In [1]:
```
from bokeh.plotting import figure, show
from bokeh.palettes import Spectral11
from bokeh.io import output_notebook
from bokeh.models import Legend
import backtrader as bt
import backtrader.analyzers as btanalyzers
import matplotlib
from datetime import datetime, timedelta
from math import pi

output_notebook()
```

```
BokehJS 2.3.1 successfully loaded.
```

In [2]:
```
### Set up which stocks to trade on and other hyperparameters ###

stocks = [
    "AAPL",
    "MSFT",
    "AMZN",
    "FB",
    "GOOGL",
    "TSLA",
    "JPM",
    "NVDA",
    "INTC",
    "VZ",
    "NFLX",
    "QCOM",
    "GS",
    "IBM",
    "WFC",
    "PNC",
    "C",
    "COF",
    "BHLB",
    "BAC"
]

max_shares_position = 1000
share_amount = 10
period = 20
deviation = 1.5
rsi_upper = 70
rsi_lower = 30
```

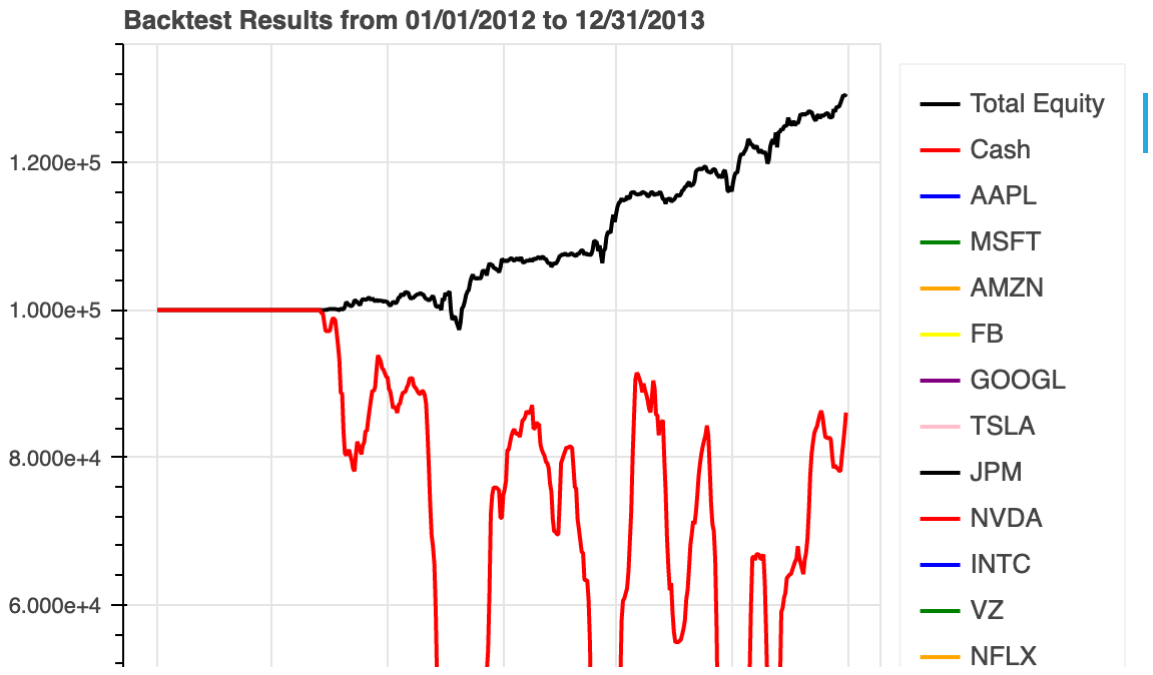
In [3]:
```
class Strategy(bt.Strategy):

    def __init__(self):
        self.rsi = []
        self.bb = []

        for d in self.datas:
            self.rsi.append(bt.indicators.RSI_SMA(d, period=period))
            self.bb.append(bt.indicators.BollingerBands(d, period=period, devfac
```

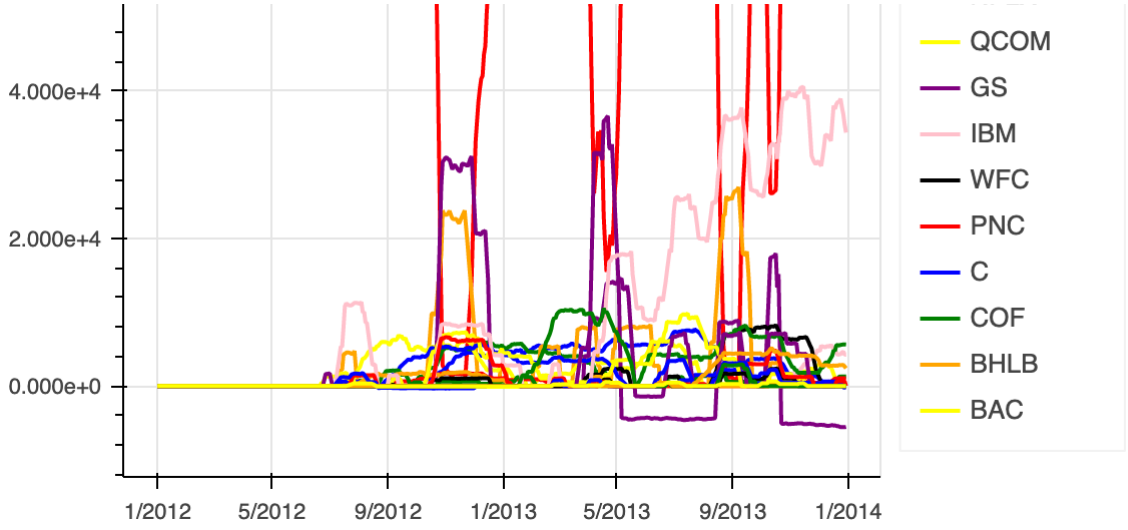
```
    def next(self):
        for i, d in enumerate(self.datas):
            if self.rsi[i] < rsi_lower and self.getposition(d).size < max_shares
                self.buy(data=d, size=share_amount)
            elif self.rsi[i] > rsi_upper and self.getposition(d).size > 0:
                self.sell(data=d, size=share_amount)

            if self.bb[i].lines.top < d.close and self.getposition(d).size > 0:
                self.sell(data=d, size=share_amount)
            elif self.bb[i].lines.bot > d.close and self.getposition(d).size < m
                self.buy(data=d, size=share_amount)
```

In [4]:

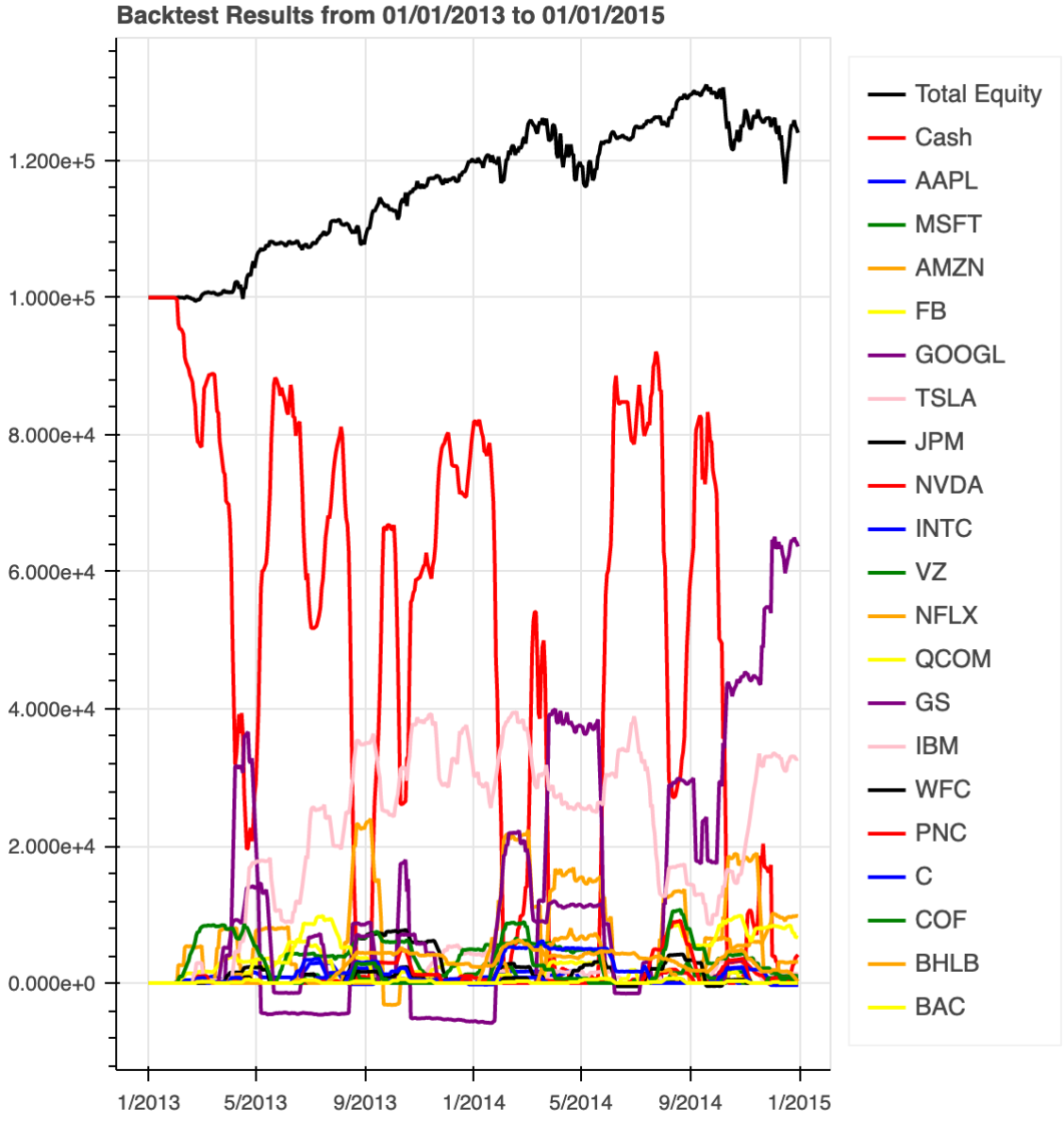
```
###### Backtesting ######

duration_in_days = 365 * 2

for from_year in range(2012, 2019):

    ### Setup for each backtest ###

    from_date = datetime(from_year, 1, 1)
    to_date = from_date + timedelta(days=duration_in_days)

    cerebro = bt.Cerebro()

    for s in stocks:
        data = bt.feeds.YahooFinanceData(dataname=s, fromdate=from_date, todate=
        cerebro.adddata(data, name=s)

    cerebro.addstrategy(Strategy)

    cerebro.broker.setcash(100000.0)

    cerebro.addanalyzer(btanalyzers.SharpeRatio, _name="sharpe")
    cerebro.addanalyzer(btanalyzers.Returns,     _name="returns")
    cerebro.addanalyzer(btanalyzers.Transactions, _name="trans")
    cerebro.addanalyzer(btanalyzers.PositionsValue, cash=True, _name="pval")


    ### Run the current backtest ###

    back = cerebro.run()

    final_portfolio_value = cerebro.broker.getvalue()
    avg_yearly_return = back[0].analyzers.returns.get_analysis()['rnorm100']
    sharpe_ratio = back[0].analyzers.sharpe.get_analysis()['sharperatio']
    num_trans = len(back[0].analyzers.trans.get_analysis())
    cerebro.broker.getvalue()
    pval = back[0].analyzers.pval.get_analysis()
    # positions = cerebro.broker.getposition(data)


    ### Plot the results ###
    from_date_str = from_date.strftime("%m/%d/%Y")
    to_date_str = to_date.strftime("%m/%d/%Y")
    graph = figure(title=f"Backtest Results from {from_date_str} to {to_date_str

    num_values = len([pos[0] for pos in pval.values()])
```

```
    time_diff = timedelta(days=duration_in_days / num_values)
    x_labels = [from_date + time_diff * i for i in range(num_values)]
    colors = ['black', 'red', 'blue', 'green', 'orange', 'yellow', 'purple', 'pi

    legend_it = []

    # Plot total equity
    c = graph.line(x_labels, [sum(pos) for pos in pval.values()], color=colors[0
    legend_it.append(("Total Equity", [c]))

    # Plot cash
    index = len(stocks)
    c = graph.line(x_labels, [pos[index] for pos in pval.values()], color=colors
    legend_it.append(("Cash", [c]))

    # Plot each individual equity
    color_index = 2
    num_colors = len(colors)
    for i in range(len(stocks)):
        stock_data = [pos[i] for pos in pval.values()]
        c = graph.line(x_labels, stock_data, color=colors[color_index % num_colo
        color_index += 1
        legend_it.append((stocks[i], [c]))

    # Add legend
    legend = Legend(items=legend_it)
    graph.add_layout(legend, 'right')
    show(graph)


    ### Print overall stats ###

    from_display = from_date.strftime("%b %Y")
    to_display = to_date.strftime("%b %Y")
    print(f"Results for year {from_display} to {to_display}")
    print(f'Final Portfolio Value: {final_portfolio_value}')
    print(f'Average Yearly Returns: {avg_yearly_return}')
    print(f'Sharpe Ratio: {sharpe_ratio}')
    print(f'Number of Transactions: {num_trans}')
```

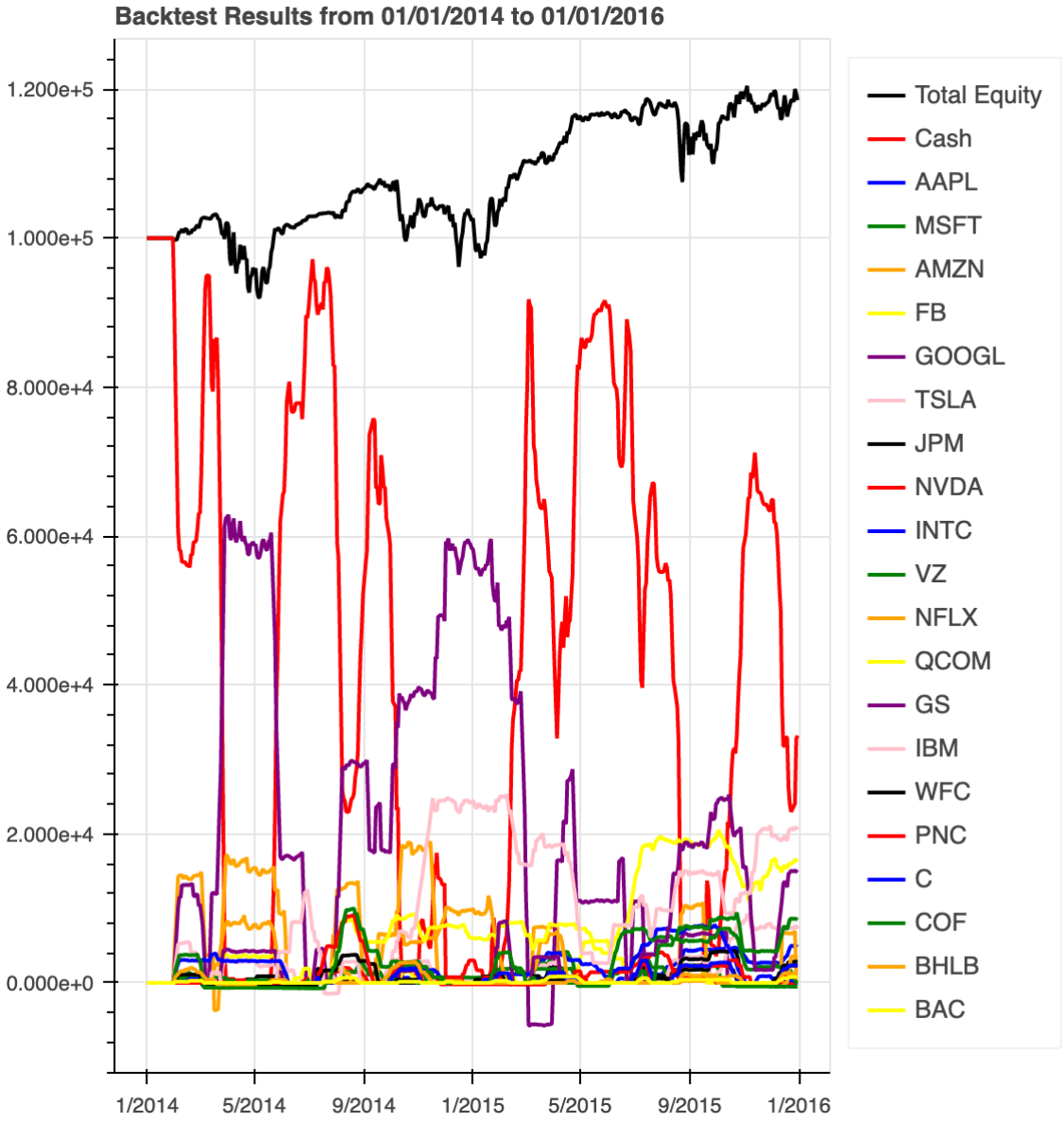




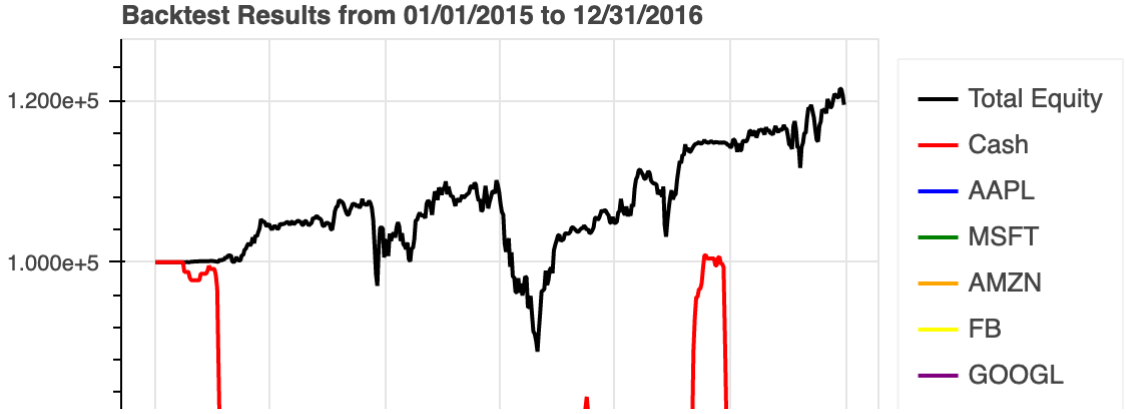

```
Results for year Jan 2012 to Dec 2013
Final Portfolio Value: 129349.39999999988
Average Yearly Returns: 13.819541885963899
Sharpe Ratio: 1.581548699083175
Number of Transactions: 355
```

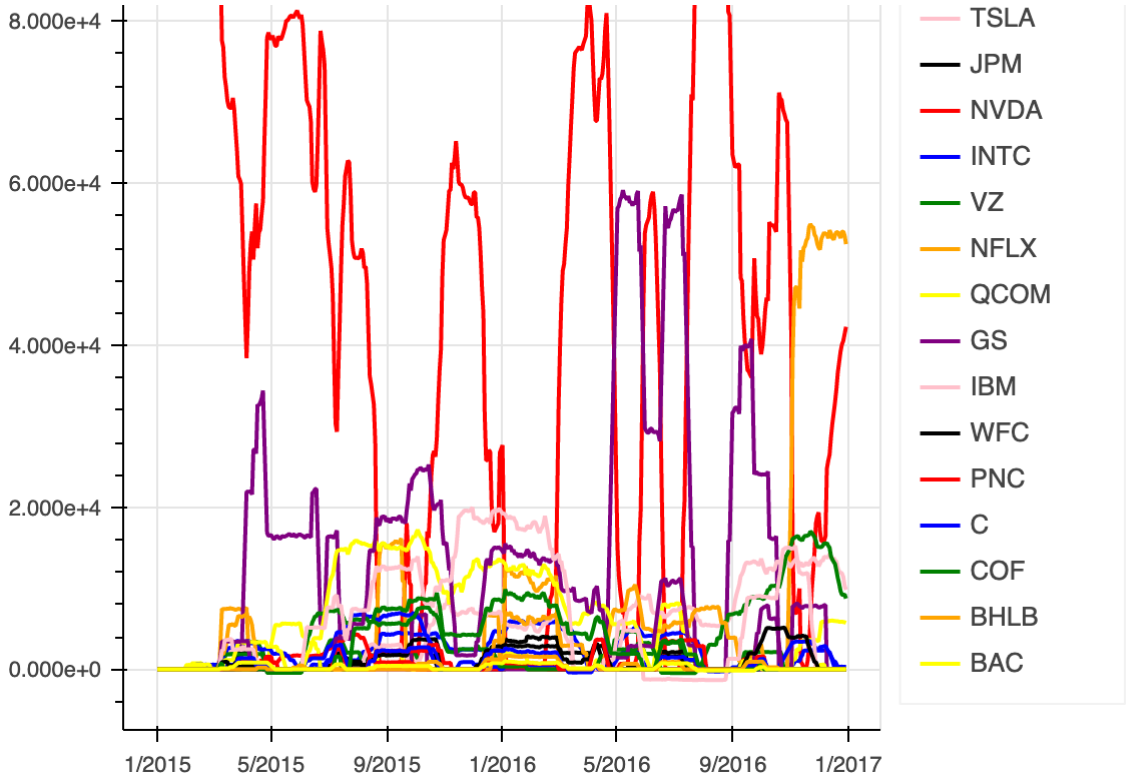


```
Results for year Jan 2013 to Jan 2015
Final Portfolio Value: 124027.09999999999
Average Yearly Returns: 11.36745485104703
Sharpe Ratio: 1.2616523166398026
Number of Transactions: 430
```

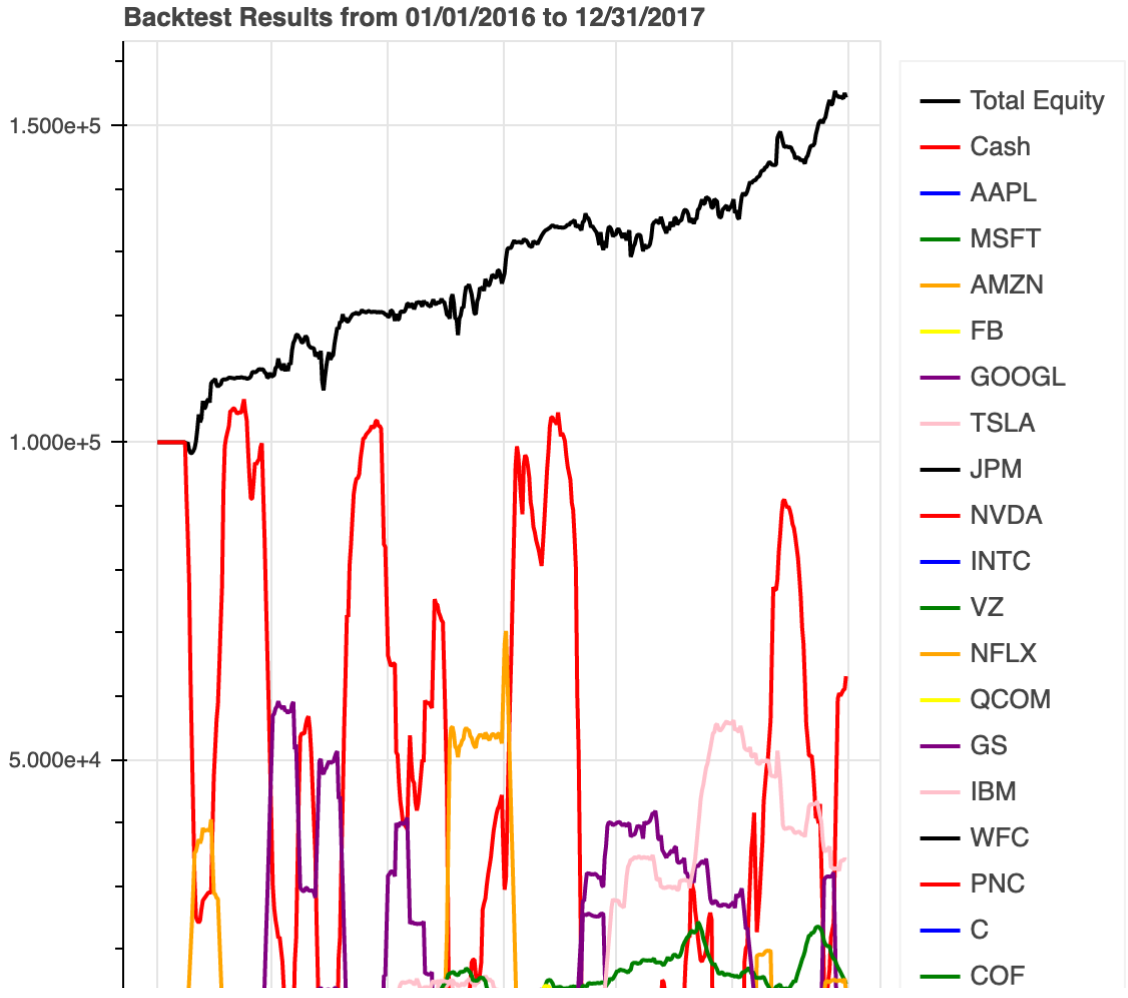


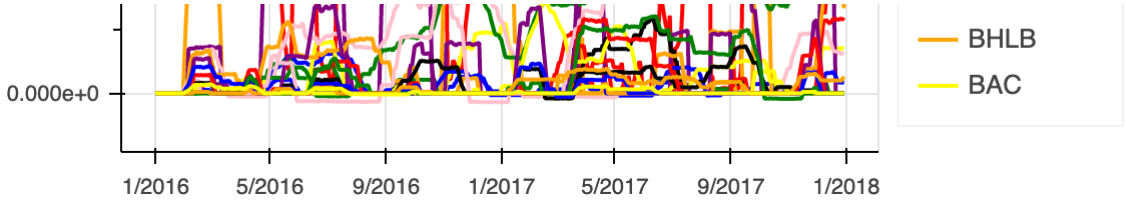

```
Results for year Jan 2014 to Jan 2016
Final Portfolio Value: 118535.7000000001
Average Yearly Returns: 8.874101603641298
Sharpe Ratio: 1.2022309705420402
Number of Transactions: 384
```



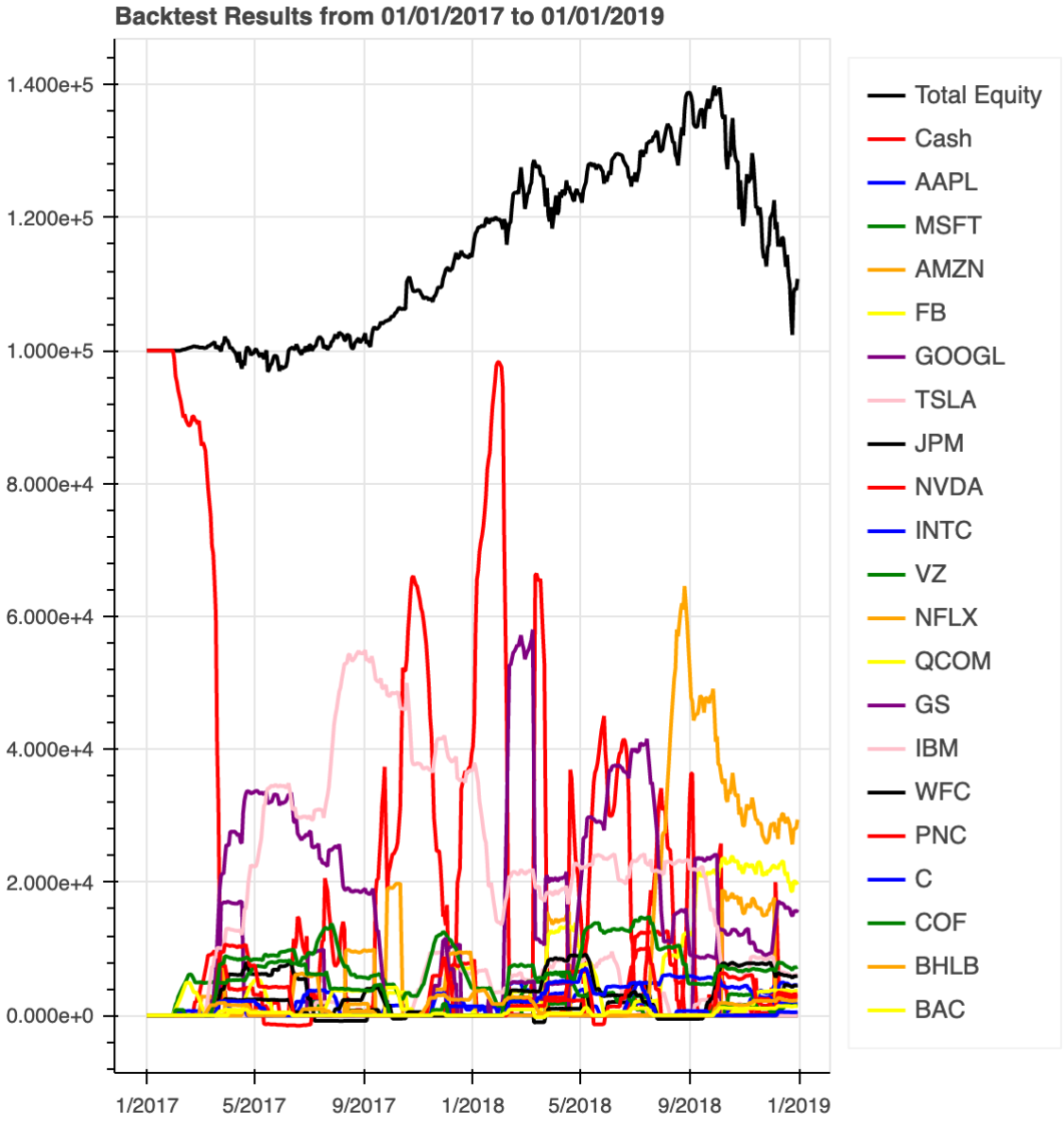



```
Results for year Jan 2015 to Dec 2016
Final Portfolio Value: 119430.30000000005
Average Yearly Returns: 9.284170857448544
Sharpe Ratio: 13.433457821133763
Number of Transactions: 392
```

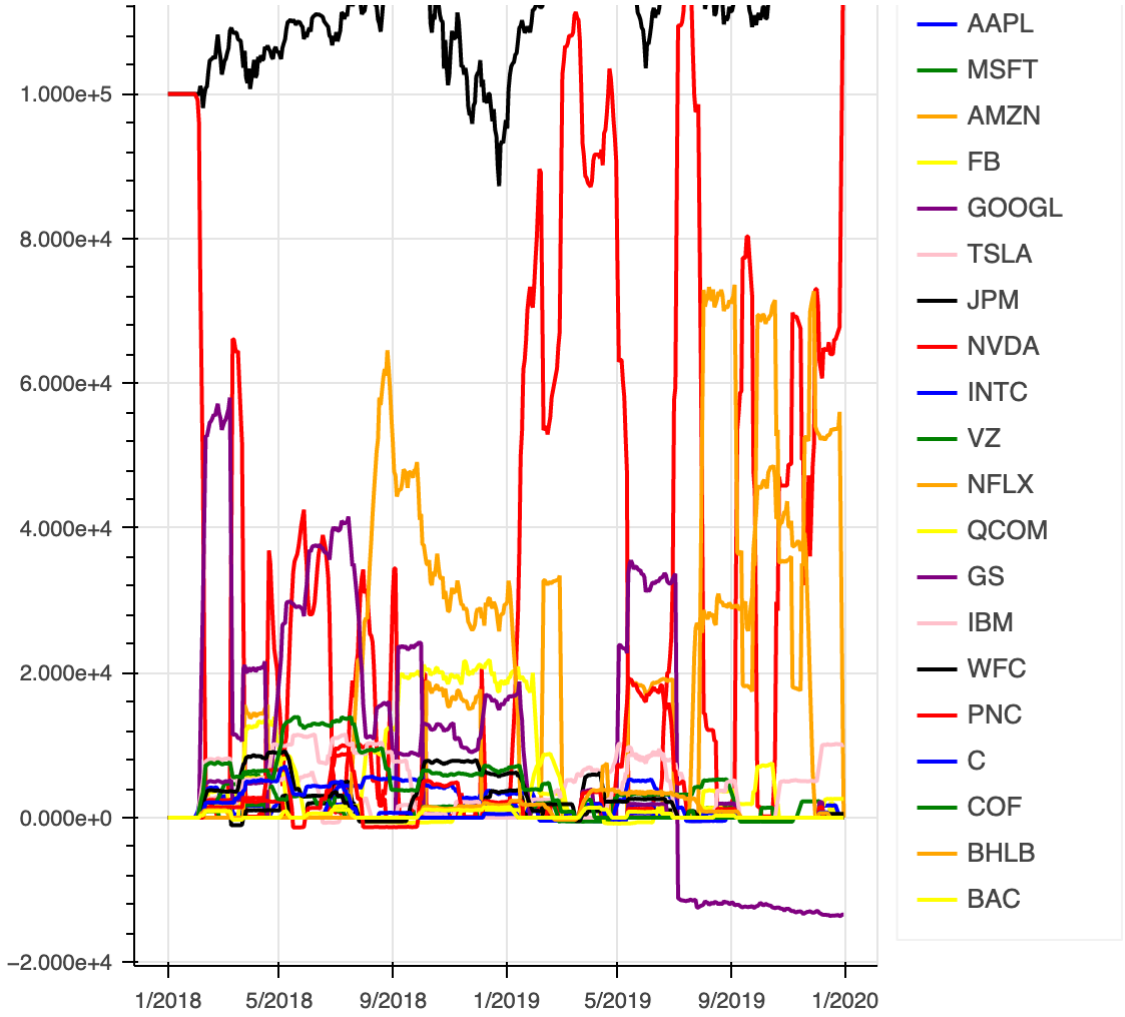





```
Results for year Jan 2016 to Dec 2017
Final Portfolio Value: 154373.6000000001
Average Yearly Returns: 24.300811618625165
Sharpe Ratio: 29.58193027579444
Number of Transactions: 437
```




```
Results for year Jan 2017 to Jan 2019
Final Portfolio Value: 110796.9000000001
Average Yearly Returns: 5.281606950007551
Sharpe Ratio: 0.5430183708023651
Number of Transactions: 381
```




```
Results for year Jan 2018 to Jan 2020
Final Portfolio Value: 123518.60000000021
Average Yearly Returns: 11.1622594525267
Sharpe Ratio: 0.6509410011384192
Number of Transactions: 346
```