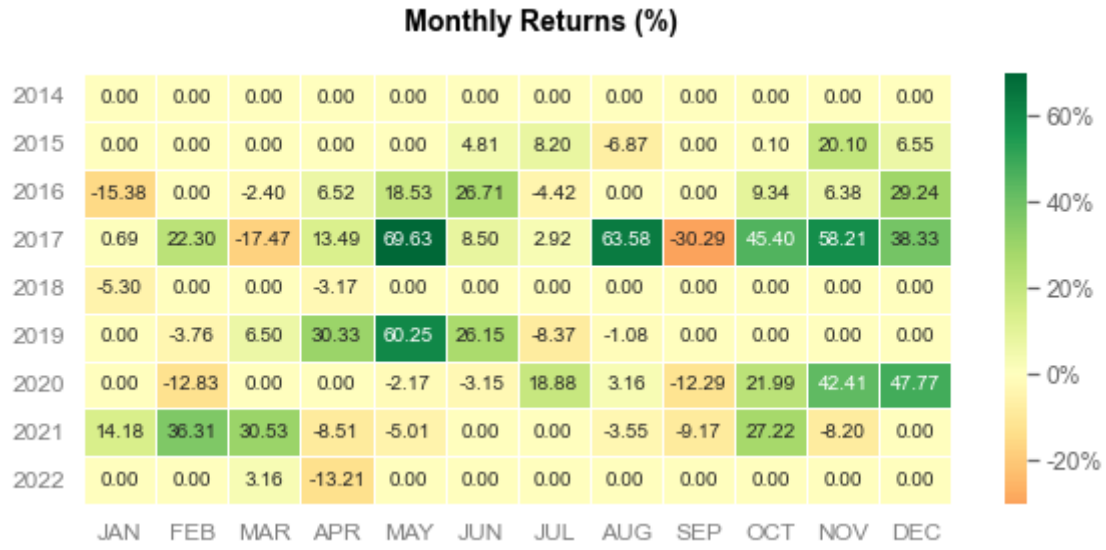
## Performance Metrics

| | Strategy |
|---|---|

| | |
|---|---|
| Start Period | 2014-11-21 |
| End Period | 2022-06-10 |
| Risk-Free Rate | 0.0% |
| Time in Market | 43.0% |
| Cumulative Return | 16,758.79% |
| CAGR% | 97.11% |
| Sharpe | 1.34 |
| Prob. Sharpe Ratio | 100.0% |
| Sortino | 2.12 |
| Sortino/√2 | 1.5 |
| Omega | 1.47 |
| Max Drawdown | -40.32% |
| Longest DD Days | 509 |
| Gain/Pain Ratio | 0.47 |
| Gain/Pain (1M) | 3.59 |
| Payoff Ratio | 1.06 |
| Profit Factor | 1.47 |
| Common Sense Ratio | 1.93 |
| CPC Index | 0.9 |
| Tail Ratio | 1.31 |
| Outlier Win Ratio | 11.87 |
| Outlier Loss Ratio | 3.01 |
| MTD | 0.0% |
| 3M | -10.47% |
| 6M | -10.47% |
| YTD | -10.47% |
| 1Y | -8.4% |
| 3Y (ann.) | 43.59% |
| 5Y (ann.) | 100.23% |
| 10Y (ann.) | 97.11% |
| All-time (ann.) | 97.11% |
| Avg. Drawdown | -9.52% |
| Avg. Drawdown Days | 41 |
| Recovery Factor | 415.7 |
| Ulcer Index | 0.22 |
| Serenity Index | 53.18 |

## Strategy Visualization






In [29]:
```
# View full perfomance metrics
qs.reports.full(strategy_returns, "BTC-USD")
```

## Performance Metrics

| | Strategy | Benchmark |
|---|---|---|
| Start Period | 2014-11-21 | 2014-11-21 |
| End Period | 2022-06-10 | 2022-06-10 |
| Risk-Free Rate | 0.0% | 0.0% |
| Time in Market | 43.0% | 100.0% |
| Cumulative Return | 16,758.79% | 8,449.18% |
| CAGR% | 97.11% | 80.17% |
| Sharpe | 1.34 | 0.97 |
| Prob. Sharpe Ratio | 100.0% | 99.94% |
| Smart Sharpe | 1.27 | 0.92 |
| Sortino | 2.12 | 1.43 |
| Smart Sortino | 2.0 | 1.35 |
| Sortino/√2 | 1.5 | 1.01 |
| Smart Sortino/√2 | 1.41 | 0.95 |
| Omega | 1.47 | 1.47 |

| | | |
|---|---|---|
| Max Drawdown | -40.32% | -83.4% |
| Longest DD Days | 509 | 1079 |
| Volatility (ann.) | 41.22% | 61.44% |
| R^2 | 0.45 | 0.45 |
| Information Ratio | -0.01 | -0.01 |
| Calmar | 2.41 | 0.96 |
| Skew | 0.65 | -0.16 |
| Kurtosis | 13.55 | 7.3 |
| | | |
| Expected Daily % | 0.19% | 0.16% |
| Expected Monthly % | 5.73% | 4.95% |
| Expected Yearly % | 76.78% | 63.93% |
| Kelly Criterion | 18.52% | 10.98% |
| Risk of Ruin | 0.0% | 0.0% |
| Daily Value-at-Risk | -4.05% | -6.13% |
| Expected Shortfall (cVaR) | -4.05% | -6.13% |
| | | |
| Max Consecutive Wins | 13 | 13 |
| Max Consecutive Losses | 6 | 7 |
| Gain/Pain Ratio | 0.47 | 0.2 |
| Gain/Pain (1M) | 3.59 | 1.37 |
| | | |
| Payoff Ratio | 1.06 | 1.06 |
| Profit Factor | 1.47 | 1.2 |
| Common Sense Ratio | 1.93 | 1.28 |
| CPC Index | 0.9 | 0.69 |
| Tail Ratio | 1.31 | 1.06 |
| Outlier Win Ratio | 12.71 | 4.14 |
| Outlier Loss Ratio | 3.55 | 3.65 |
| | | |
| MTD | 0.0% | -5.65% |
| 3M | -10.47% | -28.56% |
| 6M | -10.47% | -37.08% |
| YTD | -10.47% | -35.23% |
| 1Y | -8.4% | -19.68% |
| 3Y (ann.) | 43.59% | 34.67% |
| 5Y (ann.) | 100.23% | 65.45% |
| 10Y (ann.) | 97.11% | 80.17% |
| All-time (ann.) | 97.11% | 80.17% |
| | | |
| Best Day | 25.25% | 25.25% |
| Worst Day | -18.74% | -37.17% |
| Best Month | 69.63% | 69.63% |
| Worst Month | -30.29% | -36.41% |

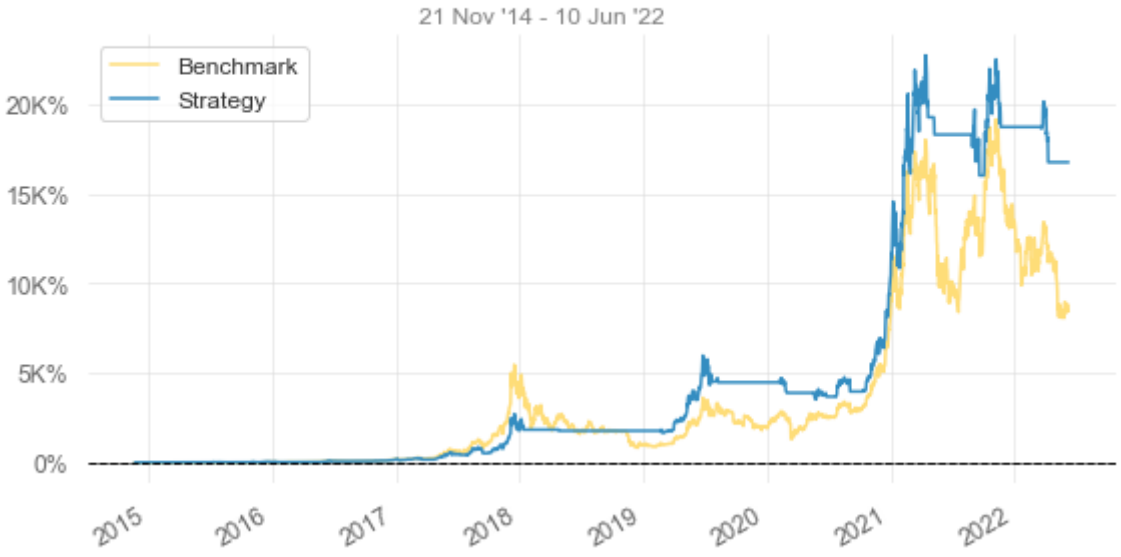
```
Best Year                692.81%     1368.9%
Worst Year               -10.47%     -73.56%

Avg. Drawdown            -9.52%      -11.41%
Avg. Drawdown Days       41          43
Recovery Factor          415.7       101.31
Ulcer Index              0.22        0.42
Serenity Index           53.18       10.68

Avg. Up Month            23.67%      26.84%
Avg. Down Month          -9.64%      -11.14%
Win Days %               58.07%      54.19%
Win Month %              62.5%       55.43%
Win Quarter %            61.54%      54.84%
Win Year %               75.0%       66.67%

Beta                     0.45        -
Alpha                    0.28        -
Correlation              67.06%      -
Treynor Ratio            37239.85%   -
None
```

## 5 Worst Drawdowns

| | Start | Valley | End | Days | Max Drawdown | 99% Max Drawdown |
|---|---|---|---|---|---|---|
| **1** | 2019-06-27 | 2020-05-24 | 2020-11-17 | 509 | -40.315026 | -37.710178 |
| **2** | 2017-12-17 | 2019-03-04 | 2019-05-09 | 508 | -37.479480 | -35.850188 |
| **3** | 2017-09-02 | 2017-09-29 | 2017-11-01 | 60 | -35.688230 | -35.508102 |
| **4** | 2017-01-05 | 2017-01-11 | 2017-02-23 | 49 | -32.645984 | -30.301110 |
| **5** | 2021-04-14 | 2021-09-20 | 2022-06-10 | 422 | -29.353218 | -28.015106 |

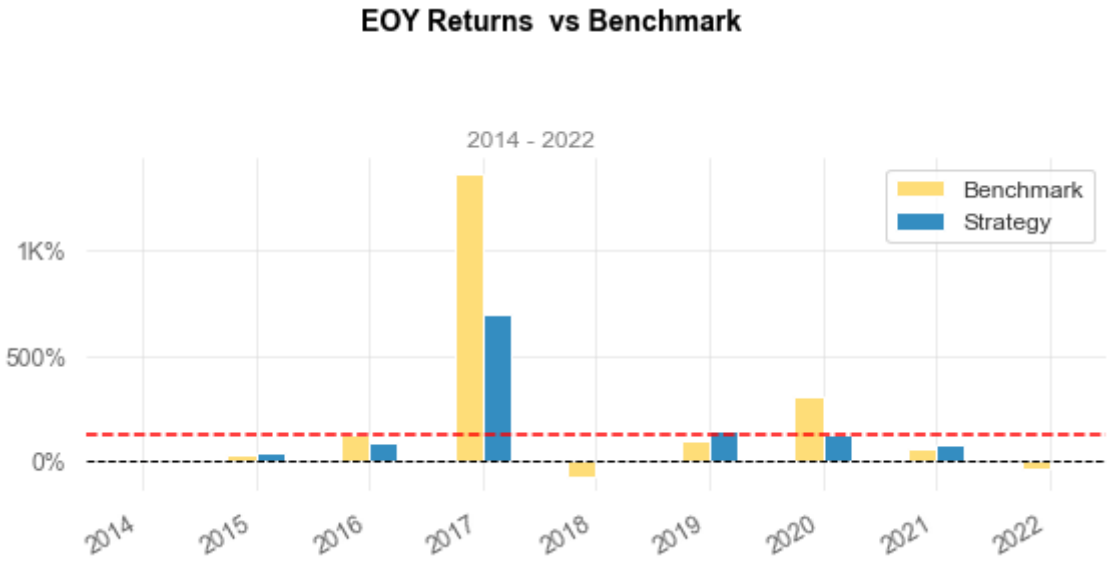
## Strategy Visualization

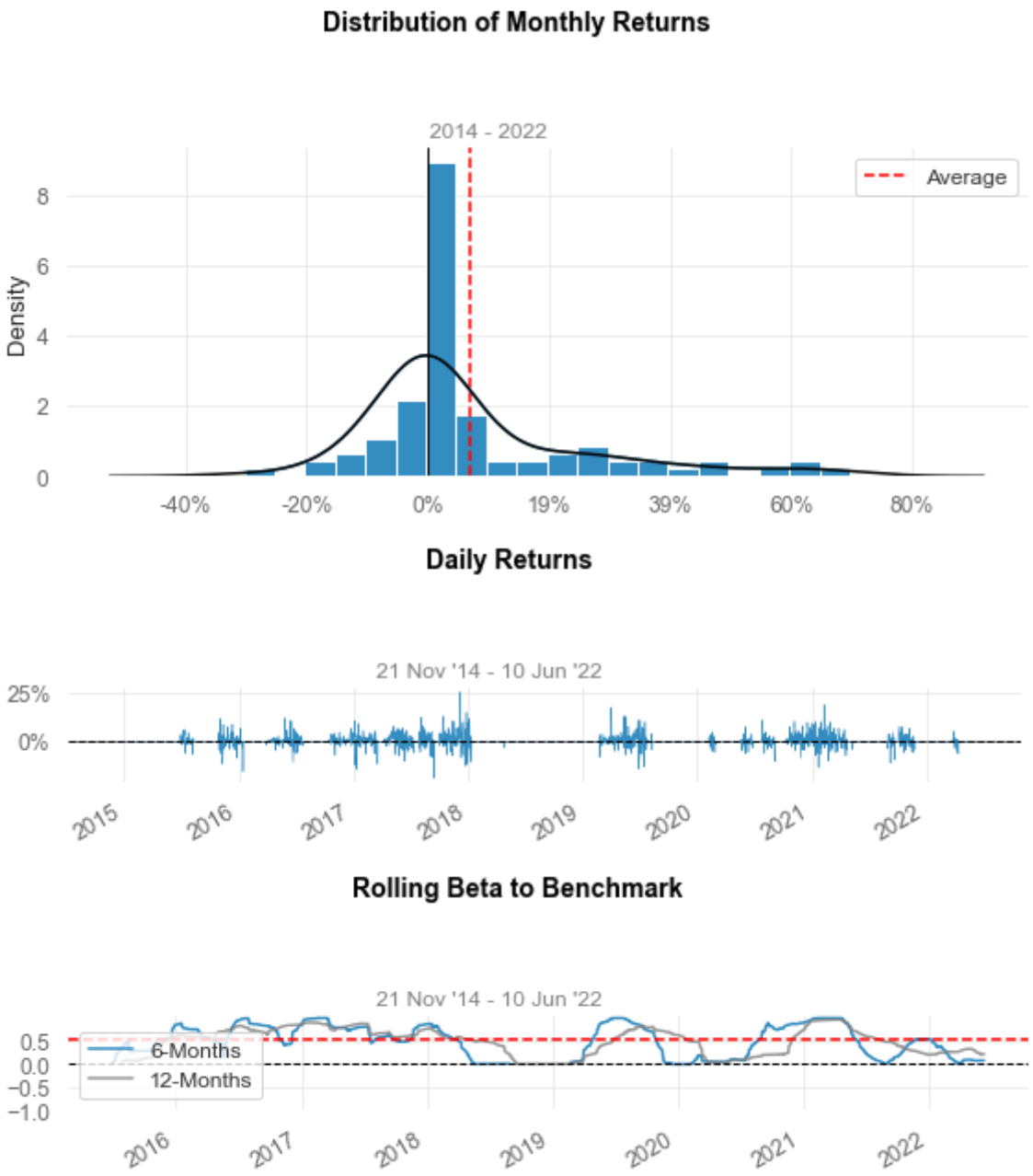
**Cumulative Returns vs Benchmark**



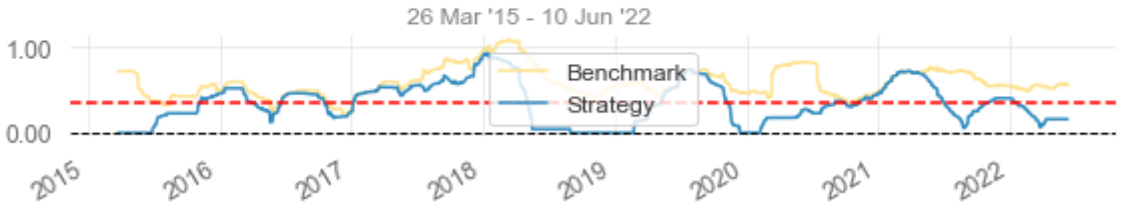

**Cumulative Returns vs Benchmark (Log Scaled)**


Cumulative Returns vs Benchmark (Volatility Matched)
21 Nov '14 - 10 Jun '22
Benchmark
Strategy
150K%
100K%
50K%
0%
2015
2016
2017
2018
2019
2020
2021
2022


EOY Returns vs Benchmark
2014 - 2022
Benchmark
Strategy
1K%
500%
0%
2014
2015
2016
2017
2018
2019
2020
2021
2022

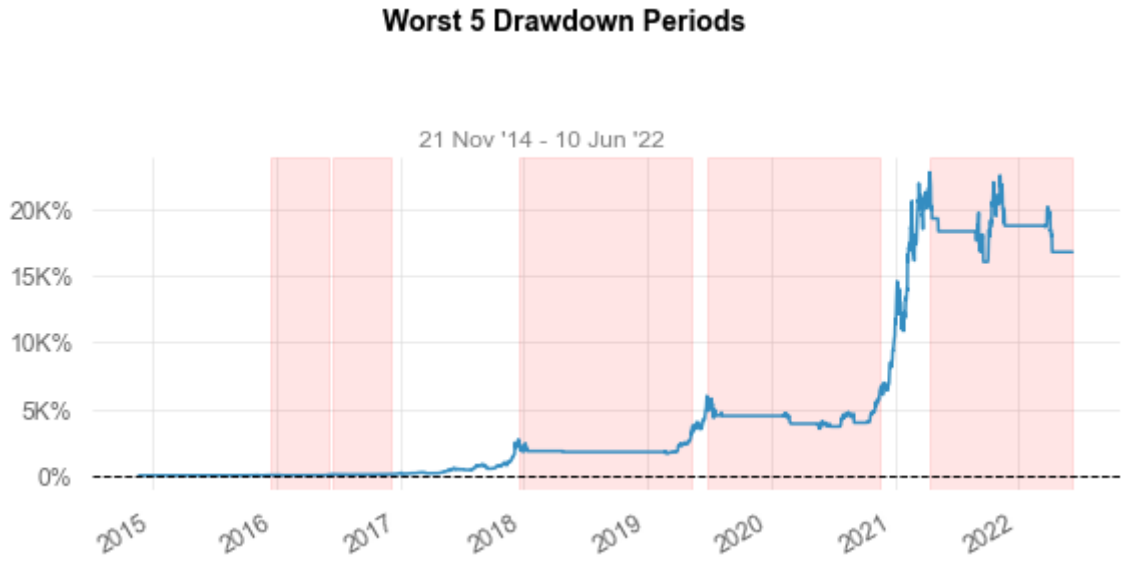

Distribution of Monthly Returns
2014 - 2022
Average
Density
8
6
4
2
0
-40%
-20%
0%
19%
39%
60%
80%
Daily Returns
21 Nov '14 - 10 Jun '22
25%
0%
2015
2016
2017
2018
2019
2020
2021
2022
Rolling Beta to Benchmark
21 Nov '14 - 10 Jun '22
0.5
0.0
−0.5
−1.0
6-Months
12-Months
2016
2017
2018
2019
2020
2021
2022

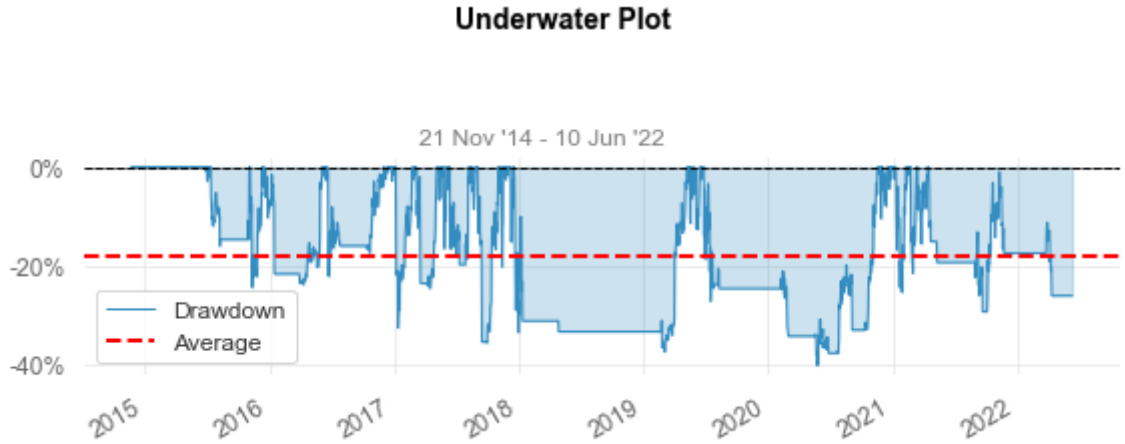
**Rolling Volatility (6-Months)**

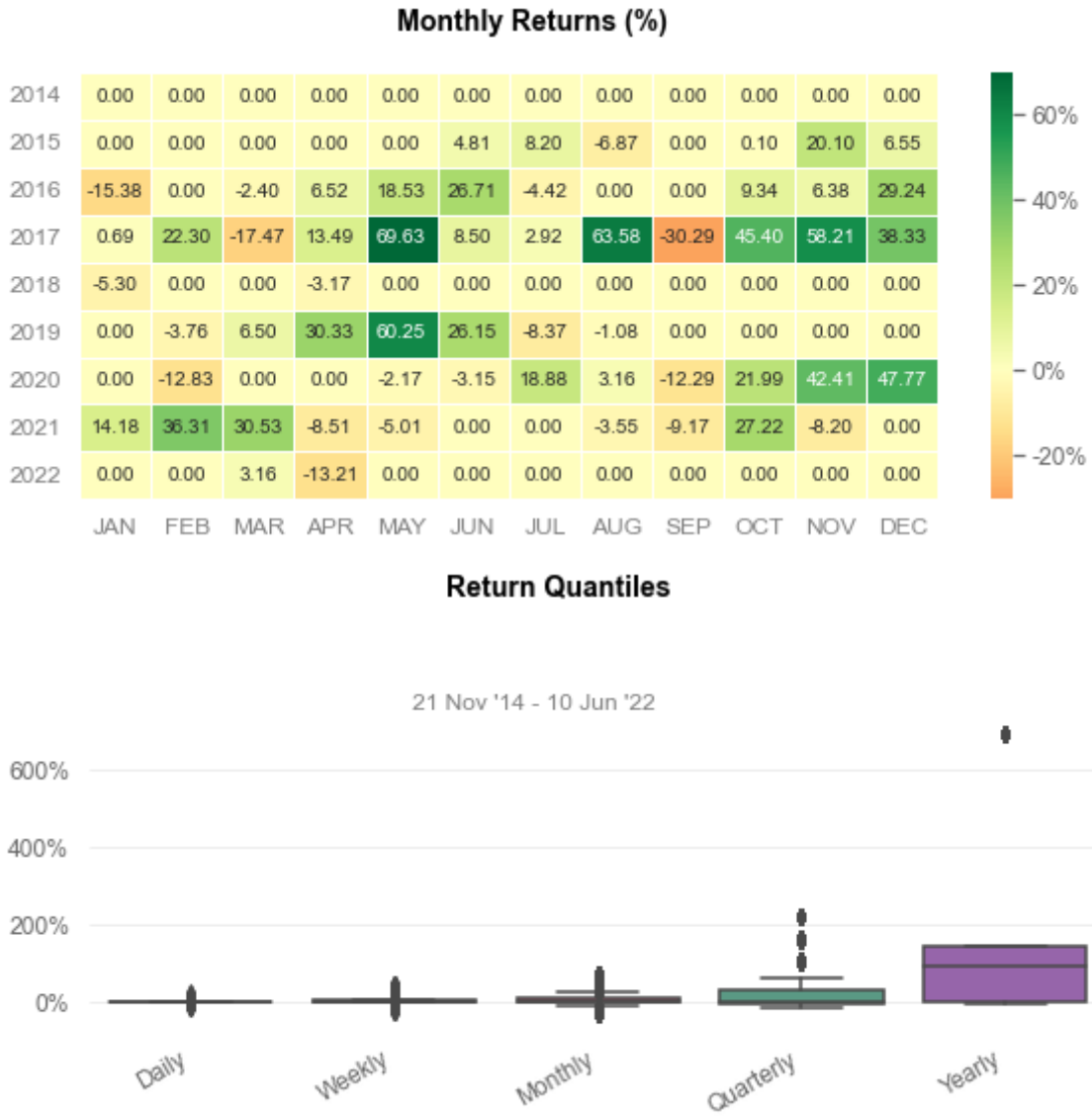



**Rolling Sharpe (6-Months)**


**Rolling Sortino (6-Months)**


Worst 5 Drawdown Periods
21 Nov '14 - 10 Jun '22
20K%
15K%
10K%
5K%
0%
2015
2016
2017
2018
2019
2020
2021
2022


Underwater Plot
21 Nov '14 - 10 Jun '22
0%
-20%
-40%
Drawdown
Average
2015
2016
2017
2018
2019
2020
2021
2022


Monthly Returns (%)
2014 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
2015 0.00 0.00 0.00 0.00 0.00 4.81 8.20 -6.87 0.00 0.10 20.10 6.55
2016 -15.38 0.00 -2.40 6.52 18.53 26.71 -4.42 0.00 0.00 9.34 6.38 29.24
2017 0.69 22.30 -17.47 13.49 69.63 8.50 2.92 63.58 -30.29 45.40 58.21 38.33
2018 -5.30 0.00 0.00 -3.17 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
2019 0.00 -3.76 6.50 30.33 60.25 26.15 -8.37 -1.08 0.00 0.00 0.00 0.00
2020 0.00 -12.83 0.00 0.00 -2.17 -3.15 18.88 3.16 -12.29 21.99 42.41 47.77
2021 14.18 36.31 30.53 -8.51 -5.01 0.00 0.00 -3.55 -9.17 27.22 -8.20 0.00
2022 0.00 0.00 3.16 -13.21 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
JAN FEB MAR APR MAY JUN JUL AUG SEP OCT NOV DEC
60%
40%
20%
0%
-20%
Return Quantiles
21 Nov '14 - 10 Jun '22
600%
400%
200%
0%
Daily
Weekly
Monthly
Quarterly
Yearly

In [ ]: