Backtesting a Long Only, Momentum Based Trading Algorithm

Introduction

The following code can be thought of in 4 major parts

> 1. Data Collection and Creation of Strategy
> 2. Backtesting of Strategy
> 3. Optimization of Input Parameters
> 4. Out of Sample Testing

Strategy: On any given rebalancing day, the SP100 is ranked by the algorithm (top 50% filtered out based on RSI and MFI values, subsequent stocks are ranked based on their EMA, SMA, and MACD scores). The top 10 stocks are purchased and held until the next rebalancing date.

Train/Test: I trained/optimized the input parameters between '2013-05-24' and '2018-06-28'. Then, during the testing part, I did a historical test (2011-2013) as well as a 'future' test (2018-2020) to see how the strategy performed out of sample.

Summary: Overall, I am quite satisfied with the results of my strategy, both in the training and testing period. The algorithm consistently outperformed the SP100 during its training period, and both testing periods. Furthermore, risk-adjusted metrics such as Sharpe and Sortino ratio also outperformed the SP100 (represented by benchmark 'OEX'). For more analysis on performance, see the testing and conclusion.

Looking Ahead: Having succesfully tested this strategy, I am eager to see how it performs on live market data. I have created an implementation of this strategy on my paper trading account with Interactive Brokers. I hope to monitor its progress and eventually use live trading. Lets see if this strategy holds up in real time!

Part 1: Data Collection and Creation of Strategy

Basic Imports

In [1]:

```
import numpy as np
import pandas as pd
import talib
import datetime
import yfinance as yf
```

Get list of S&P 100

In [2]:

```
wiki = pd.read_html('https://en.wikipedia.org/wiki/S%26P_100')
SP100 = wiki[2]['Symbol'].values.tolist()

#Temporary Fix to tweak Berkshire Hathaway Name and remove Dow Inc.
SP100[18] = 'BRK-B'

#Remove these three stock for training period
SP100.remove('DOW')
SP100.remove('PYPL')
SP100.remove('KHC')

#Remove these additional stock for the historical testing period
#SP100.remove('ABBV')
#SP100.remove('FB')
#SP100.remove('KMI')
#SP100.remove('GM')
```

Variables

In [3]:

```
STOCKS = SP100[:]
BENCHMARK = '^OEX'
#STOCKS = STI30
#BENCHMARK = '^STI'

UNIVERSE_SIZE = len(STOCKS)

#Dataset Start and End Dates
#start = '2013-01-01'
start = '2010-01-01'
end = '2020-06-30'

#Training Set Start and End Dates
training_start = '2013-05-24'
training_end = '2018-06-28'

#Technicals Creator Variables
EMAduration = 10
SMAduration = 100
RSIduration = 14
MACDfast= 12
MACDslow= 26
MACDsignal= 9
MFIduration = 14
RSIcutoff = 20 # distance from 50 (e.g. for cutoff 20, RSI range is 30-70)
MFIcutoff = 30 # distance from 50 (e.g. for cutoff 30, MFI range is 30-70)

start_index= max(EMAduration,SMAduration,RSIduration,MACDfast,MACDslow,MACDsigna
l,MFIduration,RSIcutoff,MFIcutoff)
```

Download Data

In [4]:

```
universe = yf.download(STOCKS, start, end) #,auto_adjust=True)
benchmark = yf.download(BENCHMARK, start, end) # Chicago Options index of SP100
```

```
[*********************100%***********************]  98 of 98 complet
ed
[*********************100%***********************]  1 of 1 completed
```

Basic Data Cleaning

Check for NaNs / Correct Them

In [5]:

```
universe.isna().sum().sum()
```

Out[5]:

```
12342
```

In [6]:

```
#Look at which Stocks had NaNs
universe['Adj Close'].isna().sum().sort_values(ascending = False).head(15)
```

Out[6]:

```
ABBV    757
FB      602
KMI     283
GM      225
CHTR      4
XOM       2
DUK       2
CRM       2
CSCO      2
CVS       2
CVX       2
DD        2
DHR       2
DIS       2
F         2
dtype: int64
```

In [7]:

```
 universe = universe.fillna(method='bfill')
```

In [8]:

```
#Check that backfill method worked
universe.isna().sum().sum()
```

Out[8]:

```
0
```

In [9]:

```
close = universe['Adj Close']
```

Create Indicators

EMA: Exponential Moving Average

In [10]:

```
ema = close.apply(lambda c: talib.EMA(c, EMAduration))
```

SMA: Simple Moving Average

In [11]:

```
sma = close.apply(lambda c: talib.SMA(c, SMAduration))
```

RSI: Relative Strength Index

In [12]:

```
rsi = close.apply(lambda c: talib.RSI(c, RSIduration))
```

MACD: Moving Average Convergence Divergence

In [13]:

```
#MACD is the Moving Average Convergence Divergence. MACD Signal is a n day EMA o
f the MACD.
#The difference between MACD and MACD Signal can be used as signal, therefore, n
eed to track both

macd = pd.DataFrame()
macdsignal = pd.DataFrame()
for stock in close:
    tmacd, tmacdsignal, tmacdhist = talib.MACD(close[stock] , MACDfast, MACDslow
, MACDsignal)
    macd[stock] = tmacd
    macdsignal[stock] = tmacdsignal
```

MFI: Money Flow Index

In [14]:

```
mfi = pd.DataFrame()
for stock in close:
    tmfi = talib.MFI(universe['High'][stock], universe['Low'][stock], close[stoc
k], universe['Volume'][stock], timeperiod = MFIduration)
    mfi[stock] = tmfi
```

Normalize Indicators for Ranking Stock

EMA Ranking

EMA Ranking Overview:

```
Buy --> When Close Price is Greater than EMA
Sell --> When Close Price is Less than EMA

Normalize: The percentage difference between Close Price and EMA
```

In [15]:

```
emasignal = (close-ema)/close*100
```

SMA Ranking

SMA Ranking Overview:

```
Buy --> When EMA is Greater than SMA
Sell --> When EMA is Less than SMA

Normalize: The percentage difference between EMA and SMA
```

In [16]:

```
smasignal = (ema - sma)/sma*100
```

RSI Ranking

RSI Ranking Overview:

```
Buy --> When RSI is <30
Sell --> When RSI is >70

Normalize: RSI is already normalized because it is a stock's strength RELA
TIVE to itself. For now we will process RSI into a score with the followin
g criteria:

        30-70 will be a straight 0
        The buy and sell ranges will be -30 to +30
```

In [17]:

```
def rsicheck(c):
    if c > (50+RSIcutoff): # Check if RSI indicates overbought
        return -(c-50-RSIcutoff)   #returns a negative value range -30 to 0 indi
cating sell ()
    elif c < (50-RSIcutoff): # Check if RSI indicates oversold
        return -(c-50+RSIcutoff) #returns a positive value range 0 to 30 indicat
ing buy
    else:
        return 0 # Do not use RSI as an indicator within 50 +- cutoff

rsisignal = rsi

for stock in rsisignal:
    rsisignal[stock] = rsisignal[stock].apply(lambda c: rsicheck(c))
```

MACD Ranking

MACD Ranking Overview:

```
Explanation:
1. The MACD is taken by subtracting the long EMA from short EMA
2. The MACD Signal (named MACDsignal) is an EMA of the MACD

Buy --> When MACD > MACD Signal
Sell --> When MACD is < MACD Signal

Normalize: Percent difference between MACD and MACD Signal
```

In [18]:

```
macdscore = (macd-macdsignal)/macdsignal*100
```

MFI Ranking

MFI Ranking Overview:

```
Buy --> When MFI <20
Sell --> When MFI is >80

Normalize: MFI should already be normalized
```

In [19]:

```
def mficheck(c):
    if c > (50+MFIcutoff): # Check if MFI indicates overbought
        return -(c-50-MFIcutoff)   #returns a negative value range -20 to 0 indi
cating sell ()
    elif c < (50-MFIcutoff): # Check if MFI indicates oversold
        return -(c-50+MFIcutoff) #returns a positive value range 0 to 20 indicat
ing buy
    else:
        return 0 # Do not use MFI as an indicator within 50 +- cutoff

mfisignal = mfi

for stock in mfisignal:
    mfisignal[stock] = mfisignal[stock].apply(lambda c: mficheck(c))
```

Part 2: Backtesting of Strategy

Object Oriented Backtester Class

In [20]:

```
class Backtester():
    def __init__(self, initial_capital = 50000):
        self.initial_capital = initial_capital
        self.current_balance = initial_capital


        #Create a Portfolio that will record positions held
        self.portfolio = pd.DataFrame(columns= STOCKS)
        self.portfolio.insert(0, 'Index', ['Quantity','Entry'])
        self.portfolio.set_index('Index',inplace = True)
        self.portfolio[:] = 0

    #Function to return top scored stocks on a given date.
    def selector(self, date, is_long = True):

        #STEP 1: Filter half of universe by equally weighting MFI and RSI (e.g.
 Identify Oversold Stock)
        score1 = mfisignal+rsisignal
        step1 = score1.loc[date].sort_values(ascending = not is_long)
        step1 = step1.iloc[:UNIVERSE_SIZE//2]
        filteredList = step1.index

        #STEP 2: Rank remaining stocks based on equal weightage of EMA, SMA, and
MACD
        score2 = self.ema_weight*emasignal + self.sma_weight*smasignal + self.ma
cd_weight*macdscore
        step2 = score2[filteredList].loc[date].sort_values(ascending= not is_lon
g)
        return(step2[:self.basket_size])

    #Function to Purchase a Basket of Stocks given a Date
    def purchaser(self, date, overlap_stock, overlap_value):
        target = self.selector(date)

        #Ignore the overlap_stock as we already hold positions in them
        part = self.current_balance / (self.basket_size - len(overlap_stock))
        for stock in target.index:
            #Only purchase NEW stock (ignore overlapped stock)
            if stock not in overlap_stock:
                self.portfolio[stock]['Entry'] = close[stock][date]
                self.portfolio[stock]['Quantity'] = part // ((1+self.transaction
_cost)*close[stock][date])
                remainder = part % ((1+self.transaction_cost)*close[stock][date
])
                self.current_balance -= part
                self.current_balance += remainder
            if stock in overlap_stock:
                while(self.portfolio[stock]['Quantity'] * close[stock][date] > p
art):
                    self.current_balance += close[stock][date]*(1+self.transacti
on_cost)
                    self.portfolio[stock]['Quantity'] += 1

    #Function to Fully Liquidate Portfolio
    def liquidate(self, date):
        target = self.selector(date)
        overlap_value = 0
        overlap_stock = []
```

```
        for stock in self.portfolio:
            #If you own the stock and it is not going to be repurchased --> full
y liquidate position
            if self.portfolio[stock]['Quantity'] != 0 and stock not in target.in
dex:
                self.current_balance += (self.portfolio[stock]['Quantity'] *
                                         close[stock][date] * (1-self.transactio
n_cost))
                self.portfolio[stock]['Quantity'] = 0
                self.portfolio[stock]['Entry'] = 0

            #Figure out value of "overlapping shares"
            elif self.portfolio[stock]['Quantity'] != 0:
                overlap_value += (self.portfolio[stock]['Quantity'] * close[stoc
k][date])# * (1-self.transaction_cost))
                overlap_stock.append(stock)

        part = (self.current_balance + overlap_value) // self.basket_size

        for stock in overlap_stock:
            #If your position is greater than its allocated percentage (part), s
ell off the excess
            while(self.portfolio[stock]['Quantity'] * close[stock][date] > part
):
                self.current_balance += close[stock][date]*(1-self.transaction_c
ost)
                self.portfolio[stock]['Quantity'] -= 1

        return overlap_stock, overlap_value

    #Function to fully liquidate a portfolio at the end of simulation
    def close_out(self, date):
         for stock in self.portfolio:
            if self.portfolio[stock]['Quantity'] != 0:
                self.current_balance += (self.portfolio[stock]['Quantity'] *
                                         close[stock][date] * (1-self.transactio
n_cost))
                self.portfolio[stock]['Quantity'] = 0
                self.portfolio[stock]['Entry'] = 0

    #Rebalances Portfolio (calls the liquidate function then purchase function)
    def rebalancer(self, date):
        overlap_stock, overlap_value = self.liquidate(date) # empties portfolio
        self.purchaser(date, overlap_stock, overlap_value) #fills portfolio

    #Records the value of portfolio at a given date
    def recorder(self, date):
        value = self.current_balance
        for stock in self.portfolio:
            value += self.portfolio[stock]['Quantity'] * close[stock][date]
        self.portfolio_value.loc[date] = value

    #Prints the portfolio (excludes stock that have quantity of 0)
    def print_portfolio(self):
        temp_portfolio = pd.DataFrame()
        for stock in self.portfolio:
            if self.portfolio[stock]['Quantity'] != 0:
                temp_portfolio = temp_portfolio.append(self.portfolio[stock])
        print(temp_portfolio)
```

```
    #Runs the simulation
    def simulate(self, sim_start = pd.Timestamp(training_start), sim_end = pd.Ti
mestamp(training_end),
                 rebalance_duration = 5, basket_size = 10, weights = (1/3,1/3,1/
3), transaction_cost = 0.0003):

        #Initialization/UI
        self.basket_size = basket_size
        self.ema_weight = weights[0]
        self.sma_weight = weights[1]
        self.macd_weight = weights[2]
        self.transaction_cost = transaction_cost

        #Create a DataFrame to track historical portfolio value
        self.portfolio_value = pd.Series(name = 'Value', dtype = 'float64')
        print("////////// ----------- //////////")
        self.current_balance = self.initial_capital
        print("Simulation is running from ", sim_start.date(), "until ", sim_end
.date())
        print("Universe Size is: ", UNIVERSE_SIZE)
        print("Basket Size is: ", self.basket_size)
        print("Starting Capital: ", self.initial_capital)
        print("Current Balance: ", self.current_balance)
        print("Rebalance Duration is:", rebalance_duration)
        print("Indicator Weights (EMA, SMA, MACD): ", self.ema_weight, self.sma_
weight, self.macd_weight)
        print("Transaction Costs:", self.transaction_cost)
        #Enter positions on first day of simulation
        self.purchaser(sim_start, [],0)

        print("Initial Portfolio:")
        self.print_portfolio()

        count = 0

        #Loop through days within simulation period
        for day in close.loc[sim_start:sim_end].index:

                #Rebalance on a given interval
                if((count % rebalance_duration) == 0):
                    self.rebalancer(day)

                #Record Value of Portfolio every day
                self.recorder(day)

                #IMPORTANT --> code to increment count if we are trading every _
_ days
                count += 1;

        #Exit all positions at end of simulation
        self.close_out(sim_end)

        #Exit Metrics
        print("Simulation has completed, here are some performance metrics:")


        print("Final Cash Balance = ", self.current_balance)

        performance = (self.current_balance - self.initial_capital) / self.initi
al_capital *100
        print("Performance (%) = ", performance)
```

```
        print("////////// ----------- //////////")


    #To be called after simulate, runs analysis and creates a dataframe called s
elf.analyze which tracks performance˜f
    def analyze_performance(self):
        self.analyze = self.portfolio_value.copy()
        self.analyze = self.analyze.to_frame('Portfolio Value')
        self.analyze['Daily Returns (%)'] = self.analyze['Portfolio Value'].pct_
change(1)
        self.analyze['Cumulative Returns (%)'] = ((self.analyze['Daily Returns
 (%)']+1).cumprod() - 1)
        self.analyze['Benchmark Value'] = benchmark['Adj Close'].copy()
        self.analyze['Benchmark Daily Returns (%)'] = self.analyze['Benchmark Va
lue'].pct_change(1)
        self.analyze['Benchmark Cumulative Returns (%)'] = ((self.analyze['Bench
mark Daily Returns (%)']+1).cumprod() - 1)
        #self.analyze.plot(y={'Cumulative Returns (%)','Benchmark Cumulative Ret
urns (%)'}, figsize=(10,6))
```

In [21]:

```
backtest1 = Backtester()
```

In [22]:

```
backtest1.simulate()
```

```
////////// ----------- //////////
Simulation is running from  2013-05-24 until  2018-06-28
Universe Size is:  98
Basket Size is:  10
Starting Capital:  50000
Current Balance:  50000
Rebalance Duration is: 5
Indicator Weights (EMA, SMA, MACD):  0.3333333333333333 0.3333333333
333333 0.3333333333333333
Transaction Costs: 0.0003
Initial Portfolio:
           Entry  Quantity
ABT    32.626583     153.0
BIIB  235.240005      21.0
BKNG  801.119995       6.0
CSCO   18.742292     266.0
DUK    49.744247     100.0
EXC    26.340830     189.0
FB     24.309999     205.0
GE     18.268507     273.0
IBM   154.403519      32.0
T      24.662882     202.0
```

```
/Users/jaiveerkhanna/opt/anaconda3/lib/python3.7/site-packages/ipyke
rnel_launcher.py:32: RuntimeWarning: divide by zero encountered in d
ouble_scalars
```

```
Simulation has completed, here are some performance metrics:
Final Cash Balance =  109453.10517353936
Performance (%) =  118.90621034707871
////////// ----------- //////////
```

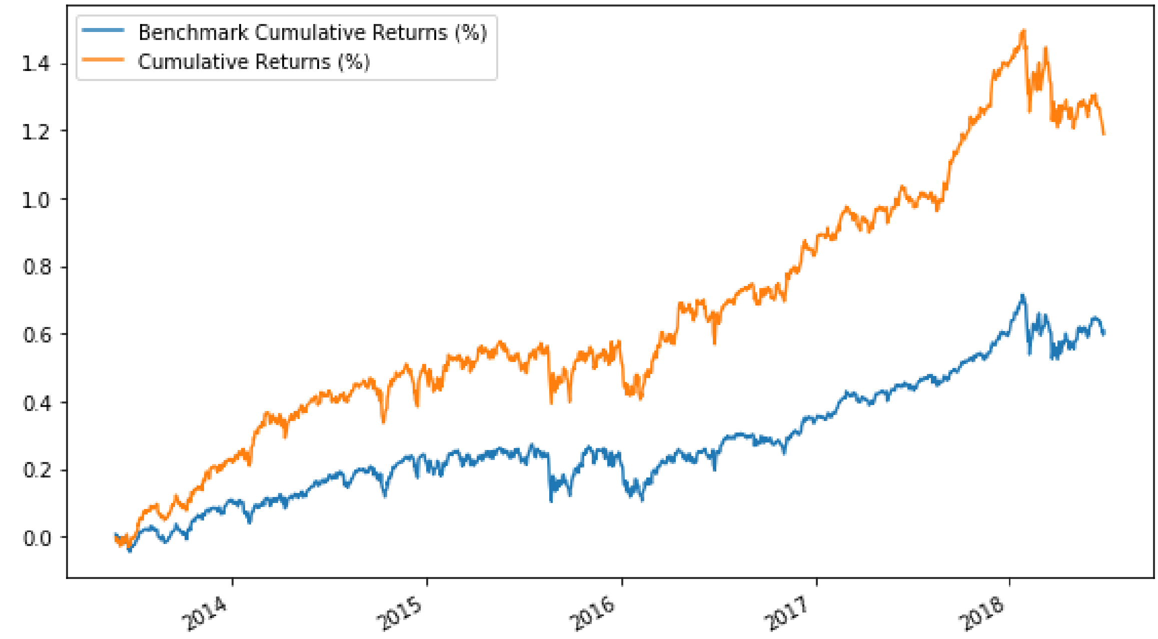
Analyze Performance

In [23]:

```
backtest1.analyze_performance()
backtest1.analyze.tail(5)
backtest1.analyze.plot(y={'Cumulative Returns (%)','Benchmark Cumulative Returns (%)'}, figsize=(10,6))
```

Out[23]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x123c79d90>
```



Sharpe Ratio

Definition: Ratio that shows risk-adjusted return of portfolio

Calculation: (Portfolio Return - Risk Free Rate) / (Portfolio Std. Dev)

In [24]:

```
#Sharpe Ratio Function (default: assumes 0% risk free rate and annualizes sharpe)
def sharpe(returns, risk_free_rate=0, days=252):
    sharpe_ratio = returns.mean()/returns.std() * np.sqrt(days)
    return sharpe_ratio
```

Sortino Ratio

Definition: Ratio that shows risk-adjusted return of portfolio (using only negative volatility) --> Doesn't penalize a stock for increases in price (which would affect standard deviation)

Calculation: (Portfolio Return - Risk Free Rate) / (Portfolio Std. Dev [Negative Values])

Parity Check: Sortino Ratio should be higher than Sharpe Ratio since its only considering the volatility of negative returns

In [25]:

```
def sortino(returns, risk_free_rate=0, days=252):
    neg_returns = returns[returns < 0]
    sortino_ratio = returns.mean()/neg_returns.std() * np.sqrt(days)
    return sortino_ratio
```

Part 3: Optimization of Input Parameters

Optimize

Step 1: Optimize by basket size

In [26]:

```
optimize = Backtester()
```

In [27]:

```
basket = pd.DataFrame()
basket_tracker = 5
while(basket_tracker < 25):
    optimize.simulate(basket_size = basket_tracker)
    optimize.analyze_performance()
    basket[optimize.basket_size] = optimize.analyze['Cumulative Returns (%)']
    basket_tracker += 5
```

```
////////// ----------- //////////
Simulation is running from  2013-05-24 until  2018-06-28
Universe Size is:  98
Basket Size is:  5
Starting Capital:  50000
Current Balance:  50000
Rebalance Duration is: 5
Indicator Weights (EMA, SMA, MACD):  0.3333333333333333 0.3333333333
333333 0.3333333333333333
Transaction Costs: 0.0003
Initial Portfolio:
          Entry  Quantity
DUK   49.744247     200.0
EXC   26.340830     379.0
FB    24.309999     411.0
IBM  154.403519      64.0
T     24.662882     405.0
```

```
/Users/jaiveerkhanna/opt/anaconda3/lib/python3.7/site-packages/ipyke
rnel_launcher.py:32: RuntimeWarning: divide by zero encountered in d
ouble_scalars
```

```
Simulation has completed, here are some performance metrics:
Final Cash Balance =  102183.28043147697
Performance (%) =  104.36656086295395
////////// ----------- //////////
////////// ----------- //////////
Simulation is running from  2013-05-24 until  2018-06-28
Universe Size is:  98
Basket Size is:  10
Starting Capital:  50000
Current Balance:  50000
Rebalance Duration is: 5
Indicator Weights (EMA, SMA, MACD):  0.3333333333333333 0.3333333333
333333 0.3333333333333333
Transaction Costs: 0.0003
Initial Portfolio:
           Entry  Quantity
ABT    32.626583     153.0
BIIB  235.240005      21.0
BKNG  801.119995       6.0
CSCO   18.742292     266.0
DUK    49.744247     100.0
EXC    26.340830     189.0
FB     24.309999     205.0
GE     18.268507     273.0
IBM   154.403519      32.0
T      24.662882     202.0
```

```
/Users/jaiveerkhanna/opt/anaconda3/lib/python3.7/site-packages/ipyke
rnel_launcher.py:32: RuntimeWarning: divide by zero encountered in d
ouble_scalars
```

```
Simulation has completed, here are some performance metrics:
Final Cash Balance =  109453.10517353936
Performance (%) =  118.90621034707871
////////// ----------- //////////
////////// ----------- //////////
Simulation is running from  2013-05-24 until  2018-06-28
Universe Size is:  98
Basket Size is:  15
Starting Capital:  50000
Current Balance:  50000
Rebalance Duration is: 5
Indicator Weights (EMA, SMA, MACD):  0.3333333333333333 0.3333333333
333333 0.3333333333333333
Transaction Costs: 0.0003
Initial Portfolio:
           Entry  Quantity
ABT    32.626583     102.0
BIIB  235.240005      14.0
BKNG  801.119995       4.0
COST   98.545853      33.0
CSCO   18.742292     177.0
CVX    93.243530      35.0
DUK    49.744247      66.0
EXC    26.340830     126.0
F      10.765032     309.0
FB     24.309999     137.0
GE     18.268507     182.0
GILD   48.009838      69.0
IBM   154.403519      21.0
PFE    22.383799     148.0
T      24.662882     135.0

/Users/jaiveerkhanna/opt/anaconda3/lib/python3.7/site-packages/ipyke
rnel_launcher.py:32: RuntimeWarning: divide by zero encountered in d
ouble_scalars
```

```
Simulation has completed, here are some performance metrics:
Final Cash Balance =  106112.55269011282
Performance (%) =  112.22510538022563
////////// ----------- //////////
////////// ----------- //////////
Simulation is running from  2013-05-24 until  2018-06-28
Universe Size is:  98
Basket Size is:  20
Starting Capital:  50000
Current Balance:  50000
Rebalance Duration is: 5
Indicator Weights (EMA, SMA, MACD):  0.3333333333333333 0.3333333333
333333 0.3333333333333333
Transaction Costs: 0.0003
Initial Portfolio:
           Entry  Quantity
ABT    32.626583      76.0
ACN    70.534546      35.0
BIIB  235.240005      10.0
BKNG  801.119995       3.0
C      45.486343      54.0
COF    52.724289      47.0
COST   98.545853      25.0
CSCO   18.742292     133.0
CVX    93.243530      26.0
DUK    49.744247      50.0
EXC    26.340830      94.0
F      10.765032     232.0
FB     24.309999     102.0
GE     18.268507     136.0
GILD   48.009838      52.0
HON    64.634583      38.0
IBM   154.403519      16.0
PFE    22.383799     111.0
T      24.662882     101.0
XOM    68.638519      36.0

/Users/jaiveerkhanna/opt/anaconda3/lib/python3.7/site-packages/ipyke
rnel_launcher.py:32: RuntimeWarning: divide by zero encountered in d
ouble_scalars

Simulation has completed, here are some performance metrics:
Final Cash Balance =  100827.2529460118
Performance (%) =  101.65450589202358
////////// ----------- //////////
```

In [28]:

```
basket['Benchmark'] = optimize.analyze['Benchmark Cumulative Returns (%)']
```

In [29]:

```
#Displays the Cumulative Returns for all basket sizes and benchmark
basket.iloc[-1]
```

Out[29]:

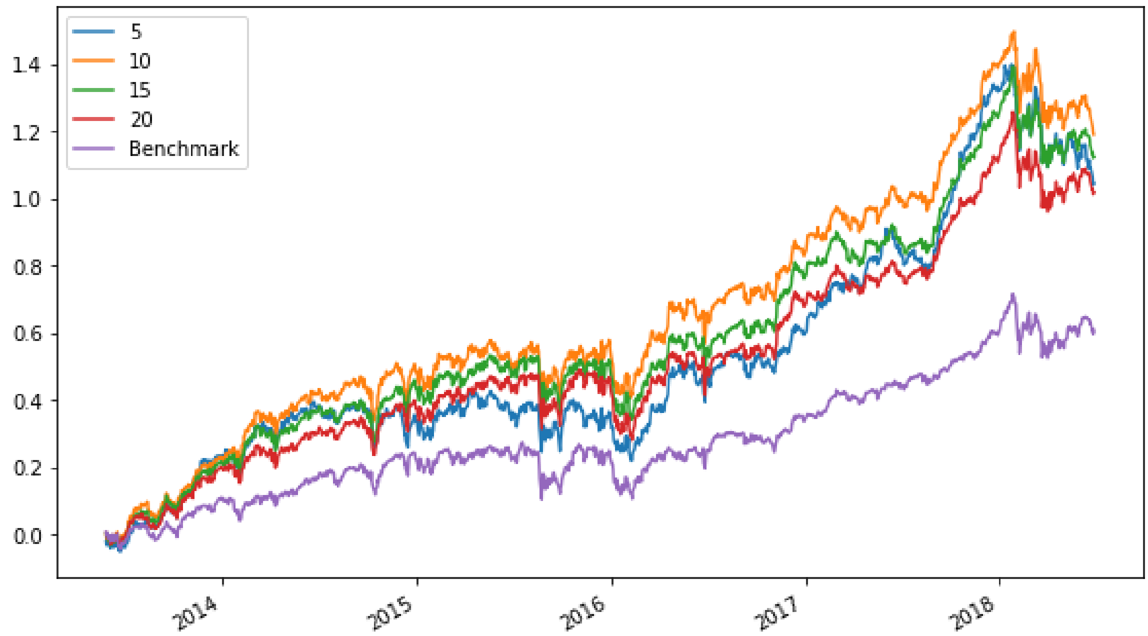
```
5            1.044889
10           1.190368
15           1.123515
20           1.017744
Benchmark    0.608523
Name: 2018-06-28 00:00:00, dtype: float64
```

In [30]:

```
basket.plot(figsize=(10,6))
```

Out[30]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x120ad4bd0>
```



Step 2: Check that each year produces returns (make sure one year isnt abnormally affecting total return)

In [31]:

```
optimize = Backtester()
```

In [32]:

```
sim_start = datetime.datetime.strptime('2013-05-24', '%Y-%m-%d')
sim_end = sim_start + datetime.timedelta(days=364)
optimize.basket_size = 10
years = pd.DataFrame()
years.insert(0,'Index', ['Annual Return', 'Benchmark Annual Return', 'Sharpe','B
enchmark Sharpe', 'Sortino', 'Benchmark Sortino'])
years.set_index('Index',inplace = True)

while(sim_end < pd.Timestamp(training_end)):
    optimize.simulate(sim_start,sim_end)
    optimize.analyze_performance()
    analysis = optimize.analyze
    years[sim_end] = analysis['Cumulative Returns (%)'].iloc[-1]
    years[sim_end].loc['Sharpe'] = sharpe(analysis['Daily Returns (%)'])
    years[sim_end].loc['Sortino'] = sortino(analysis['Daily Returns (%)'])
    years[sim_end].loc['Benchmark Annual Return'] = analysis['Benchmark Cumulati
ve Returns (%)'].iloc[-1]
    years[sim_end].loc['Benchmark Sharpe'] = sharpe(analysis['Benchmark Daily Re
turns (%)'])
    years[sim_end].loc['Benchmark Sortino'] = sortino(analysis['Benchmark Daily
 Returns (%)'])
    sim_start = sim_end
    sim_end += datetime.timedelta(days=356)
```

```
////////// ----------- //////////
Simulation is running from  2013-05-24 until  2014-05-23
Universe Size is:  98
Basket Size is:  10
Starting Capital:  50000
Current Balance:  50000
Rebalance Duration is: 5
Indicator Weights (EMA, SMA, MACD):  0.3333333333333333 0.3333333333
333333 0.3333333333333333
Transaction Costs: 0.0003
Initial Portfolio:
           Entry  Quantity
ABT    32.626583     153.0
BIIB  235.240005      21.0
BKNG  801.119995       6.0
CSCO   18.742292     266.0
DUK    49.744247     100.0
EXC    26.340830     189.0
FB     24.309999     205.0
GE     18.268507     273.0
IBM   154.403519      32.0
T      24.662882     202.0
```

```
/Users/jaiveerkhanna/opt/anaconda3/lib/python3.7/site-packages/ipyke
rnel_launcher.py:32: RuntimeWarning: divide by zero encountered in d
ouble_scalars
```

```
Simulation has completed, here are some performance metrics:
Final Cash Balance =  69473.96913079817
Performance (%) =  38.94793826159634
////////// ----------- //////////
////////// ----------- //////////
Simulation is running from  2014-05-23 until  2015-05-14
Universe Size is:  98
Basket Size is:  10
Starting Capital:  50000
Current Balance:  50000
Rebalance Duration is: 5
Indicator Weights (EMA, SMA, MACD):  0.3333333333333333 0.3333333333
333333 0.3333333333333333
Transaction Costs: 0.0003
Initial Portfolio:
           Entry  Quantity
BMY    41.081051     121.0
CMCSA  23.114378     216.0
CSCO   20.133902     248.0
DD     58.512604      85.0
DHR    39.007969     128.0
DIS    76.680748      65.0
DUK    53.035511      94.0
F      11.983428     417.0
GM     25.955523     192.0
HON    76.545982      65.0
```

```
/Users/jaiveerkhanna/opt/anaconda3/lib/python3.7/site-packages/ipyke
rnel_launcher.py:32: RuntimeWarning: divide by zero encountered in d
ouble_scalars
```

```
Simulation has completed, here are some performance metrics:
Final Cash Balance =  56519.38516823365
Performance (%) =  13.038770336467307
////////// ----------- //////////
////////// ----------- //////////
Simulation is running from  2015-05-14 until  2016-05-04
Universe Size is:  98
Basket Size is:  10
Starting Capital:  50000
Current Balance:  50000
Rebalance Duration is: 5
Indicator Weights (EMA, SMA, MACD):  0.3333333333333333 0.3333333333
333333 0.3333333333333333
Transaction Costs: 0.0003
Initial Portfolio:
            Entry  Quantity
ABT     43.804420     114.0
ADBE    79.430000      62.0
AMGN   140.434067      35.0
BMY     58.315853      85.0
BRK-B  145.779999      34.0
CHTR   176.789993      28.0
CMCSA   25.571249     195.0
DUK     59.341072      84.0
FDX    163.147812      30.0
GILD    93.542175      53.0
```

```
/Users/jaiveerkhanna/opt/anaconda3/lib/python3.7/site-packages/ipyke
rnel_launcher.py:32: RuntimeWarning: divide by zero encountered in d
ouble_scalars
```

```
Simulation has completed, here are some performance metrics:
Final Cash Balance =  50563.80626865874
Performance (%) =  1.1276125373174728
////////// ----------- //////////
////////// ----------- //////////
Simulation is running from  2016-05-04 until  2017-04-25
Universe Size is:  98
Basket Size is:  10
Starting Capital:  50000
Current Balance:  50000
Rebalance Duration is: 5
Indicator Weights (EMA, SMA, MACD):  0.3333333333333333 0.3333333333
333333 0.3333333333333333
Transaction Costs: 0.0003
Initial Portfolio:
            Entry  Quantity
AAPL    21.972309     227.0
ALL     59.852459      83.0
AMZN   670.900024       7.0
CHTR   212.429993      23.0
CL      65.037239      76.0
FB     118.059998      42.0
GD     129.285248      38.0
GOOG   695.700012       7.0
GOOGL  711.369995       7.0
NFLX    90.790001      55.0
```

```
/Users/jaiveerkhanna/opt/anaconda3/lib/python3.7/site-packages/ipyke
rnel_launcher.py:32: RuntimeWarning: divide by zero encountered in d
ouble_scalars
```

```
Simulation has completed, here are some performance metrics:
Final Cash Balance =  59978.95749616403
Performance (%) =  19.957914992328064
////////// ----------- //////////
////////// ----------- //////////
Simulation is running from  2017-04-25 until  2018-04-16
Universe Size is:  98
Basket Size is:  10
Starting Capital:  50000
Current Balance:  50000
Rebalance Duration is: 5
Indicator Weights (EMA, SMA, MACD):  0.3333333333333333 0.3333333333
333333 0.3333333333333333
Transaction Costs: 0.0003
Initial Portfolio:
            Entry  Quantity
ALL     75.889183      65.0
CMCSA   36.011585     138.0
DHR     81.781654      61.0
EXC     31.138861     160.0
GD     180.118408      27.0
GE      26.205322     190.0
HON    114.951820      43.0
INTC    33.777348     147.0
JNJ    112.410667      44.0
VZ      40.396355     123.0

/Users/jaiveerkhanna/opt/anaconda3/lib/python3.7/site-packages/ipyke
rnel_launcher.py:32: RuntimeWarning: divide by zero encountered in d
ouble_scalars

Simulation has completed, here are some performance metrics:
Final Cash Balance =  59667.86794953947
Performance (%) =  19.335735899078937
////////// ----------- //////////
```

In [33]:

```
#years will show different performance metrics for each individual year
years
```

Out[33]:

Index	2014-05-23	2015-05-14	2016-05-04	2017-04-25	2018-04-16
Annual Return	0.390309	0.131063	0.011874	0.200292	0.194068
Benchmark Annual Return	0.134924	0.107170	-0.025137	0.160494	0.114108
Sharpe	2.579903	0.988539	0.157047	1.649616	1.478571
Benchmark Sharpe	1.220030	0.940555	-0.069741	1.672225	0.956650
Sortino	3.974340	1.455679	0.226985	2.395735	1.511776
Benchmark Sortino	1.719448	1.367978	-0.099754	2.243504	0.969907

Step 3: Optimize for different rebalance periods

In [34]:

```
optimize = Backtester()
```

In [35]:

```
rebalance_dur = [1, 3, 5, 10, 15, 20, 30]
rebalance = pd.DataFrame()
rebalance.insert(0,'Index', ['Net Return', 'Sharpe', 'Sortino'])
rebalance.set_index('Index',inplace = True)
rebalance_detailed = pd.DataFrame()

for duration in rebalance_dur:
    optimize.simulate(rebalance_duration = duration)
    optimize.analyze_performance()
    analysis = optimize.analyze
    rebalance[duration] = analysis['Cumulative Returns (%)'].iloc[-1]
    rebalance_detailed[duration] = analysis['Cumulative Returns (%)']
    rebalance[duration].loc['Sharpe'] = sharpe(analysis['Daily Returns (%)'])
    rebalance[duration].loc['Sortino'] = sortino(analysis['Daily Returns (%)'])
```

```
////////// ----------- //////////
Simulation is running from  2013-05-24 until  2018-06-28
Universe Size is:  98
Basket Size is:  10
Starting Capital:  50000
Current Balance:  50000
Rebalance Duration is: 1
Indicator Weights (EMA, SMA, MACD):  0.3333333333333333 0.3333333333
333333 0.3333333333333333
Transaction Costs: 0.0003
Initial Portfolio:
           Entry  Quantity
ABT    32.626583     153.0
BIIB  235.240005      21.0
BKNG  801.119995       6.0
CSCO   18.742292     266.0
DUK    49.744247     100.0
EXC    26.340830     189.0
FB     24.309999     205.0
GE     18.268507     273.0
IBM   154.403519      32.0
T      24.662882     202.0
```

```
/Users/jaiveerkhanna/opt/anaconda3/lib/python3.7/site-packages/ipyke
rnel_launcher.py:32: RuntimeWarning: divide by zero encountered in d
ouble_scalars
```

```
Simulation has completed, here are some performance metrics:
Final Cash Balance =  104148.2412070529
Performance (%) =  108.29648241410578
////////// ----------- //////////
////////// ----------- //////////
Simulation is running from  2013-05-24 until  2018-06-28
Universe Size is:  98
Basket Size is:  10
Starting Capital:  50000
Current Balance:  50000
Rebalance Duration is: 3
Indicator Weights (EMA, SMA, MACD):  0.3333333333333333 0.3333333333
333333 0.3333333333333333
Transaction Costs: 0.0003
Initial Portfolio:
           Entry  Quantity
ABT    32.626583     153.0
BIIB  235.240005      21.0
BKNG  801.119995       6.0
CSCO   18.742292     266.0
DUK    49.744247     100.0
EXC    26.340830     189.0
FB     24.309999     205.0
GE     18.268507     273.0
IBM   154.403519      32.0
T      24.662882     202.0
```

```
/Users/jaiveerkhanna/opt/anaconda3/lib/python3.7/site-packages/ipyke
rnel_launcher.py:32: RuntimeWarning: divide by zero encountered in d
ouble_scalars
```

```
Simulation has completed, here are some performance metrics:
Final Cash Balance =  108261.90892926234
Performance (%) =  116.52381785852468
////////// ----------- //////////
////////// ----------- //////////
Simulation is running from  2013-05-24 until  2018-06-28
Universe Size is:  98
Basket Size is:  10
Starting Capital:  50000
Current Balance:  50000
Rebalance Duration is: 5
Indicator Weights (EMA, SMA, MACD):  0.3333333333333333 0.3333333333
333333 0.3333333333333333
Transaction Costs: 0.0003
Initial Portfolio:
           Entry  Quantity
ABT    32.626583     153.0
BIIB  235.240005      21.0
BKNG  801.119995       6.0
CSCO   18.742292     266.0
DUK    49.744247     100.0
EXC    26.340830     189.0
FB     24.309999     205.0
GE     18.268507     273.0
IBM   154.403519      32.0
T      24.662882     202.0

/Users/jaiveerkhanna/opt/anaconda3/lib/python3.7/site-packages/ipyke
rnel_launcher.py:32: RuntimeWarning: divide by zero encountered in d
ouble_scalars

Simulation has completed, here are some performance metrics:
Final Cash Balance =  109453.10517353936
Performance (%) =  118.90621034707871
////////// ----------- //////////
////////// ----------- //////////
Simulation is running from  2013-05-24 until  2018-06-28
Universe Size is:  98
Basket Size is:  10
Starting Capital:  50000
Current Balance:  50000
Rebalance Duration is: 10
Indicator Weights (EMA, SMA, MACD):  0.3333333333333333 0.3333333333
333333 0.3333333333333333
Transaction Costs: 0.0003
Initial Portfolio:
           Entry  Quantity
ABT    32.626583     153.0
BIIB  235.240005      21.0
BKNG  801.119995       6.0
CSCO   18.742292     266.0
DUK    49.744247     100.0
EXC    26.340830     189.0
FB     24.309999     205.0
GE     18.268507     273.0
IBM   154.403519      32.0
T      24.662882     202.0

/Users/jaiveerkhanna/opt/anaconda3/lib/python3.7/site-packages/ipyke
rnel_launcher.py:32: RuntimeWarning: divide by zero encountered in d
ouble_scalars
```

```
Simulation has completed, here are some performance metrics:
Final Cash Balance =  101188.98218433018
Performance (%) =  102.37796436866036
////////// ----------- //////////
////////// ----------- //////////
Simulation is running from  2013-05-24 until  2018-06-28
Universe Size is:  98
Basket Size is:  10
Starting Capital:  50000
Current Balance:  50000
Rebalance Duration is: 15
Indicator Weights (EMA, SMA, MACD):  0.3333333333333333 0.3333333333
333333 0.3333333333333333
Transaction Costs: 0.0003
Initial Portfolio:
           Entry  Quantity
ABT    32.626583     153.0
BIIB  235.240005      21.0
BKNG  801.119995       6.0
CSCO   18.742292     266.0
DUK    49.744247     100.0
EXC    26.340830     189.0
FB     24.309999     205.0
GE     18.268507     273.0
IBM   154.403519      32.0
T      24.662882     202.0

/Users/jaiveerkhanna/opt/anaconda3/lib/python3.7/site-packages/ipyke
rnel_launcher.py:32: RuntimeWarning: divide by zero encountered in d
ouble_scalars

Simulation has completed, here are some performance metrics:
Final Cash Balance =  99458.60273724164
Performance (%) =  98.91720547448328
////////// ----------- //////////
////////// ----------- //////////
Simulation is running from  2013-05-24 until  2018-06-28
Universe Size is:  98
Basket Size is:  10
Starting Capital:  50000
Current Balance:  50000
Rebalance Duration is: 20
Indicator Weights (EMA, SMA, MACD):  0.3333333333333333 0.3333333333
333333 0.3333333333333333
Transaction Costs: 0.0003
Initial Portfolio:
           Entry  Quantity
ABT    32.626583     153.0
BIIB  235.240005      21.0
BKNG  801.119995       6.0
CSCO   18.742292     266.0
DUK    49.744247     100.0
EXC    26.340830     189.0
FB     24.309999     205.0
GE     18.268507     273.0
IBM   154.403519      32.0
T      24.662882     202.0

/Users/jaiveerkhanna/opt/anaconda3/lib/python3.7/site-packages/ipyke
rnel_launcher.py:32: RuntimeWarning: divide by zero encountered in d
ouble_scalars
```

```
Simulation has completed, here are some performance metrics:
Final Cash Balance =  83025.33537858419
Performance (%) =  66.05067075716839
////////// ----------- //////////
////////// ----------- //////////
Simulation is running from  2013-05-24 until  2018-06-28
Universe Size is:  98
Basket Size is:  10
Starting Capital:  50000
Current Balance:  50000
Rebalance Duration is: 30
Indicator Weights (EMA, SMA, MACD):  0.3333333333333333 0.3333333333
333333 0.3333333333333333
Transaction Costs: 0.0003
Initial Portfolio:
           Entry  Quantity
ABT    32.626583     153.0
BIIB  235.240005      21.0
BKNG  801.119995       6.0
CSCO   18.742292     266.0
DUK    49.744247     100.0
EXC    26.340830     189.0
FB     24.309999     205.0
GE     18.268507     273.0
IBM   154.403519      32.0
T      24.662882     202.0

/Users/jaiveerkhanna/opt/anaconda3/lib/python3.7/site-packages/ipyke
rnel_launcher.py:32: RuntimeWarning: divide by zero encountered in d
ouble_scalars

Simulation has completed, here are some performance metrics:
Final Cash Balance =  92516.00967335136
Performance (%) =  85.03201934670273
////////// ----------- //////////
```

In [36]:

```
#rebalance holds the  return for each of the different rebalance periods
rebalance
```

Out[36]:

Index	1	3	5	10	15	20	30
Net Return	1.084209	1.166524	1.190368	1.024986	0.990359	0.661496	0.851415
Sharpe	1.088488	1.148000	1.173386	1.059033	1.049830	0.784842	0.930970
Sortino	1.446183	1.551438	1.588641	1.411903	1.400704	1.043802	1.246218

In [37]:

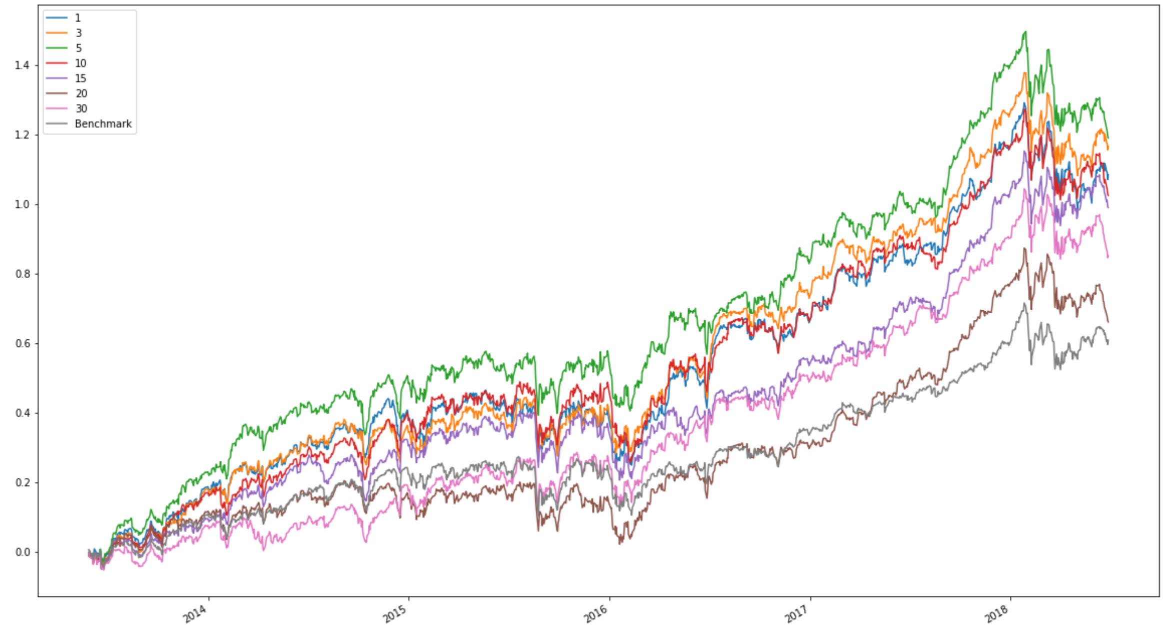
```
rebalance_detailed['Benchmark'] = optimize.analyze['Benchmark Cumulative Returns
(%)']
```

In [38]:

```
#rebalance_detailed shows the daily cumulative returns for all different rebalan
ce periods compared to the benchmark
rebalance_detailed.plot(figsize=(20,12))
```

Out[38]:

<matplotlib.axes._subplots.AxesSubplot at 0x120b8b990>



Step 4: Optimize for Weightage of Stocks

In [39]:

```
optimize = Backtester()
```

In [40]:

```
weightage = [(1/3,1/3,1/3),(1/2,1/4,1/4),(1/4,1/2,1/4),(1/4,1/4,1/2),(1/6,1/6,2/
3),(1/6,2/3,1/6),(1/6,1/6,2/3)]
indicator_weight = pd.DataFrame()
indicator_weight.insert(0,'Index', ['Net Return', 'Sharpe', 'Sortino'])
indicator_weight.set_index('Index',inplace = True)
indicator_weight_detailed = pd.DataFrame()

for combination in weightage:
    optimize.simulate(weights = combination)
    optimize.analyze_performance()
    analysis = optimize.analyze
    indicator_weight[combination] = analysis['Cumulative Returns (%)'].iloc[-1]
    indicator_weight_detailed[combination] = analysis['Cumulative Returns (%)']
    indicator_weight[combination].loc['Sharpe'] = sharpe(analysis['Daily Returns
(%)'])
    indicator_weight[combination].loc['Sortino'] = sortino(analysis['Daily Retur
ns (%)'])
```

```
////////// ----------- //////////
Simulation is running from  2013-05-24 until  2018-06-28
Universe Size is:  98
Basket Size is:  10
Starting Capital:  50000
Current Balance:  50000
Rebalance Duration is: 5
Indicator Weights (EMA, SMA, MACD):  0.3333333333333333 0.3333333333
333333 0.3333333333333333
Transaction Costs: 0.0003
Initial Portfolio:
           Entry  Quantity
ABT    32.626583     153.0
BIIB  235.240005      21.0
BKNG  801.119995       6.0
CSCO   18.742292     266.0
DUK    49.744247     100.0
EXC    26.340830     189.0
FB     24.309999     205.0
GE     18.268507     273.0
IBM   154.403519      32.0
T      24.662882     202.0
```

```
/Users/jaiveerkhanna/opt/anaconda3/lib/python3.7/site-packages/ipyke
rnel_launcher.py:32: RuntimeWarning: divide by zero encountered in d
ouble_scalars
```

```
Simulation has completed, here are some performance metrics:
Final Cash Balance =  109453.10517353936
Performance (%) =  118.90621034707871
////////// ----------- //////////
////////// ----------- //////////
Simulation is running from  2013-05-24 until  2018-06-28
Universe Size is:  98
Basket Size is:  10
Starting Capital:  50000
Current Balance:  50000
Rebalance Duration is: 5
Indicator Weights (EMA, SMA, MACD):  0.5 0.25 0.25
Transaction Costs: 0.0003
Initial Portfolio:
           Entry  Quantity
ABT    32.626583     153.0
BIIB  235.240005      21.0
BKNG  801.119995       6.0
CSCO   18.742292     266.0
DUK    49.744247     100.0
EXC    26.340830     189.0
FB     24.309999     205.0
GE     18.268507     273.0
IBM   154.403519      32.0
T      24.662882     202.0
```

```
/Users/jaiveerkhanna/opt/anaconda3/lib/python3.7/site-packages/ipyke
rnel_launcher.py:32: RuntimeWarning: divide by zero encountered in d
ouble_scalars
```

```
Simulation has completed, here are some performance metrics:
Final Cash Balance =  110355.01229132117
Performance (%) =  120.71002458264233
////////// ----------- //////////
////////// ----------- //////////
Simulation is running from  2013-05-24 until  2018-06-28
Universe Size is:  98
Basket Size is:  10
Starting Capital:  50000
Current Balance:  50000
Rebalance Duration is: 5
Indicator Weights (EMA, SMA, MACD):  0.25 0.5 0.25
Transaction Costs: 0.0003
Initial Portfolio:
           Entry  Quantity
ABT    32.626583     153.0
BIIB  235.240005      21.0
CSCO   18.742292     266.0
DUK    49.744247     100.0
EXC    26.340830     189.0
FB     24.309999     205.0
GE     18.268507     273.0
GILD   48.009838     104.0
IBM   154.403519      32.0
T      24.662882     202.0
```

```
/Users/jaiveerkhanna/opt/anaconda3/lib/python3.7/site-packages/ipyke
rnel_launcher.py:32: RuntimeWarning: divide by zero encountered in d
ouble_scalars
```

```
Simulation has completed, here are some performance metrics:
Final Cash Balance =  102667.21214139783
Performance (%) =  105.33442428279567
////////// ----------- //////////
////////// ----------- //////////
Simulation is running from  2013-05-24 until  2018-06-28
Universe Size is:  98
Basket Size is:  10
Starting Capital:  50000
Current Balance:  50000
Rebalance Duration is: 5
Indicator Weights (EMA, SMA, MACD):  0.25 0.25 0.5
Transaction Costs: 0.0003
Initial Portfolio:
           Entry  Quantity
ABT    32.626583     153.0
BIIB  235.240005      21.0
CSCO   18.742292     266.0
DUK    49.744247     100.0
EXC    26.340830     189.0
FB     24.309999     205.0
GE     18.268507     273.0
IBM   154.403519      32.0
PFE    22.383799     223.0
T      24.662882     202.0
```

```
/Users/jaiveerkhanna/opt/anaconda3/lib/python3.7/site-packages/ipyke
rnel_launcher.py:32: RuntimeWarning: divide by zero encountered in d
ouble_scalars
```

```
Simulation has completed, here are some performance metrics:
Final Cash Balance =  112589.3314215654
Performance (%) =  125.1786628431308
////////// ----------- //////////
////////// ----------- //////////
Simulation is running from  2013-05-24 until  2018-06-28
Universe Size is:  98
Basket Size is:  10
Starting Capital:  50000
Current Balance:  50000
Rebalance Duration is: 5
Indicator Weights (EMA, SMA, MACD):  0.16666666666666666 0.166666666
66666666 0.6666666666666666
Transaction Costs: 0.0003
Initial Portfolio:
           Entry  Quantity
ABT    32.626583     153.0
COST   98.545853      50.0
CSCO   18.742292     266.0
DUK    49.744247     100.0
EXC    26.340830     189.0
FB     24.309999     205.0
GE     18.268507     273.0
IBM   154.403519      32.0
PFE    22.383799     223.0
T      24.662882     202.0

/Users/jaiveerkhanna/opt/anaconda3/lib/python3.7/site-packages/ipyke
rnel_launcher.py:32: RuntimeWarning: divide by zero encountered in d
ouble_scalars

Simulation has completed, here are some performance metrics:
Final Cash Balance =  111957.61289469128
Performance (%) =  123.91522578938256
////////// ----------- //////////
////////// ----------- //////////
Simulation is running from  2013-05-24 until  2018-06-28
Universe Size is:  98
Basket Size is:  10
Starting Capital:  50000
Current Balance:  50000
Rebalance Duration is: 5
Indicator Weights (EMA, SMA, MACD):  0.16666666666666666 0.666666666
6666666 0.16666666666666666
Transaction Costs: 0.0003
Initial Portfolio:
           Entry  Quantity
BIIB  235.240005      21.0
BKNG  801.119995       6.0
CSCO   18.742292     266.0
DUK    49.744247     100.0
EXC    26.340830     189.0
FB     24.309999     205.0
GE     18.268507     273.0
GILD   48.009838     104.0
IBM   154.403519      32.0
T      24.662882     202.0
```

```
/Users/jaiveerkhanna/opt/anaconda3/lib/python3.7/site-packages/ipyke
rnel_launcher.py:32: RuntimeWarning: divide by zero encountered in d
ouble_scalars
```

```
Simulation has completed, here are some performance metrics:
Final Cash Balance =  107151.69604785078
Performance (%) =  114.30339209570157
////////// ----------- //////////
////////// ----------- //////////
Simulation is running from  2013-05-24 until  2018-06-28
Universe Size is:  98
Basket Size is:  10
Starting Capital:  50000
Current Balance:  50000
Rebalance Duration is: 5
Indicator Weights (EMA, SMA, MACD):  0.16666666666666666 0.166666666
66666666 0.6666666666666666
Transaction Costs: 0.0003
Initial Portfolio:
           Entry  Quantity
ABT    32.626583     153.0
COST   98.545853      50.0
CSCO   18.742292     266.0
DUK    49.744247     100.0
EXC    26.340830     189.0
FB     24.309999     205.0
GE     18.268507     273.0
IBM   154.403519      32.0
PFE    22.383799     223.0
T      24.662882     202.0
```

```
/Users/jaiveerkhanna/opt/anaconda3/lib/python3.7/site-packages/ipyke
rnel_launcher.py:32: RuntimeWarning: divide by zero encountered in d
ouble_scalars
```

```
Simulation has completed, here are some performance metrics:
Final Cash Balance =  111957.61289469128
Performance (%) =  123.91522578938256
////////// ----------- //////////
```

In [41]:

```
#indicator_weight holds the returns for different combinations of weights for EM
A, SMA, and MACD
indicator_weight
```

Out[41]:

Index	(0.3333333333333333, 0.3333333333333333, 0.3333333333333333)	(0.5, 0.25, 0.25)	(0.25, 0.5, 0.25)	(0.25, 0.25, 0.5)	(0.16666666666666666, 0.16666666666666666, 0.6666666666666666)	(0.1666666 0.666666 0.1666666
Net Return	1.190368	1.208417	1.054573	1.253133	1.240491	
Sharpe	1.173386	1.183265	1.082978	1.217231	1.205799	
Sortino	1.588641	1.599895	1.466780	1.659277	1.639819	

In [42]:

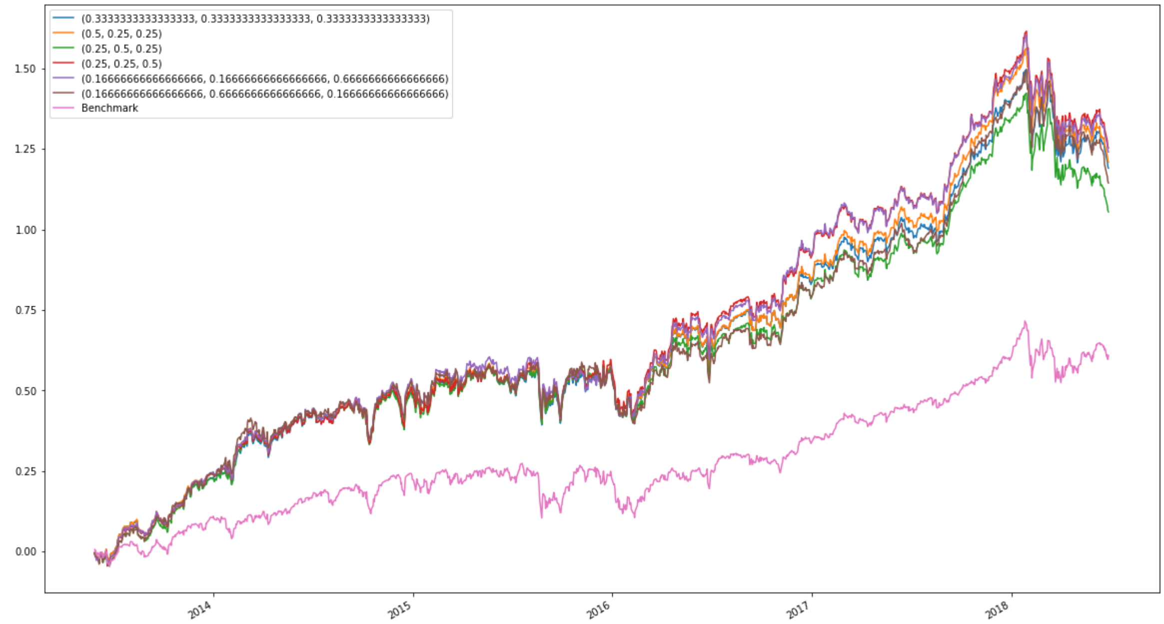
```
indicator_weight_detailed['Benchmark'] = analysis['Benchmark Cumulative Returns
 (%)']
```

In [43]:

```
indicator_weight_detailed.plot(figsize=(20,12))
```

Out[43]:

<matplotlib.axes._subplots.AxesSubplot at 0x1206a8850>



Step 5: Try optimizing all variables in one go (eyeballing)

In [44]:

```
optimize = Backtester()
```

In [45]:

```
optimize.simulate(basket_size = 10, rebalance_duration= 5, weights= (1/6,2/6,3/6
))
optimize.analyze_performance()
```

```
////////// ----------- //////////
Simulation is running from  2013-05-24 until  2018-06-28
Universe Size is:  98
Basket Size is:  10
Starting Capital:  50000
Current Balance:  50000
Rebalance Duration is: 5
Indicator Weights (EMA, SMA, MACD):  0.16666666666666666 0.333333333
3333333 0.5
Transaction Costs: 0.0003
Initial Portfolio:
           Entry  Quantity
ABT    32.626583     153.0
BIIB  235.240005      21.0
CSCO   18.742292     266.0
DUK    49.744247     100.0
EXC    26.340830     189.0
FB     24.309999     205.0
GE     18.268507     273.0
IBM   154.403519      32.0
PFE    22.383799     223.0
T      24.662882     202.0

/Users/jaiveerkhanna/opt/anaconda3/lib/python3.7/site-packages/ipyke
rnel_launcher.py:32: RuntimeWarning: divide by zero encountered in d
ouble_scalars

Simulation has completed, here are some performance metrics:
Final Cash Balance =  112753.61791458014
Performance (%) =  125.50723582916028
////////// ----------- //////////
```

In [46]:

```
optimize.analyze.plot(y={'Cumulative Returns (%)','Benchmark Cumulative Returns
 (%)'}, figsize=(20,12))
```

Out[46]:

<matplotlib.axes._subplots.AxesSubplot at 0x120cf8f10>

Step 6: Optimize across all variables using for loops

WARNING! This is not efficient as it uses multiple nested for loops. Takes a long time to run!

In [47]:

```
best_basket = 0
best_rebalance = 0
best_weight = [0,0,0]
best_return = 0
```

In [48]:

```
weightage = [(1/3,1/3,1/3),(1/2,1/4,1/4),(1/4,1/2,1/4),(1/4,1/4,1/2),(1/6,1/6,2/
3),(1/6,2/3,1/6),(1/6,1/6,2/3)]

for baskets in (5, 10, 15):
    for rebalances in (1, 5, 10, 15):
        for weight in weightage:
            optimize.simulate(basket_size= baskets, rebalance_duration= rebalanc
es, weights= weight)
            optimize.analyze_performance()
            if (best_return < optimize.analyze['Cumulative Returns (%)'].iloc[-1
]):
                best_basket = baskets
                best_rebalance = rebalances
                best_weight = weight
                best_return = optimize.analyze['Cumulative Returns (%)'].iloc[-1
]
```

```
Simulation has completed, here are some performance metrics:
Final Cash Balance =  87884.59978380619
Performance (%) =  75.76919956761238
////////// ----------- //////////
////////// ----------- //////////
Simulation is running from  2013-05-24 until  2018-06-28
Universe Size is:  98
Basket Size is:  15
Starting Capital:  50000
Current Balance:  50000
Rebalance Duration is: 15
Indicator Weights (EMA, SMA, MACD):  0.16666666666666666 0.166666666
66666666 0.6666666666666666
Transaction Costs: 0.0003
Initial Portfolio:
           Entry  Quantity
ABT    32.626583     102.0
AMZN  261.739990      12.0
BIIB  235.240005      14.0
BKNG  801.119995       4.0
COST   98.545853      33.0
CSCO   18.742292     177.0
CVX    93.243530      35.0
DUK    49.744247      66.0
EXC    26.340830     126.0
FB     24.309999     137.0
GE     18.268507     182.0
IBM   154.403519      21.0
PFE    22.383799     148.0
T      24.662882     135.0
XOM    68.638519      48.0
```

```
/Users/jaiveerkhanna/opt/anaconda3/lib/python3.7/site-packages/ipyke
rnel_launcher.py:32: RuntimeWarning: divide by zero encountered in d
ouble_scalars
```

```
Simulation has completed, here are some performance metrics:
Final Cash Balance =  97450.45524151801
Performance (%) =  94.90091048303601
////////// ----------- //////////
```

In [49]:

```
print(best_return, best_basket, best_rebalance, best_weight)
```

```
1.4826061520201104 5 1 (0.25, 0.5, 0.25)
```

Note: Previously there was an attempt to optimize simulation using scipy or pulp. However, these approaches did not work well and have been omitted from this version.

Part 4: Out of Sample Testing

Testing

After optimizing and processing the data, it is important to test it using out-of-sample data sets to make sure that the model works, and that we didn't overfit the data. We want this method of generating returns to be valid across any time period, so that we are confident it can be applied to future data.

Step 1: Future Data Test (June 2018 - June 2020)

In [50]:

```
test = Backtester()

#Start and End Dates are outside of the training dataset
test_start = '2018-06-28'
test_end = '2020-06-15'
```

In [51]:

```
test.simulate(sim_start = pd.Timestamp(test_start), sim_end= pd.Timestamp(test_e
nd),
              basket_size = 10, rebalance_duration= 5, weights= (1/6,2/6,3/6))
test.analyze_performance()
```

```
////////// ----------- //////////
Simulation is running from  2018-06-28 until  2020-06-15
Universe Size is:  98
Basket Size is:  10
Starting Capital:  50000
Current Balance:  50000
Rebalance Duration is: 5
Indicator Weights (EMA, SMA, MACD):  0.16666666666666666 0.333333333
3333333 0.5
Transaction Costs: 0.0003
Initial Portfolio:
            Entry  Quantity
BA     322.092133      15.0
BKNG  2019.489990       2.0
BLK    473.291321      10.0
CL      61.386063      81.0
DHR     97.607010      51.0
DUK     71.863907      69.0
EMR     64.122101      77.0
FDX    219.685364      22.0
HON    130.626175      38.0
UPS     98.038277      50.0

/Users/jaiveerkhanna/opt/anaconda3/lib/python3.7/site-packages/ipyke
rnel_launcher.py:32: RuntimeWarning: divide by zero encountered in d
ouble_scalars

Simulation has completed, here are some performance metrics:
Final Cash Balance =  60299.07667748315 Final Portfolio + Cash Valu
e: 60317.077766901755
Performance (%) =  20.5981533549663
////////// ----------- //////////
```

In [52]:

```
test.analyze.plot(y={'Cumulative Returns (%)','Benchmark Cumulative Returns (%)'
}, figsize=(20,12))
```

Out[52]:

<matplotlib.axes._subplots.AxesSubplot at 0x11e691950>

Metrics for Strategy

In [53]:

```
print('Return: ', test.analyze['Cumulative Returns (%)'].iloc[-1])
print('Sharpe: ', sharpe(test.analyze['Daily Returns (%)']))
print('Sortino: ', sortino(test.analyze['Daily Returns (%)']))
```

```
Return:  0.20669015582721406
Sharpe:  0.4790563876326162
Sortino:  0.5704965959577769
```

Metrics for Benchmark

In [54]:

```
print('Return: ', test.analyze['Benchmark Cumulative Returns (%)'].iloc[-1])
print('Sharpe: ', sharpe(test.analyze['Benchmark Daily Returns (%)']))
print('Sortino: ', sortino(test.analyze['Benchmark Daily Returns (%)']))
```

```
Return:  0.1749288502390769
Sharpe:  0.44836092107927045
Sortino:  0.4949031944418458
```

Step 2: Past Data Test (Jan 2010- Jan 2013)

Note: Since this data set is from before the training period, we must remove a few stock that did not exist during this time from our list of the SP100. Look at the commented out code in the first few cells to see that ABBV, FB, KMI, and GM have to be removed for this backtest to work.

In [324]:

```
test = Backtester()
test_start = '2010-01-04'
test_end = '2012-12-28'
```

In [325]:

```
test.simulate(sim_start = pd.Timestamp(test_start), sim_end= pd.Timestamp(test_e
nd), basket_size = 10, rebalance_duration= 5, weights= (1/6,2/6,3/6))
test.analyze_performance()
```

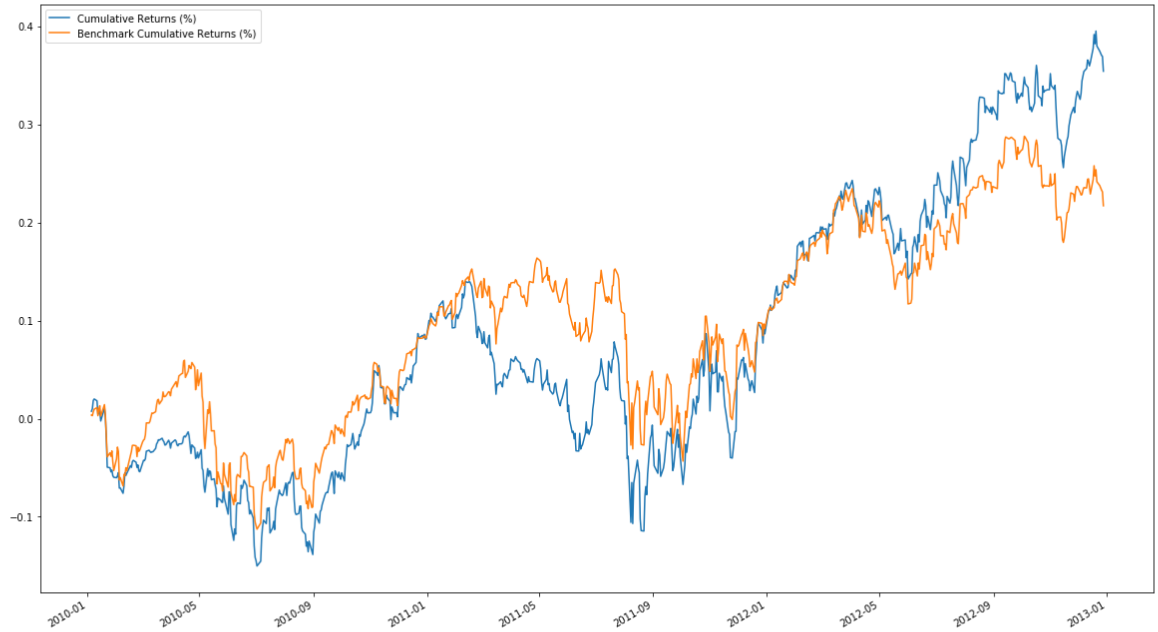
```
////////// ----------- //////////
Simulation is running from  2010-01-04 until  2012-12-28
Universe Size is:  98
Basket Size is:  10
Starting Capital:  50000
Current Balance:  50000
Rebalance Duration is: 5
Indicator Weights (EMA, SMA, MACD):  0.16666666666666666 0.333333333
3333333 0.5
Transaction Costs: 0.0003
Initial Portfolio:
          Entry  Quantity
COF   33.047562     151.0
COST  45.815430     109.0
CRM   18.705000     267.0
CSCO  18.785395     266.0
CVS   26.591206     187.0
CVX   53.047504      94.0
DD    30.727852     162.0
DHR   18.391703     271.0
DIS   27.933924     178.0
XOM   48.404034     103.0

/Users/jaiveerkhanna/opt/anaconda3/lib/python3.7/site-packages/ipyke
rnel_launcher.py:33: RuntimeWarning: divide by zero encountered in d
ouble_scalars

Simulation has completed, here are some performance metrics:
Final Cash Balance =  67665.37732452578 Final Portfolio + Cash Valu
e: 67685.63390696366
Performance (%) =  35.330754649051556
////////// ----------- //////////
```

In [326]:

```
test.analyze.plot(y={'Cumulative Returns (%)','Benchmark Cumulative Returns (%)'
}, figsize=(20,12))
```

Out[326]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x11d198e90>
```



Metrics for Strategy

In [327]:

```
print('Return: ', test.analyze['Cumulative Returns (%)'].iloc[-1])
print('Sharpe: ', sharpe(test.analyze['Daily Returns (%)']))
print('Sortino: ', sortino(test.analyze['Daily Returns (%)']))
```

```
Return:  0.3541177284838506
Sharpe:  0.6376540789857855
Sortino:  0.8632473637229557
```

Metrics for Benchmark

In [328]:

```
print('Return: ', test.analyze['Benchmark Cumulative Returns (%)'].iloc[-1])
print('Sharpe: ', sharpe(test.analyze['Benchmark Daily Returns (%)']))
print('Sortino: ', sortino(test.analyze['Benchmark Daily Returns (%)']))
```

```
Return:  0.21701453270324933
Sharpe:  0.45958147539397604
Sortino:  0.5955055896505499
```

Conclusion

Overall, I am extremely satisfied with the results of my strategy. Not only did it outperform the benchmark in raw returns, it also outperformed in terms of risk-adjusted returns (healthier Sharpe and Sortino ratios).

During my optimization period, I was glad to see that the strategy consistently outperformed the benchmark, even when calculated on a year by year basis. This improves my confidence in the strategy's power, and it is a good sign to see that each individual year was succesful.

Furthermore, I am inclined to believe the strategy is not overfitted to its training data, as it outperformed the benchmark during both test periods (beating the benchmark by 3% and 15% during the future and historical test period resepectively).

I am looking forward to seeing if it stands the test of time.

Disclaimer: Past performance is not indicative of future returns