```
import pandas as pd
import code_base as cb
import matplotlib.pyplot as plt
import numpy as np

import statsmodels.api as sm

end
```

Group Project - Earnings Adjusted Momentum & Value Stratergies

Group-21 Member: LAM Chun Ming EID: 58555905

Note **some of the format is broken when converting this to the current pdf file** for the full notebook file , code base, and data set used ; they can be found here : https://github.com/Toby3220/EF-5058-HFS/ under the **Group Project** folder

1 Introduction

The phenomenon of abnormal returns surrounding corporate earnings releases has been documented since the late 1960s. Beaver (1968) first showed significant price reaction to quarterly results; Ball and Brown (1968) measured significant returns in a multi day window around the earnings announcement. Brown and Warner (1985) then reconfirmed these results by showing the persistence even accounting for other potential explainations, whilst many other works propose explanations to this phenomenon, from the risk associated from the disclosure of idosyncratic information (Chari et al, 1988; Cohen et al, 2007), attention-grabbing effects (Lee, 1992).

However, given that much of the academic literature has demonstrated and focused on the strong effect in a tight window around 1-2 weeks, before and after the earnings accrual day, longer-term effects have been less focused on. Some studies have looked into earning announcement drift that show the effect lasts on a longer horizon up to 60 trading days (Bernard & Thomass, 1989). However, they are less integrated with the complete momentum value framework.

The current factor framework was attributed to Fama and French's (1992) seminal work, and more recently, Asness (2013) shows that the combination of value and momentum factors outperformed compaired to each factor alone. Whilst some studies have examined the relationship between earnings and momentum (Chan, Jegadeesh and Lakonishok 1996), we wish to investigate the whether of earnings signals add additonal information when intergrated into a composite momentum-value stratergy; Given we might expect a significant earnings shock to materially alter a stock's momentum and value characteristics in the coming months, especially given large surprises.

2 Data

2.1 Collection Method

We collected 21 years of monthly data from 2000 to 2021 from a Bloomberg Terminal, which was used to generate a 20-year backtest period between 2001 and 2001. The first year was unusable as the momentum factor requires a 12-month lookback period.

Specifically, we collected the adjusted closing price, market cap, announcement earnings, announcement earnings estimate, price to book ratio, and S&P 500 membership information data, totalling 264 sample periods and 1134 stock tickers, which were all stock tickers that were members of the S&P 500 at any point within this period.

Membership information is sampled at the end of the year, whilst stock data is sampled at the start of the month. Lastly, any announcement information is sampled at the time of announcement, then attributed to the month at which it occurred (for example, announcement in 2003-05-23 -> 2003-05)

2.2 Data Formatting

Using the data collected, we then created Momentum (MOM), Value (VAL), and Earnings Surprise (SUP) fields.

```
# importing the collected data
df_dict = cb.read_multi_csv(

filenames=["df_memboolG_spx","df_mcap_spx","df_monthly_spx","df_annsup
_spx","df_annest_spx","df_anndat_spx","df_annacc_spx","df_P2BR_spx","d
f_Beta_spx"],
    name = ["tmbool", "mcap", "price", "annsup", "annest", "anndat",
"annacc", "ptbr", "beta"]
)
df_dict["tmbool"].shape
```

```
(264, 1134)
```

Factor Constructution: Value (VAL)

The value factor "**VAL**" is the inverse of the price to book ratio (PTBR), to which we applied mean, standard deviation normalisation from the cross-sectional mean and standard deviations at time t. we shifted the information from 6 months ago to prevent any potential hindsight biases

$$\mathrm{V}_{t,i}=\frac{1}{\mathrm{PTBR}_{t-6,i}}$$

$$\mathrm{VAL}_{t,i}=\frac{V_{t,i}-\mu_t}{\sigma_t}$$

```
# value factor, normalisation applied later
df_dict["VAL"] = 1/df_dict["ptbr"]
df_dict["VAL"]= df_dict["VAL"].shift(6)
```

Factor Construction: Momentum (MOM)

We created this first from calculating the monthly returns, due to our notation, this is the returns for holding stock ticker $[i]$ from period $[t-1] \rightarrow [t]$

$$R_{t,i} = \frac{P_{t,i} - P_{t-1,i}}{P_{t-1,i}}$$

The momentum factor "**MOM**" is then the cumulative returns of the past year, excluding the most recent month. We then applied a mean standard deviation normalisation (again applied later)

$$M_{t,i} = \prod_{k=1}^{12} \left(1 + R_{t-k,i}\right)$$

$$\text{MOM}_{t,i} = \frac{M_{t,i} - \mu_t}{\sigma_t}$$

```
# creating stock returns
df_dict["ret"] = (df_dict["price"]/df_dict["price"].shift(1))-1

# calculating momentum factor, the returns from -2 to -12
tdf = pd.DataFrame(df_dict["ret"])
tdf +=1
tdf = tdf.rolling(11,11).apply(np.prod, raw=True)
tdf = tdf**(1/11)

df_dict["MOM"] = tdf.shift(1)
df_dict["MOM"] -=1
```

Factor Construction: Earnings Surprise & Other Earnings Data (SUP, ERG, ETG)

We delayed (shifted) actual Earnings results by 1 period to prevent any hindsight bias (for example, earnings announced: 2003-05-23 => data collected: 2003-05, => data available: 2003-06). We did not delay the earnings estimate as we assume reasonable estimates are likely available by the start of the month

The surprise factor is directly collected from Bloomberg and delayed by 1 period; mathematically, this is simply the difference between actual earnings and earnings estimate. For factor "**SUP-Z**" we then applied the mean standard deviation normalisation, whilst for factor "**SUP**" we did not.

We then also created supplementary factors like earnings growth "**ERG**" and earnings estimate growth "**ETG**", which are the % growth from four earnings announcements ago. We chose four

earnings announcements as earnings are typically quarterly, and can also be seasonal; this choice allows us to focus on relative growth on an annual basis, to which we applied a standard deviation normalisation

$$E_{t,i} = \frac{E_{t,i} - E_i^{(4)}}{E_i^{(4)}}$$

$$\mathrm{ERG}_i / \mathrm{ETG}_i = \frac{E_{t,i}}{\sigma_t}$$

```
# shifting regieme by 1 to prevent hindsight bias
df_dict["annacc"]=df_dict["annacc"].shift(1)
df_dict["annsup"]=df_dict["annsup"].shift(1)

# creating a earnings suprise factor (SUP/ SUP-Z)
df_dict["SUP"] =  df_dict["anndat"].shift(1)*df_dict["annsup"]
# df_dict["SUP"].ffill(inplace=True)
df_dict["SUP-Z"] = df_dict["SUP"].copy()
df_dict["SUP-Z"].ffill(inplace=True)

# earnings growth factor (ERG)
df_dict["AP4"] = cb.past_nvalid_values(df_dict["annacc"],4)
ans = (df_dict["annacc"].divide(df_dict["AP4"])-1)
ans[ans<-100]=-100
ans[ans>2000]=2000
df_dict["ERG"] = ans
df_dict["ERG"].ffill(inplace=True)

# earning estimates (EST), actual earnings (ERN)
df_dict["EST"] = df_dict["annest"]

df_dict["EP4"] = cb.past_nvalid_values(df_dict["annest"],4)
ans = (df_dict["annest"].ffill().divide(df_dict["EP4"])-1)
ans[ans<-100]=-100
ans[ans>2000]=2000
df_dict["ETG"] = ans
df_dict["ETG"].ffill(inplace=True)



out_dict = cb.dict_transfer(df_dict,
["ret","VAL","MOM","SUP","ERG","ETG","SUP-Z"])
```

In Sample & Out of Sample Sets

We split the collected data into an In Sample (60% from 2001 -> 2013) and Reserved Out of Sample Data Set (40% from 2013-> 2021) to reduce bias and data mining risks

```
# generate a "test" set & "training" set
tmbool =  df_dict["tmbool"].copy()
# ratio = 60%
ratio = 0.6
row = int(round((tmbool.shape[0])*ratio,0))
tmbool_train = tmbool.iloc[12:row,:]
tmbool_test = tmbool.iloc[row:,:]

# index portfolio weights
mcap_weight = (df_dict["mcap"]*tmbool).copy()
mcap_weight =
mcap_weight.divide(mcap_weight.sum(1,skipna=True),"index")

mcap_weight_train = mcap_weight.iloc[12:row,:]
mcap_weight_test = mcap_weight.iloc[row:,:]
```

Normalisation

we then applied mean standard deviation normalisation to Momentum Factor and Value Factor to cacluat standard scores.

we also applied just standard deviation normalistion to our supplimentary factors (ERG and ETG) to preserve the positive and negative directional information

values above 5 ("standard deviations") are discarded by setting them to 0, as they likely represent extereme cases data points less suitable to high level statistical analysis

```
data = cb.data_obj(tmbool,out_dict)

# apply nomalisation
data.calc_z_scores(True,["MOM","VAL"],clip="Discard")
data.calc_z_scores(True,["ERG","ETG","SUP-Z"],False,clip="Discard")
```

3 Emperical Analysis

3.1 Portfolio Construction and Momentum and Value Benchmark

We first investigate the effect of the value and momentum factor portfolio, which is used as our benchmark.

Following Asness 2013, we construct a composite score by taking the simple average of our momentum and value standard score.

$$S_{i,t} = 0.5 * \mathrm{MOM}_{i,t} + 0.5 * \mathrm{VAL}_{i,t}$$

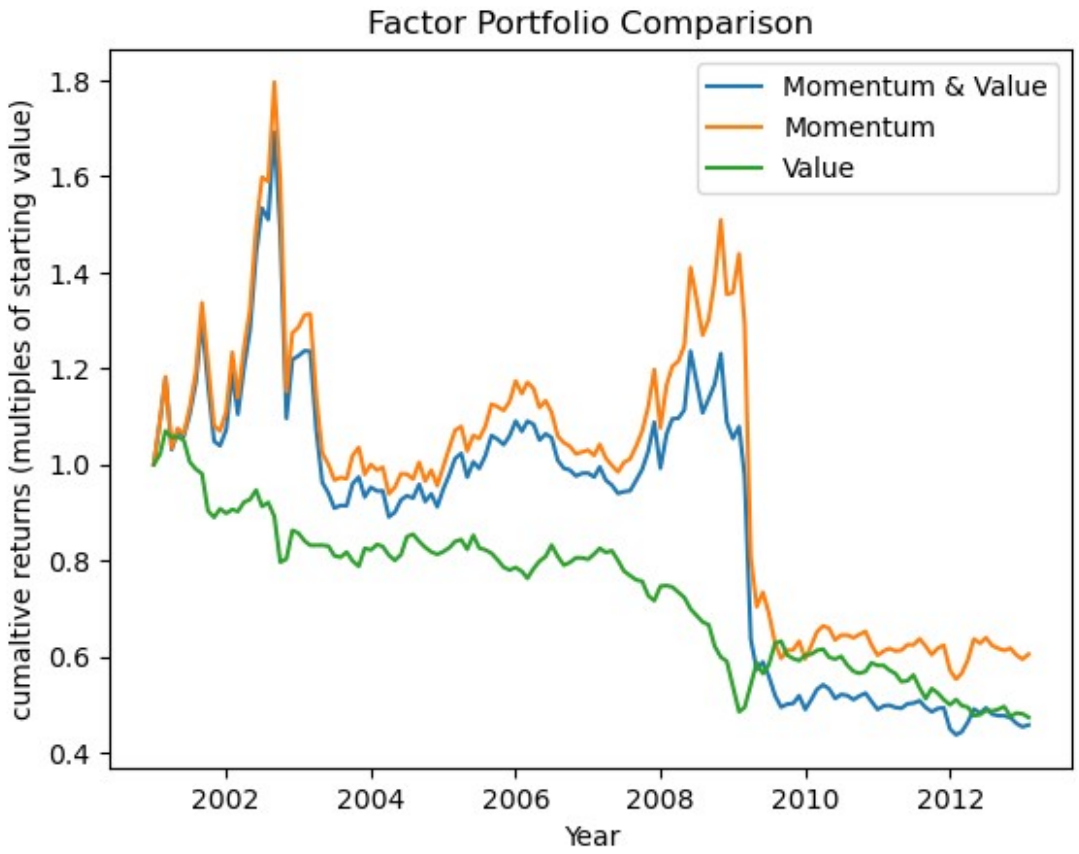
We construct the portfolio by ranking all scores and holding a long position in the top 30% of all stocks, whilst shorting the bottom 30%, for each stock selected, they hold equal weighting and both long and short portfolio net to 0 for a dollar neutrality overall

As our returns are constructed as the returns holding stock $[i]$ from period $[t-1]->[t]$, we shift our portfolio weights one period forward to represent returns from $[t]->[t+1]$. Each month, the portfolio is rebalanced in accordance with our score. We apply a 10 basis point net transaction fee representing any transaction costs and slippage , calculated as the difference between the portfolio weights in each period.

```
# calculate momentum value composite factor
momval = (0.5*data.get("VAL").fillna(0)
          +0.5*data.get("MOM").fillna(0))
data.append("MOM_VAL", momval)
# data.regulate(["MOM_VAL"])
# data.calc_z_scores(inplace=True,fields=["MOM_VAL"])
port_momval = data.to_port("MOM_VAL",tmbool=tmbool_train)


port_momval.gen_weights_from_score(0.3)
momval_ret, momval_cret =
port_momval.get_port_ret(weight=port_momval.lsw,bps=10)
```

To evaluate the result of our momentum & value portfolio, we also applied the same method to our momentum and value factors by themselves, we then plotted the cumulative long-short portfolio factor returns, net of transaction fees below

```
port_mom = data.to_port("MOM",tmbool=tmbool_train)
port_mom.gen_weights_from_score(0.3)
mom_ret, mom_cret = port_mom.get_port_ret(weight=port_mom.lsw,bps=10)

port_val = data.to_port("VAL",tmbool=tmbool_train)
port_val.gen_weights_from_score(0.3)
val_ret, val_cret = port_val.get_port_ret(weight=port_val.lsw,bps=10)

comb_ret = (val_ret+mom_ret)/2
comb_cret=(comb_ret+1).cumprod()
# ploting graph
plt.plot(momval_cret)
plt.plot(mom_cret)
plt.plot(val_cret)

plt.legend(["Momentum & Value","Momentum","Value"])
plt.xlabel("Year")
plt.ylabel("cumaltive returns (multiples of starting value)")
plt.title("Factor Portfolio Comparison")


Text(0.5, 1.0, 'Factor Portfolio Comparison')
```




```
# Information Ratios
print("Information Ratios")
print("Combined Relative to Momentum :
{}".format(round(cb.information_ratio(momval_ret,mom_ret),3)))
print("Combined Relative to Value :
{}".format(round(cb.information_ratio(momval_ret,val_ret),3)))
```

```
                                  Relative to Momentum Relative to
Value
Annualised Information Ratio                    -0.840
0.067
Annualised Sharpe Ratio (rf = 3%)               -0.334              -
0.334
Max Drawdown                                     0.742
0.742
Annualised Volatility                            0.209
0.209
Annualised Average Return                       -0.040              -
0.040
```

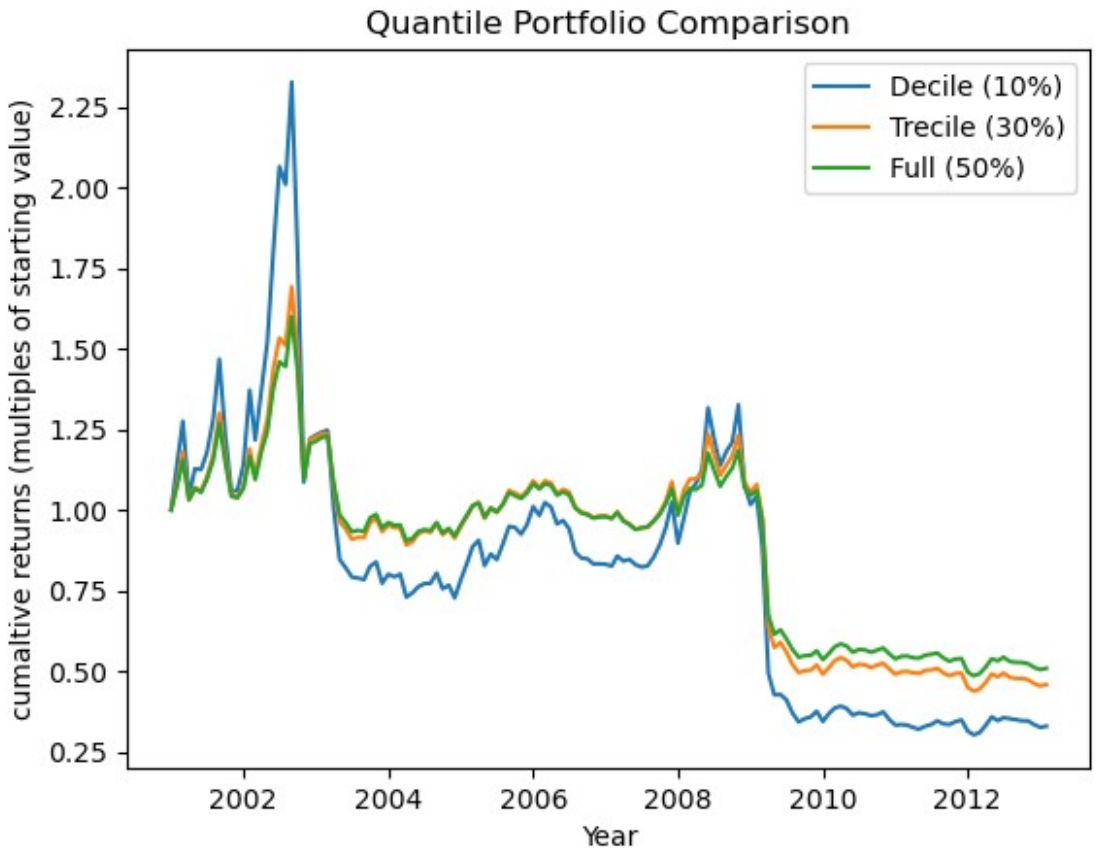
We actually find that all factors portfolios underperformed the S&P 500 index, but also that the value and momentum combined portfolio also underperformed the momentum factor by itself.

This is contrary to the findings in Asness 2013. Nevertheless, we shall use this as our benchmark, due to our stated research interest

Investigating the result: Different Long Short Portfolios

```
# Decile Portfolios (10%)
port_momval.gen_weights_from_score(0.1)
momval_tret, momval_tcret =
port_momval.get_port_ret(weight=port_momval.lsw,bps=10)

# Trecile Portfolios (30%), standard in Fama French
port_momval.gen_weights_from_score(0.3)
momval_tret2, momval_tcret2 =
port_momval.get_port_ret(weight=port_momval.lsw,bps=10)

# Full Portfolios (50%), holding the entire universe
port_momval.gen_weights_from_score(0.5)
momval_tret3, momval_tcret3 =
port_momval.get_port_ret(weight=port_momval.lsw,bps=10)

plt.plot(momval_tcret)
plt.plot(momval_tcret2)
plt.plot(momval_tcret3)

plt.legend(["Decile (10%)","Trecile (30%)","Full (50%)"])
plt.xlabel("Year")
plt.ylabel("cumaltive returns (multiples of starting value)")
plt.title("Quantile Portfolio Comparison")
```

```
Text(0.5, 1.0, 'Quantile Portfolio Comparison')
```




```
# Information Ratios
print("Information Ratios Relative to Decile (10%) Portfolio")
print("Trecile (30%) Portfolio :
{}".format(round(cb.information_ratio(momval_tret2,momval_tret),3)))
print("Full (50%) Portfolio :
{}".format(round(cb.information_ratio(momval_tret3,momval_tret),3)))
```

```
Information Ratios Relative to Decile (10%) Portfolio
Trecile (30%) Portfolio : -0.022
Full (50%) Portfolio : 0.004
```

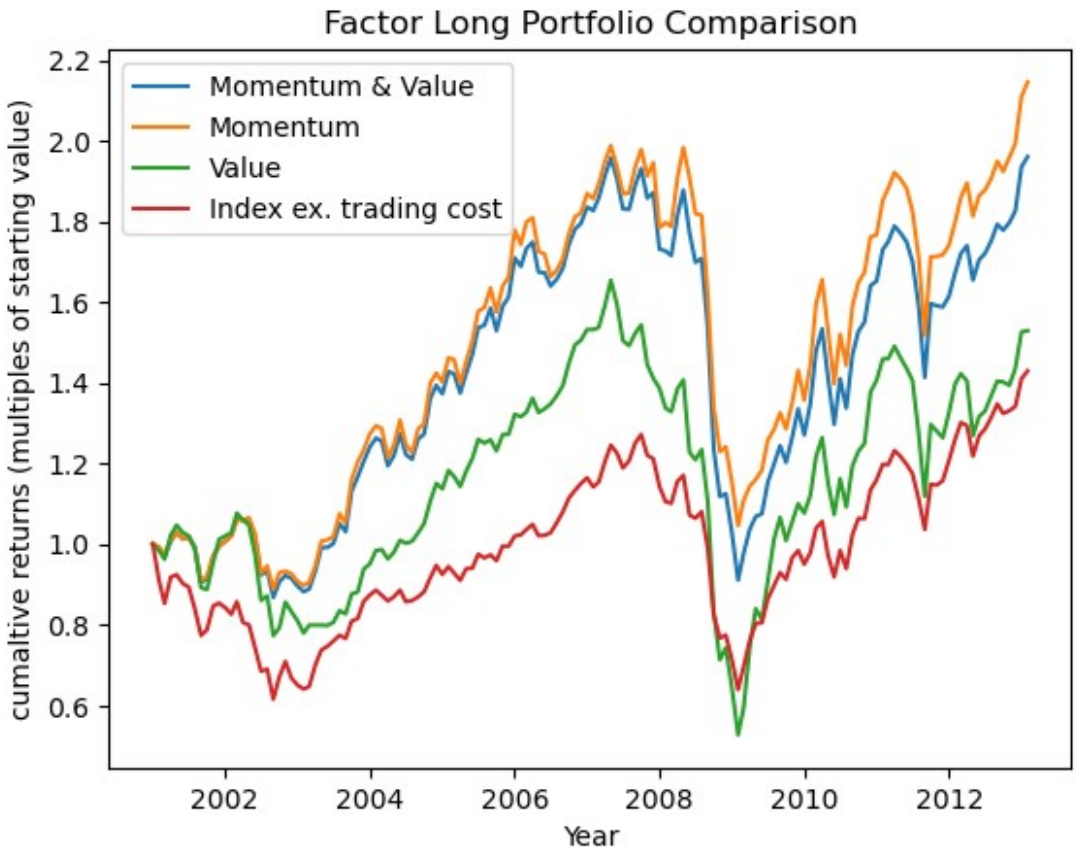
We see that the trecile portfolio actually captures the majority of the benefits. A full portfolio only offers marginal improvements, whilst the decile portfolios actually risk overconcentration and noise, therefor going forwardwe shall keep this value, and the bad long short portfolio performance was **not** due to our choice of this value.

Investigating the result: Long vs Short Portfolios

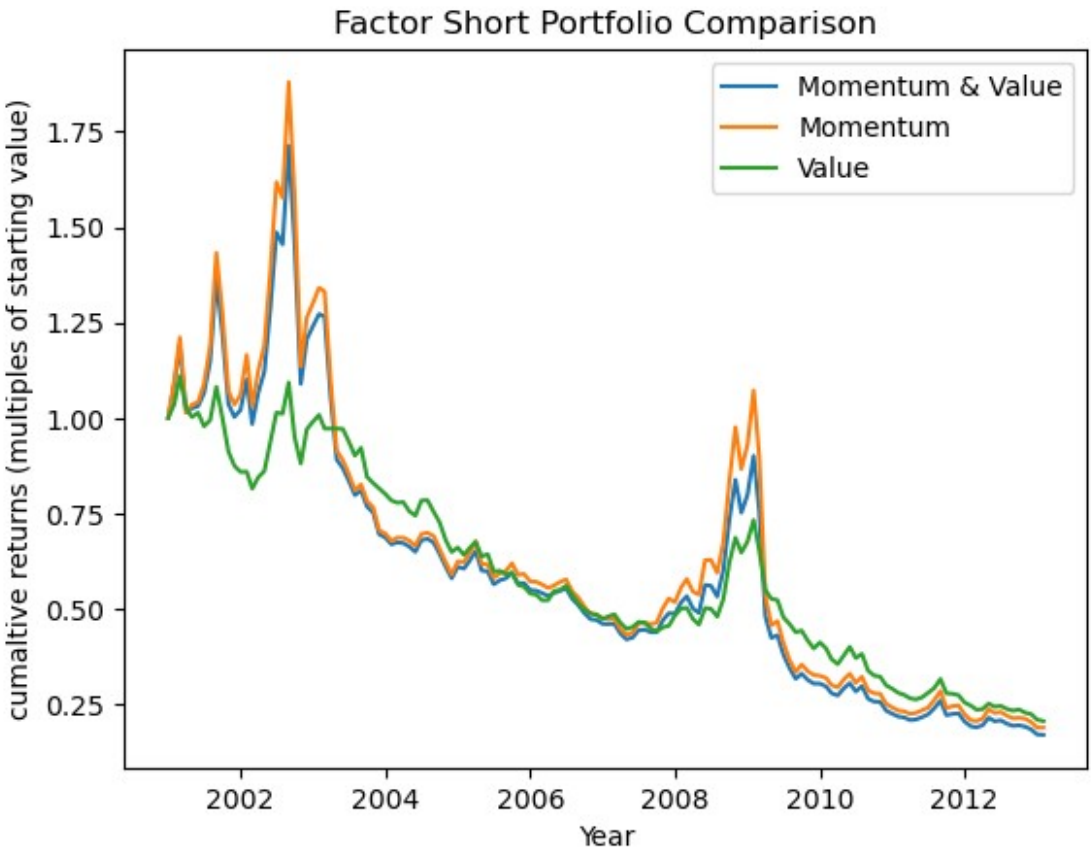
Next, we investigate further the impact of the long and short portfolios. Again, we did not see an outperformance from our momentum and value factors; however, we do see that the factor performance is much stronger on the long side

```
momval_tret, momval_tcret =
port_momval.get_port_ret(weight=port_momval.lw,bps=10)
mom_tret, mom_tcret = port_mom.get_port_ret(weight=port_mom.lw,bps=10)
val_tret, val_tcret = port_val.get_port_ret(weight=port_val.lw,bps=10)

# Index Excluding trading costs
mkt_ret, mkt_tcret =
port_val.get_port_ret(weight=mcap_weight_train,bps=0)

plt.plot(momval_tcret)
plt.plot(mom_tcret)
plt.plot(val_tcret)
plt.plot(mkt_tcret)

plt.legend(["Momentum & Value","Momentum","Value", "Index ex. trading
cost"])
plt.xlabel("Year")
plt.ylabel("cumaltive returns (multiples of starting value)")
plt.title("Factor Long Portfolio Comparison")

res = port_momval.calc_evals(momval_tret,mkt_ret,Feild_Name="Momentum
Value")
res =
port_momval.calc_evals(mom_tret,mkt_ret,res,Feild_Name="Momentum")
res = port_momval.calc_evals(val_tret,mkt_ret,res,Feild_Name="Value")
res = port_momval.calc_evals(mkt_ret,mkt_ret,res,Feild_Name="Index")

print(res)
```

```
                                  Momentum Value Momentum  Value
Index
Annualised Information Ratio                 0.386    0.412  0.157
nan
Annualised Sharpe Ratio (rf = 3%)            0.240    0.288  0.131
0.075
Max Drawdown                                 0.535    0.474  0.681
0.497
Annualised Volatility                        0.162    0.158  0.212
0.156
Annualised Average Return                    0.069    0.076  0.058
0.042

c:\Users\ChunMing\OneDrive - EL Accounting\Documents\VS Code\MSFE HFS-
Project N\code_base.py:90: RuntimeWarning: invalid value encountered
in double_scalars
  ratio = error.mean(skipna=True)/(error.std(skipna=True))
```




```
_, momval_tcret =
port_momval.get_port_ret(weight=port_momval.sw,bps=10)
_, mom_tcret = port_mom.get_port_ret(weight=port_mom.sw,bps=10)
_, val_tcret = port_val.get_port_ret(weight=port_val.sw,bps=10)

plt.plot(momval_tcret)
plt.plot(mom_tcret)
plt.plot(val_tcret)

plt.legend(["Momentum & Value","Momentum","Value"])
plt.xlabel("Year")
plt.ylabel("cumaltive returns (multiples of starting value)")
plt.title("Factor Short Portfolio Comparison")

Text(0.5, 1.0, 'Factor Short Portfolio Comparison')
```



Again, there has been no outperformance from the value momentum portfolio. Furthermore, we can also observe that the short portfolios for all factors have been suffering from losses equal to or greater than their long portfolios; and in times of large drawdowns, the gains from the short portfolios were insufficient to cover for losses from the long portfolios.

this suggests perhaps uniquely from this data set:

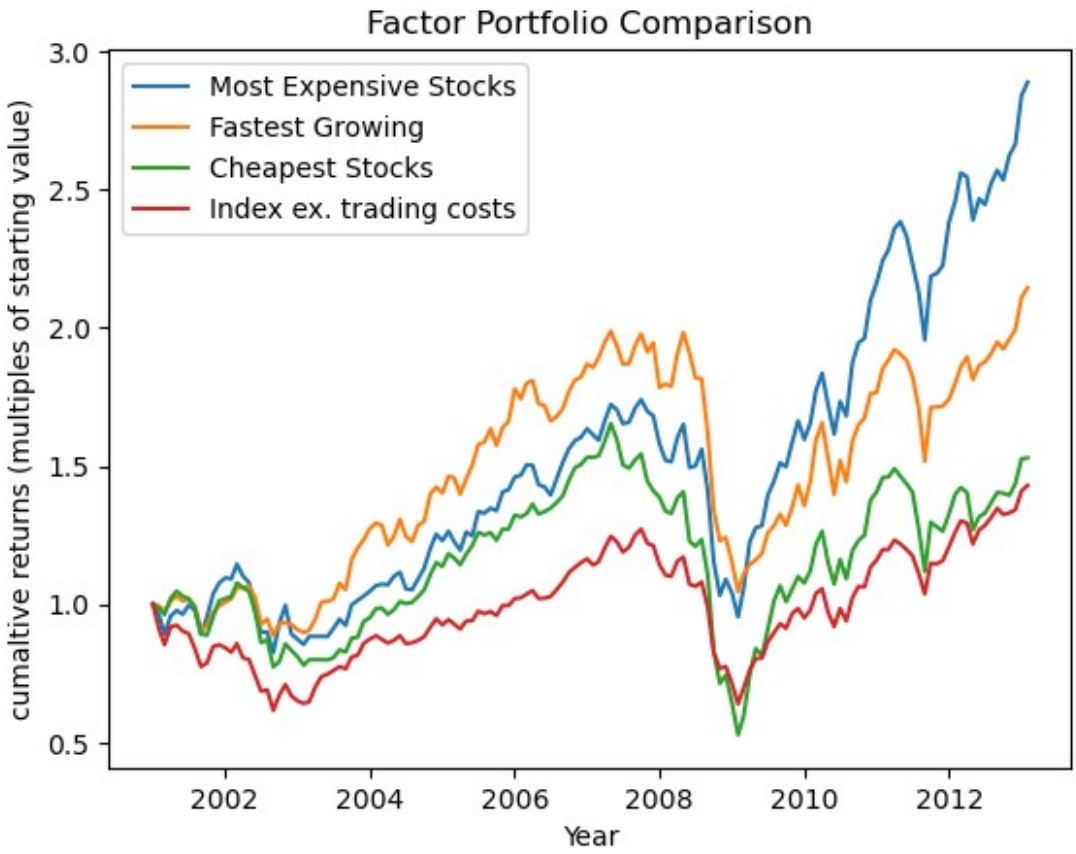
- **From the momentum effects:** this suggests that most stocks in the market have been going up at similar rates regardless of past returns (pointing to a high level of market efficiency during normal periods), whilst in drawdown events, the fastest growing stocks are also drawing down much quicker than low-returning stocks
- **From the value effects:** cheap stocks have consistently underperformed in this universe (this might be the case, as it is a managed large cap index where relatively cheap stocks in the index are not truly "cheap")
- **Most Interestingly:** As can be seen from the figure below, it is the most expensive (**Inverse Momentum**) stocks that have outperformed, both the recently highest returning stocks, as well as the cheapest stocks.

```
# ploting graph
neg_val_ret, val_tcret = port_val.get_port_ret(weight=-
port_val.sw,bps=10)
```

```
_, mom_tcret = port_val.get_port_ret(weight=port_mom.lw,bps=10)
_, val2_tcret = port_val.get_port_ret(weight=port_val.lw,bps=10)

plt.plot(val_tcret)
plt.plot(mom_tcret)
plt.plot(val2_tcret)
plt.plot(mkt_tcret)

plt.legend(["Most Expensive Stocks","Fastest Growing","Cheapest
Stocks", "Index ex. trading costs"])
plt.xlabel("Year")
plt.ylabel("cumaltive returns (multiples of starting value)")
plt.title("Factor Portfolio Comparison")

res = port_momval.calc_evals(neg_val_ret,mkt_ret,Feild_Name="Inverse
Momentum Long")
res = port_momval.calc_evals(mkt_ret,mkt_ret,res,Feild_Name="Index")

print(res)
```

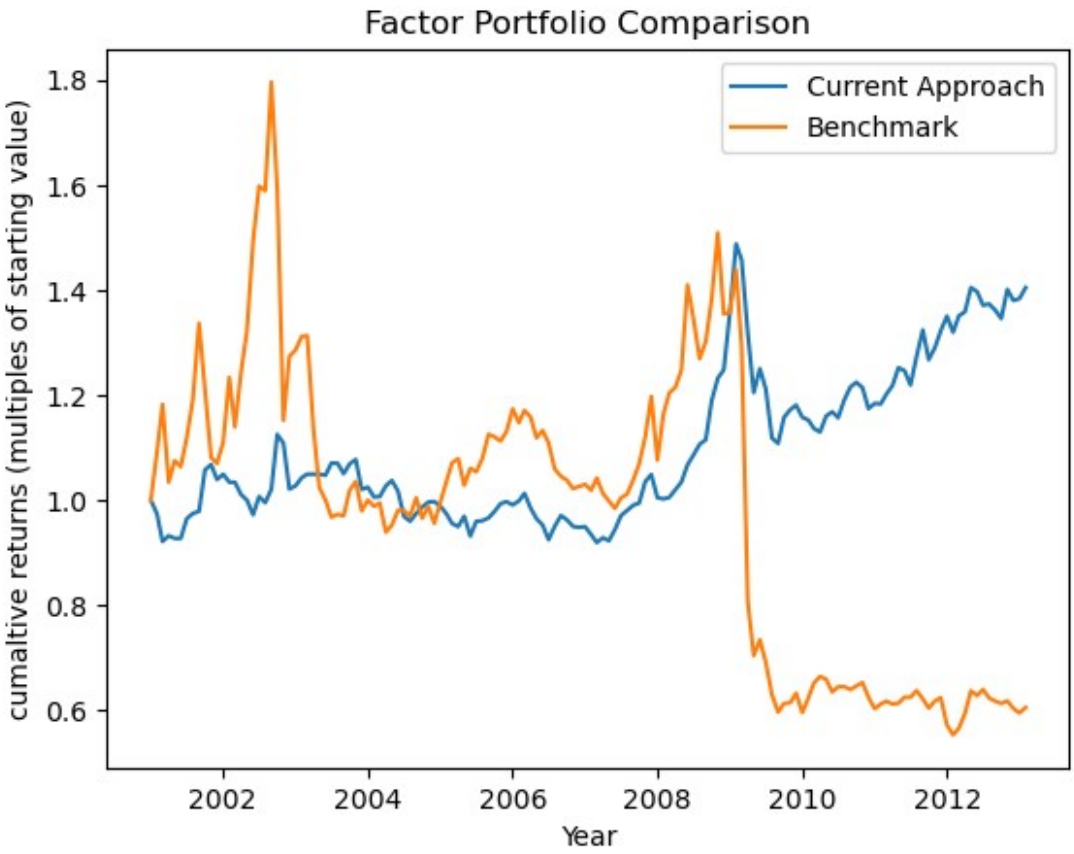
	Inverse Momentum Long	Index
Annualised Information Ratio	0.905	nan
Annualised Sharpe Ratio (rf = 3%)	0.415	0.075
Max Drawdown	0.452	0.497
Annualised Volatility	0.176	0.156
Annualised Average Return	0.103	0.042

```
c:\Users\ChunMing\OneDrive - EL Accounting\Documents\VS Code\MSFE HFS-
Project N\code_base.py:90: RuntimeWarning: invalid value encountered
in double_scalars
  ratio = error.mean(skipna=True)/(error.std(skipna=True))
```



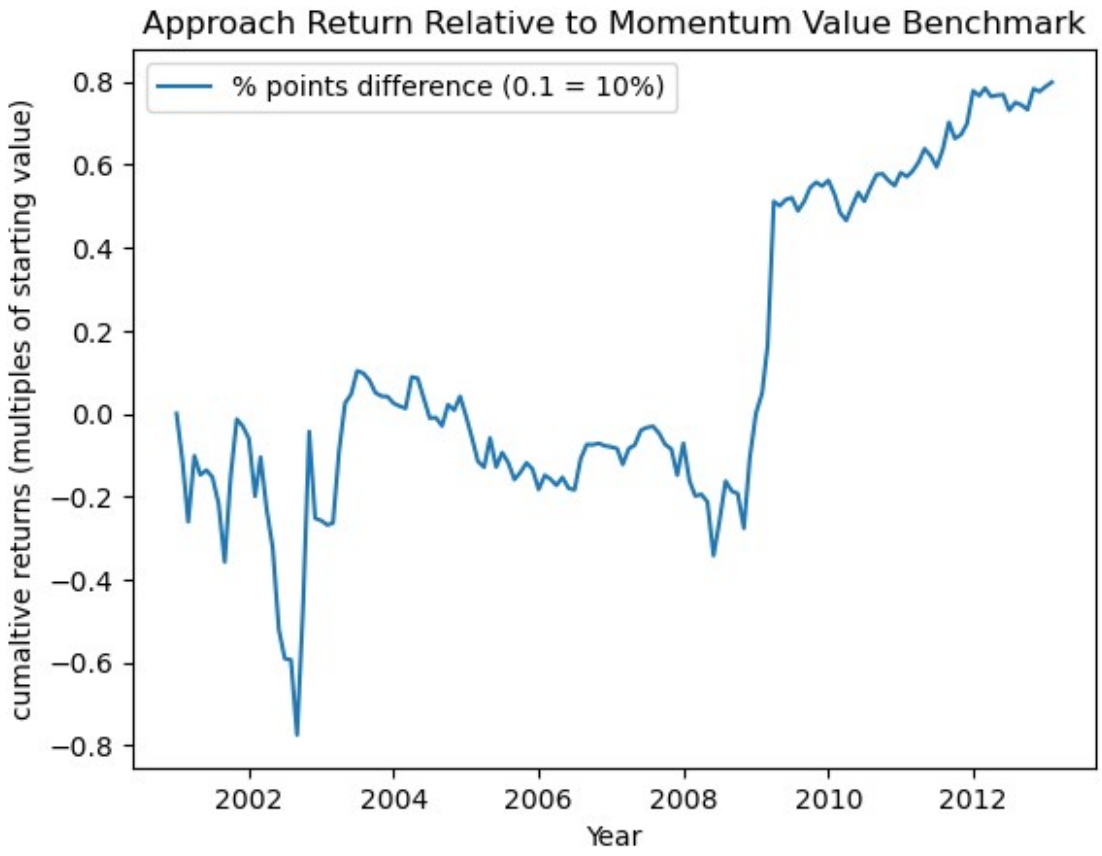
Creating the inverse momentum stratergy and showcasing it's perforamnce

```
tdf = data.get("VAL")
data.append("INV-Z",-tdf)

inv_z_port = data.to_port("INV-Z",tmbool_train)
inv_z_port.gen_weights_from_score(0.3)
inv_z_port.quick_plt_diff(port_mom)
# _,cret=inv_z_port.get_port_ret(inv_z_port.lsw,10)
```




```
                                  Current Approach Benchmark
Annualised Information Ratio                 0.206       nan
Annualised Sharpe Ratio (rf = 3%)            0.030    -0.194
Max Drawdown                                 0.255     0.692
Annualised Volatility                        0.102     0.224
Annualised Average Return                    0.033    -0.013
```

```
c:\Users\ChunMing\OneDrive - EL Accounting\Documents\VS Code\MSFE HFS-
Project N\code_base.py:90: RuntimeWarning: invalid value encountered
in double_scalars
  ratio = error.mean(skipna=True)/(error.std(skipna=True))
```



3.2 Method 1 : Earnings Factors, and Simple Combined Score

To try to incorporate the idiosyncratic effects from earnings and investigate how they affect momentum and value on a monthly frequency, we first investigate a straightforward approach from our earnings growth, earnings estimate, and surprise factors

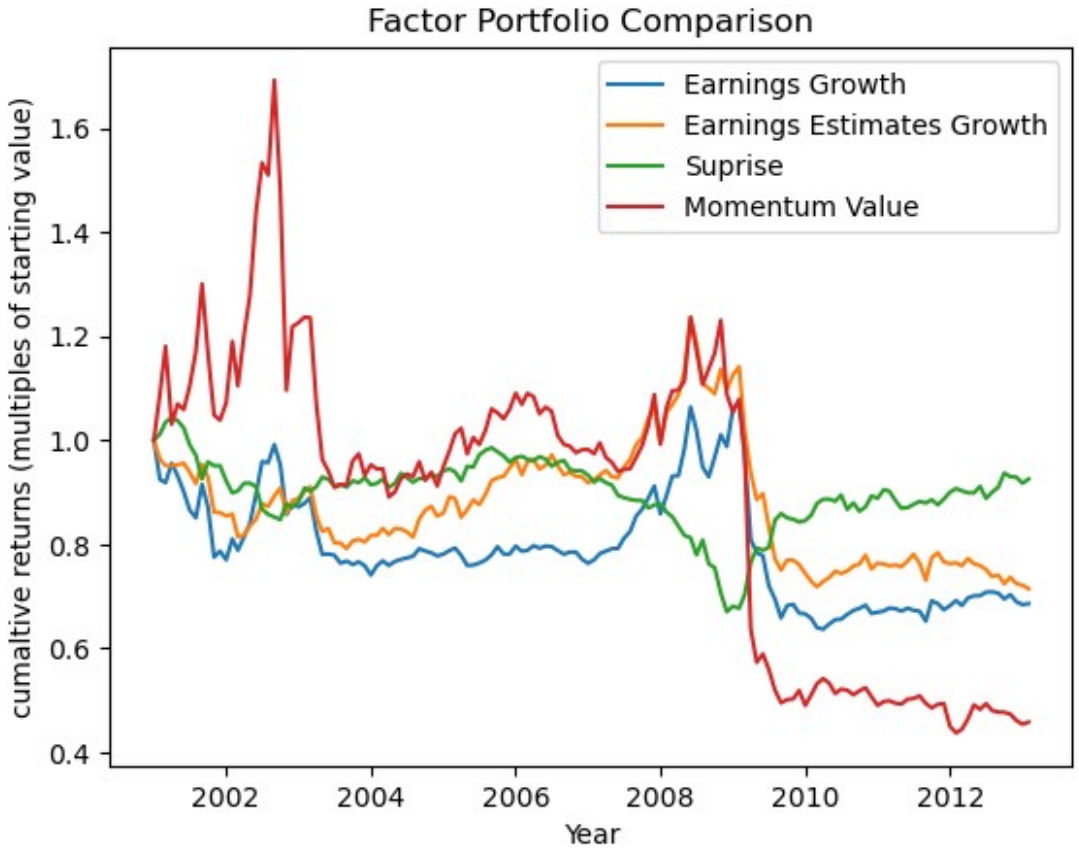
(Note in doing so, we are implicitly disagreeing with the market efficiency hypothesis, where excess returns are only given to the undertaking of risk)

To do so, we follow a 3-step plan 1) construct single-factor portfolios, **inspect for negatively correlated Factor Returns** (this comes from our understanding that the negative factor correlation can complement portfolio returns) 2) **consider if fundamental reasoning behind the negative correlation exists** (for example, Earnings Surprise can serve as additional inputs to whether a momentum trend continues, whilst Earnings Growth and Earnings Estimates Growth can not only do the former but provide information on a value stock's future price) 3) **construct a combined score portfolio** as before

Constructing Simple Factor Portfolios

```
# Earnings Growth Portfolio
port_erg = data.to_port("ERG",tmbool=tmbool_train)
port_erg.gen_weights_from_score(0.3)
_, erg_cret = port_erg.get_port_ret(weight=port_erg.lsw,bps=10)
plt.plot(erg_cret)
```

```
# Earnings Estimates Portfolio
port_etg = data.to_port("ETG",tmbool=tmbool_train)
port_etg.gen_weights_from_score(0.3)
_, etg_cret = port_etg.get_port_ret(weight=port_etg.lsw,bps=10)
plt.plot(etg_cret)

# Earnings Suprise Portfolio
port_sup = data.to_port("SUP-Z",tmbool=tmbool_train)
port_sup.gen_weights_from_score(0.3)
_, sup_cret = port_sup.get_port_ret(weight=-port_sup.lsw,bps=10)
plt.plot(sup_cret)

plt.plot(momval_cret)

plt.legend(["Earnings Growth","Earnings Estimates Growth","Suprise",
"Momentum Value"])
plt.xlabel("Year")
plt.ylabel("cumaltive returns (multiples of starting value)")
plt.title("Factor Portfolio Comparison")

Text(0.5, 1.0, 'Factor Portfolio Comparison')
```



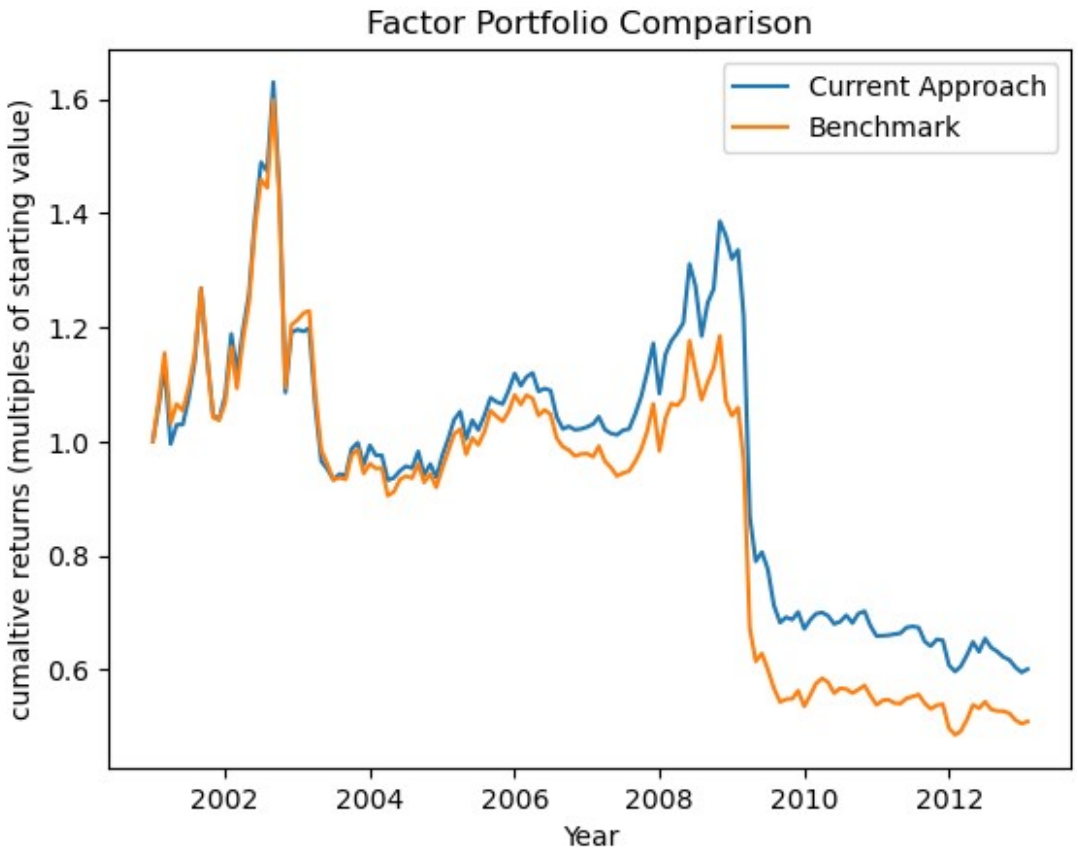
Factors Selection & Creation

Here we see that Earnings estimates growth has much more "information" than Earnings Growth as visible by the out-performance. However Earnings Suprise Factor likely be prefered as it would benefit from having the negative correlation to momentum and value.

There is also some fundamental reason to believe earnings surprise would contain additional information, as it provides information for market expectations. Whilst there are also some reasons to believe that Earnings Estimates Growth (ETG) can provide additional information to momentum and value, they **somewhat weaker** as prices should already reflect investor beliefs; therefore, we expect it to be less likely to hold in the out-of-sample period.

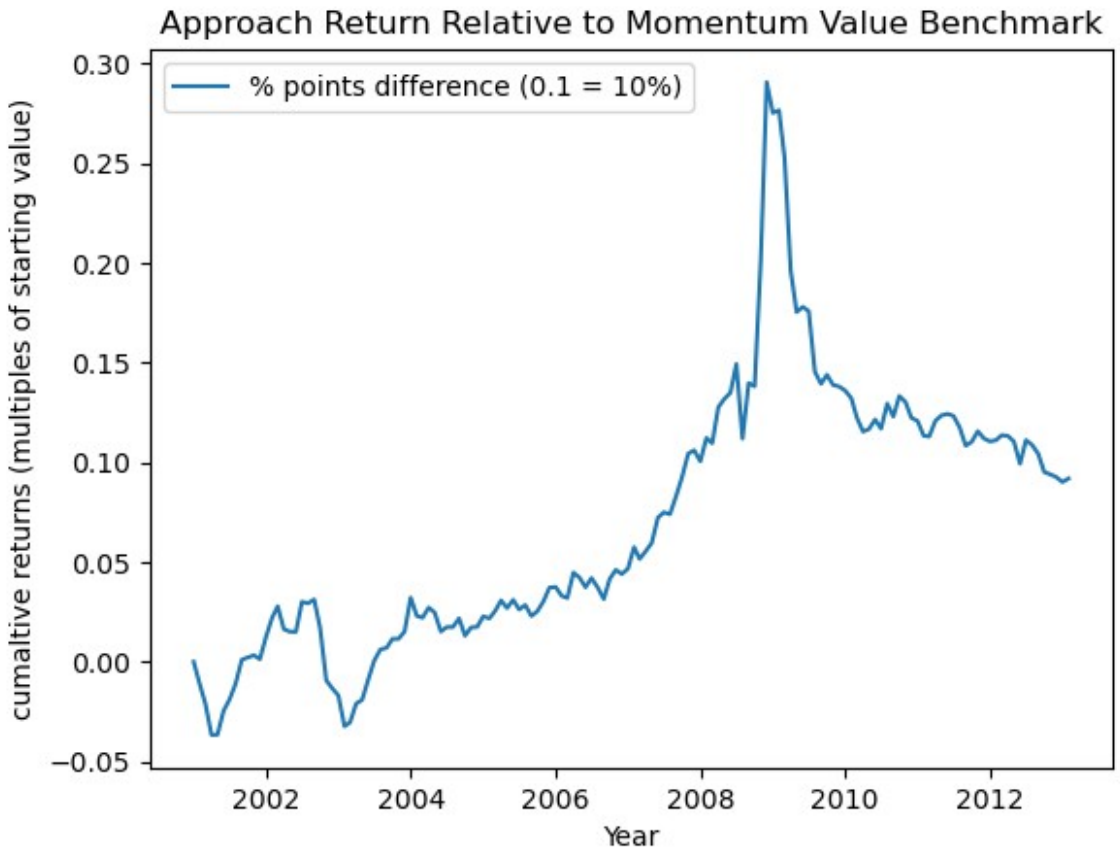
Finally, to create the factor using **SUP**, we adjusted the weighting as we found that an equal weight did not lead to any meaningful impact, therefore leading to a increased final weighting ratio of 0.5, 0.5 & 3

```
# calculate momentum value composite factor
composite = (+0.5*data.get("VAL").fillna(0)
             +0.5*data.get("MOM").fillna(0)
             +3*data.get("SUP-Z").fillna(0))
data.append("COM", composite)

data.calc_z_scores(inplace=True,fields=["COM"])
port_com = data.to_port("COM",tmbool=tmbool_train)
port_com.gen_weights_from_score(0.3)

port_com.quick_plt_diff(port_momval)
```

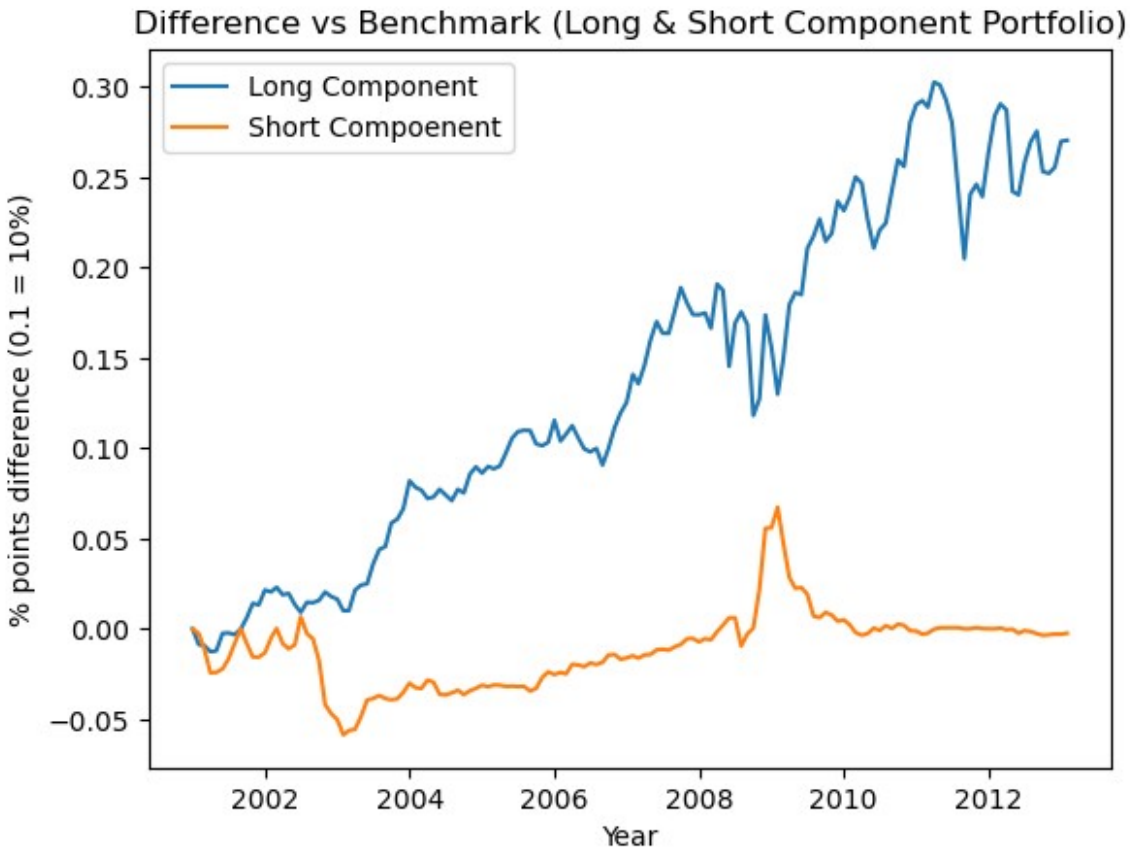



```
                                  Current Approach Benchmark
Annualised Information Ratio                 0.352       nan
Annualised Sharpe Ratio (rf = 3%)           -0.290    -0.365
Max Drawdown                                 0.635     0.696
Annualised Volatility                        0.185     0.184
Annualised Average Return                   -0.023    -0.037
```

```
c:\Users\ChunMing\OneDrive - EL Accounting\Documents\VS Code\MSFE HFS-
Project N\code_base.py:90: RuntimeWarning: invalid value encountered
in double_scalars
  ratio = error.mean(skipna=True)/(error.std(skipna=True))
```



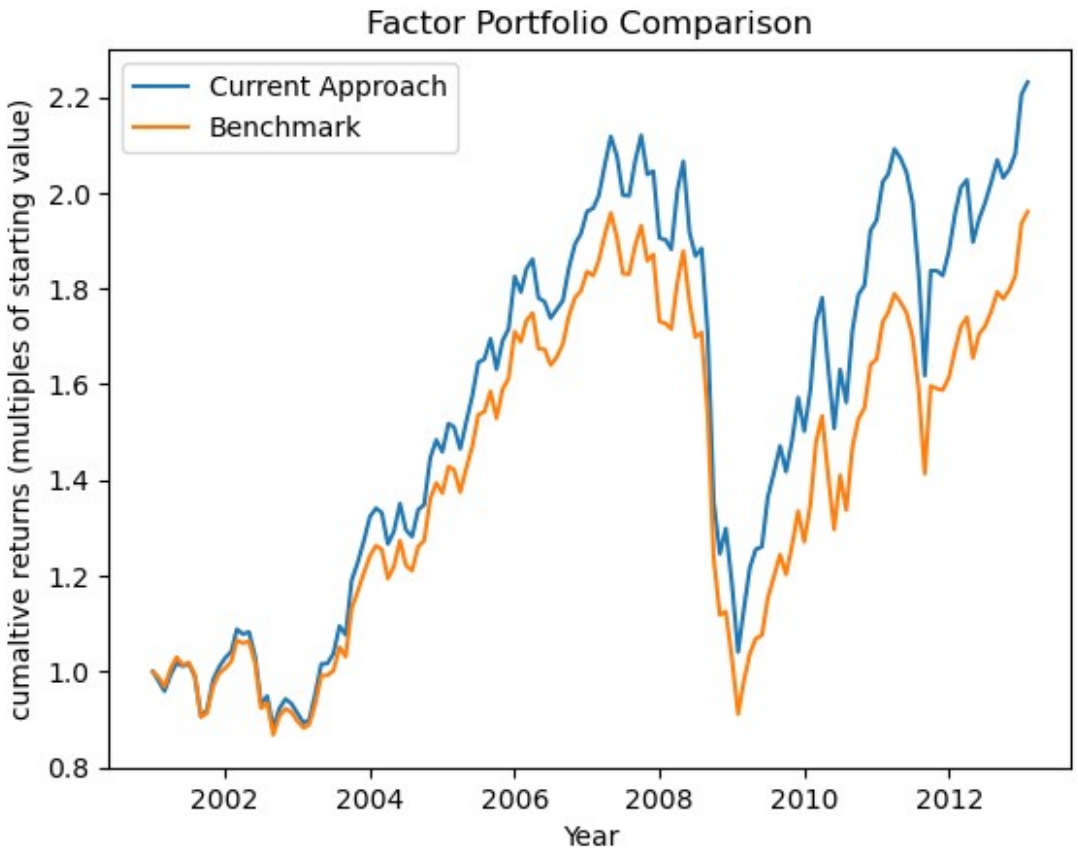

```
port_com.quick_plt_ls(port_momval)
```



This approach led to a positive, but small result. Empirically, we observed that this approach provides substantial alpha relative to our benchmark only up to 2008, and since then, the information provided from earnings surprise has been much less, if trending not negative. Earnings Surprise represents the new idiosyncratic information injected into the market; the reduction in the impact of this information suggests that the market has been more efficient since then.

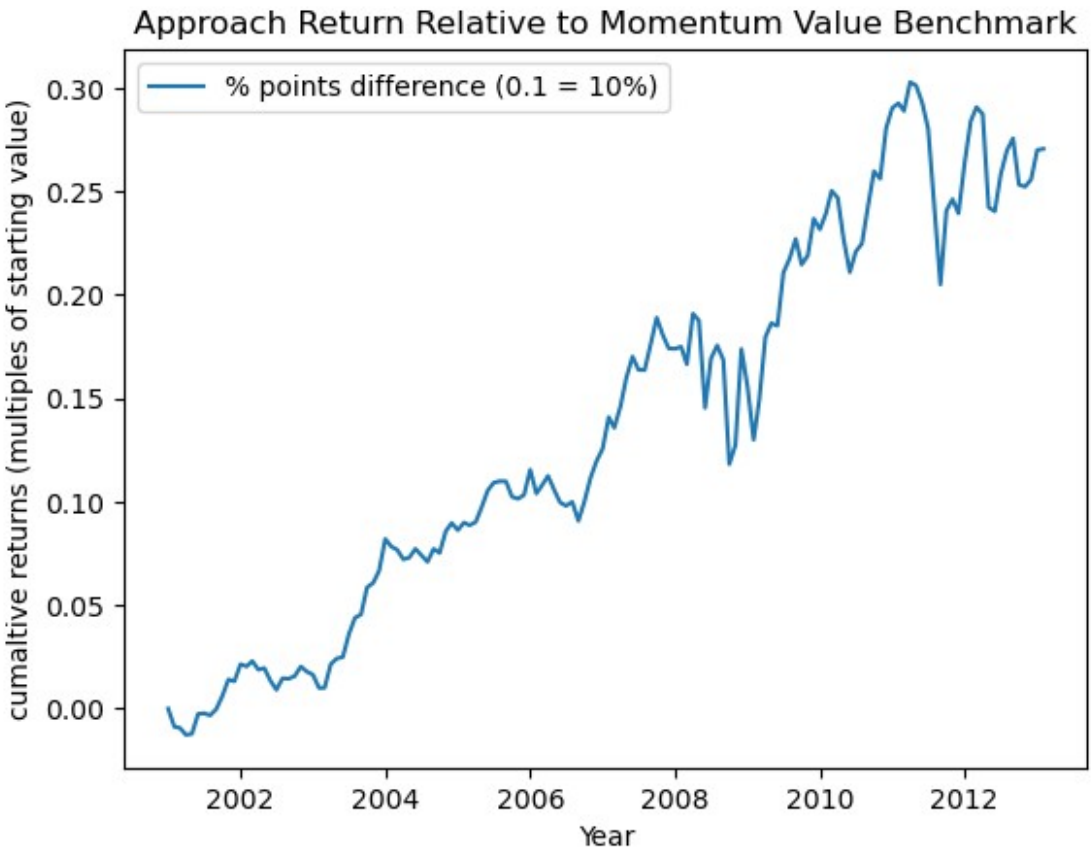
for the **Long Only Portfolio**, we see that the simple score portfolio, not only beat the momentum value, but also the better performing momentum portfolio, suggesting some additional information being introduced

```
## vs Momentum-Value Bench Mark
port_com.quick_plt_diff(port_momval, type = "lw")
```

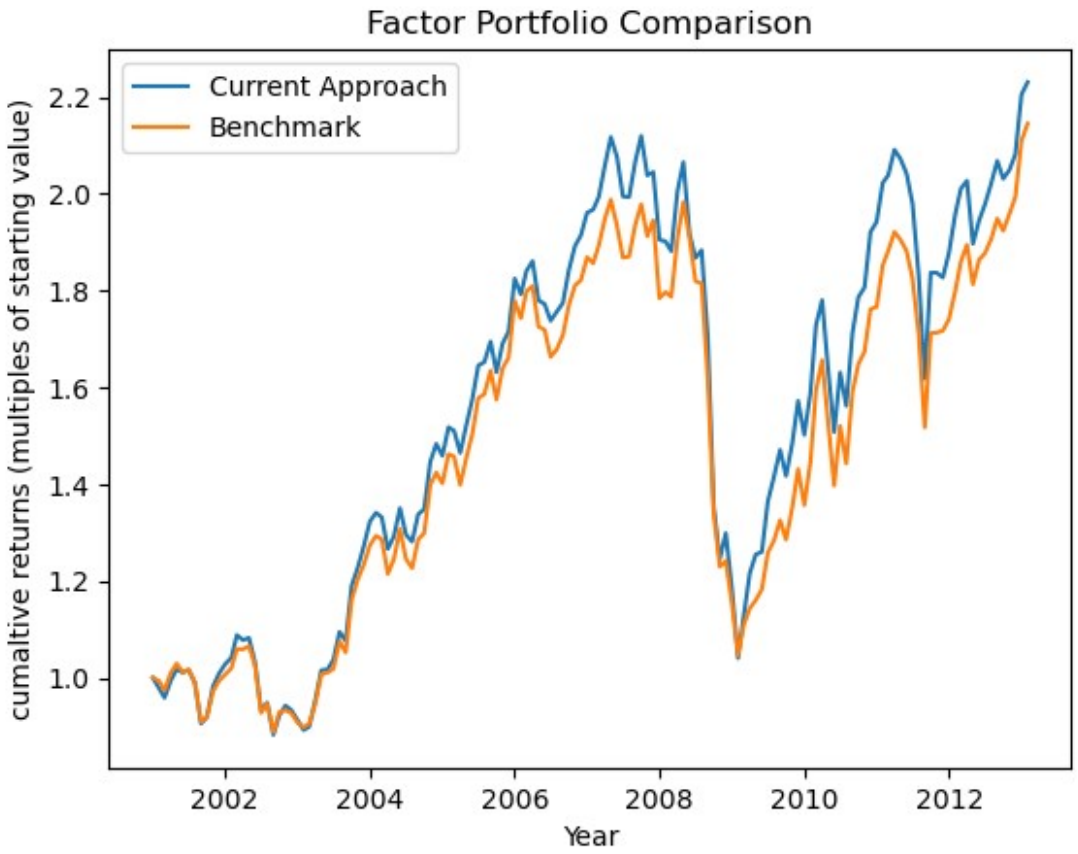


	Current Approach	Benchmark
Annualised Information Ratio	0.543	nan
Annualised Sharpe Ratio (rf = 3%)	0.301	0.240
Max Drawdown	0.509	0.535
Annualised Volatility	0.167	0.162
Annualised Average Return	0.080	0.069

```
c:\Users\ChunMing\OneDrive - EL Accounting\Documents\VS Code\MSFE HFS-
Project N\code_base.py:90: RuntimeWarning: invalid value encountered
in double_scalars
  ratio = error.mean(skipna=True)/(error.std(skipna=True))
```

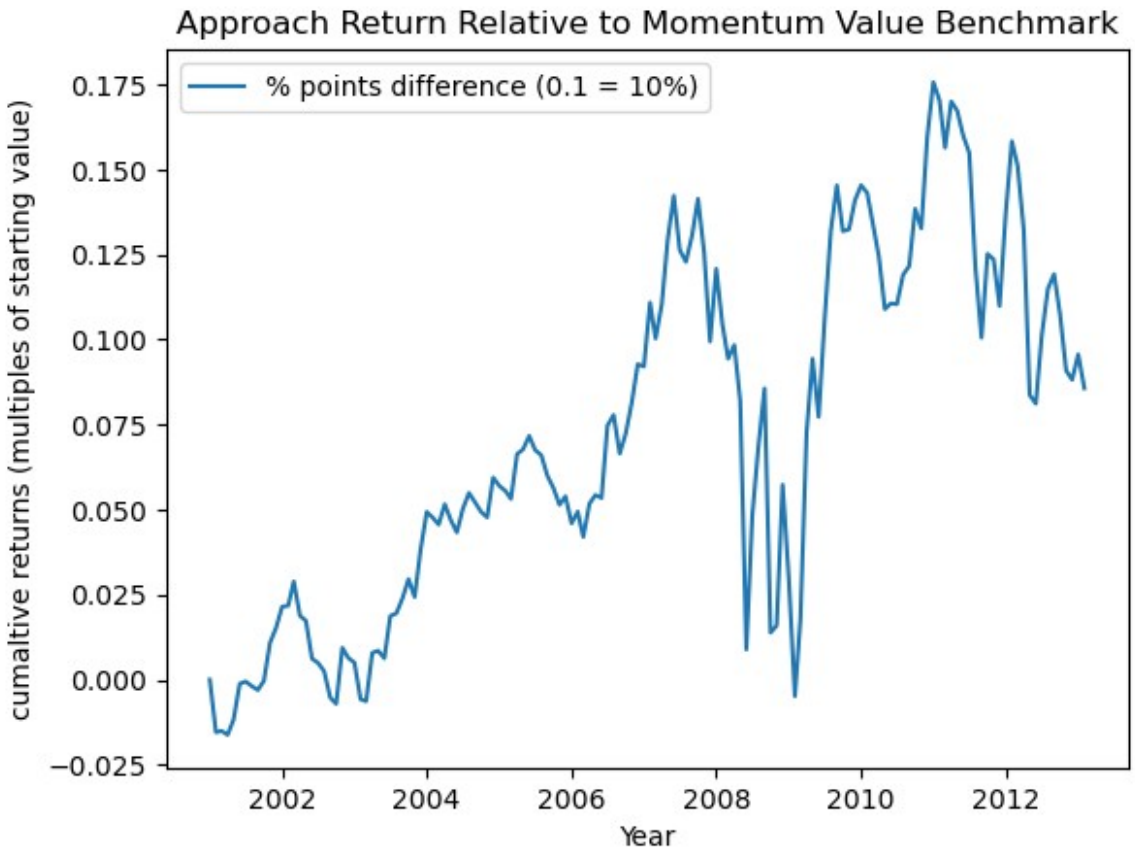



```
## vs Momentum
port_com.quick_plt_diff(port_mom, type = "lw")
```



	Current Approach	Benchmark
Annualised Information Ratio	0.135	nan
Annualised Sharpe Ratio (rf = 3%)	0.301	0.288
Max Drawdown	0.509	0.474
Annualised Volatility	0.167	0.158
Annualised Average Return	0.080	0.076

```
c:\Users\ChunMing\OneDrive - EL Accounting\Documents\VS Code\MSFE HFS-
Project N\code_base.py:90: RuntimeWarning: invalid value encountered
in double_scalars
  ratio = error.mean(skipna=True)/(error.std(skipna=True))
```



3.3 Method 2 : Conditional Factor Weighting

We also attempted another approach. Here, we considered heuristically how the market would react to a large shock from earnings results for the month immediately following earnings. This led to the creation of 3 tables, each with 3 columns (High, Mid, Low Value), and 3 rows (High, Mid, Low Momentum) for each a positive shock, a negative shock, and a neutral shock scenario, which we translate into weights.

For our **base case** where the shock does not cross an earnings surprise percentage threshold of 50, we follow our original momentum value strategy. Due to the delayed nature of our earnings, and the literature indicating that most of the earnings announcement drift occurs around a small window, we to try to consider any secondary effect for the month immediately following.

For the **positive shock case**, we tried to consider potential **mean reversal effects** where we model the effect of the large shock as not well understood, and large positive movements and negative movements are going to revert back to a more reasonable price.

Positive Shock: Score Table |Momentum Value |High|Medium|Low| |:-:|:-:|:-:|:-:| |High| High | Medium | Very Low | |Medium| High| Medium | Low | |Low|Very High | High | Low|

For the **negative shock case**, this is instead a **delayed effects** where an unexpected very large earnings surprise might uncover critical issues, which can be much more serious than a large positive shock.

Positive Shock: Score Table |Momentum Value |High|Medium|Low| |:-:|:-:|:-:|:-:|:-:| |High| Medium | Medium | Negative | |Medium| Negative| Negative | Negative | |Low|Very Negative | Very Negative | Very Negative |

Translating this to weights, this leads to

$$S_{i,t}^{[\text{Base}]} = +0.500 * \text{MOM}_{i,t} + 0.500 * \text{VAl}_{i,t} + 0.00$$

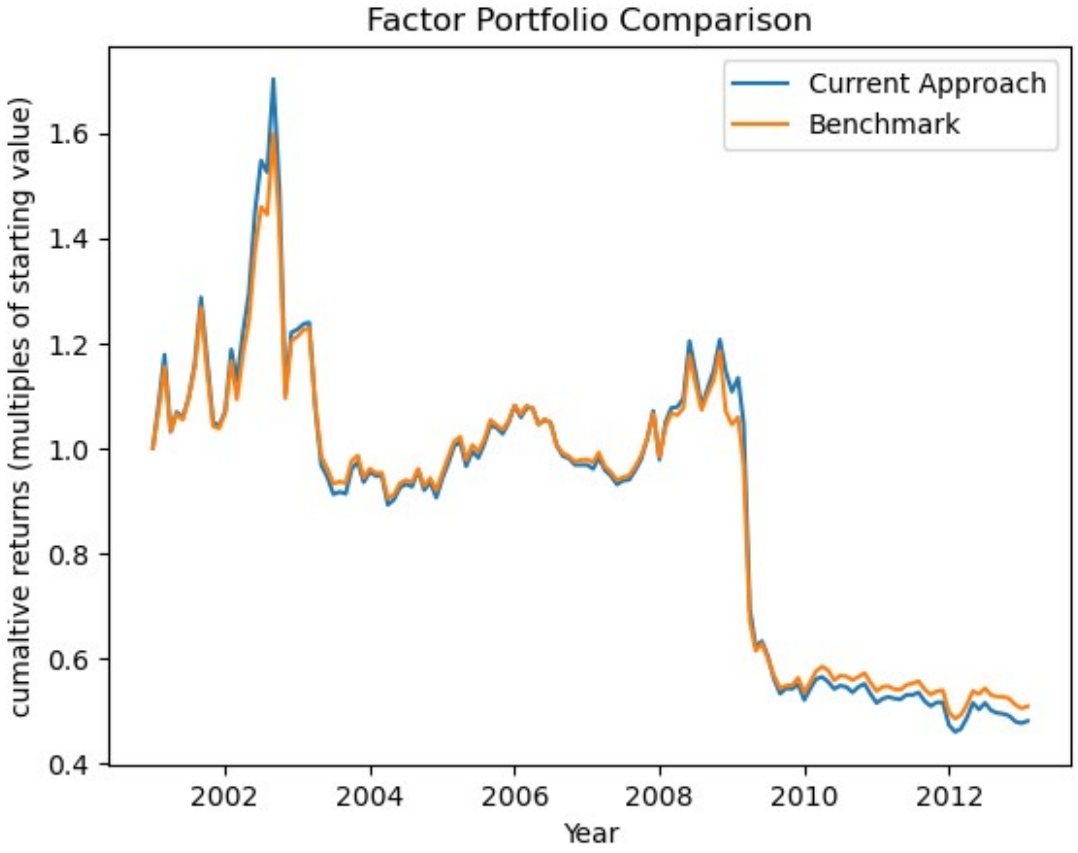
$$S_{i,t}^{[\text{Positive}]} = -0.175 * \text{MOM}_{i,t} + 0.875 * \text{VAl}_{i,t} + 0.00$$

$$S_{i,t}^{[\text{Negative}]} = +0.825 * \text{MOM}_{i,t} - 0.175 * \text{VAl}_{i,t} - 1.75$$

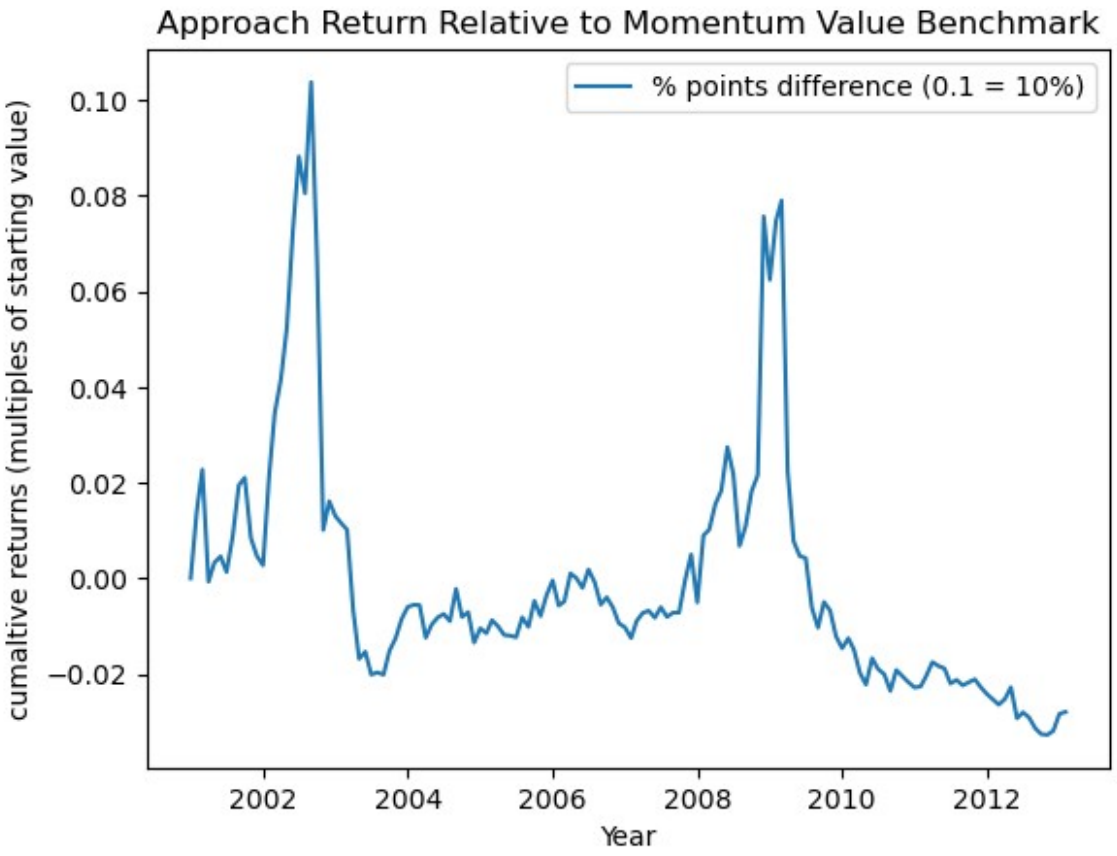
Here we point out that the bias values (intercept) do make an impact as each stock will be under separate earnings shock cases, therefore the overall ranking can be altered by how negative or positive the bias, and weight magnitude are, rather than just their relative ratios

```
try:
    data.data_dict.pop("EJC")
except:
    pass

sup = data.get("SUP")
mom = data.get("MOM")
val = data.get("VAL")

mom.fillna(0,inplace = True)
val.fillna(0,inplace = True)
cuttoff = 0.5

fmb = 0.5
fvb = 0.5
fcb = 0.0

fmp = -0.175
fvp = 0.875
fcp = 0.0

fmn = 0.825
fvn = -0.175
fcn = -1.75


# base case
ejustc = fmb*mom + fvb*val + fcb

# positive case
ejustc[sup>=cuttoff] = fmp*mom[sup>=cuttoff] + fvp*val[sup>=cuttoff] +
fcp

# negative case
```

```
ejustc[sup<= -cuttoff] = fmn*mom[sup<= -cuttoff] + fvn*val[sup<= -
cuttoff] + fcn

data.append("EJC",ejustc)


port_ejc = data.to_port("EJC", tmbool_train)
port_ejc.gen_weights_from_score(0.3)

port_ejc.quick_plt_diff(port_momval)
```




```
                                  Current Approach Benchmark
Annualised Information Ratio                -0.028       nan
Annualised Sharpe Ratio (rf = 3%)           -0.338    -0.365
Max Drawdown                                 0.730     0.696
Annualised Volatility                        0.201     0.184
Annualised Average Return                   -0.038    -0.037

c:\Users\ChunMing\OneDrive - EL Accounting\Documents\VS Code\MSFE HFS-
Project N\code_base.py:90: RuntimeWarning: invalid value encountered
in double_scalars
  ratio = error.mean(skipna=True)/(error.std(skipna=True))
```



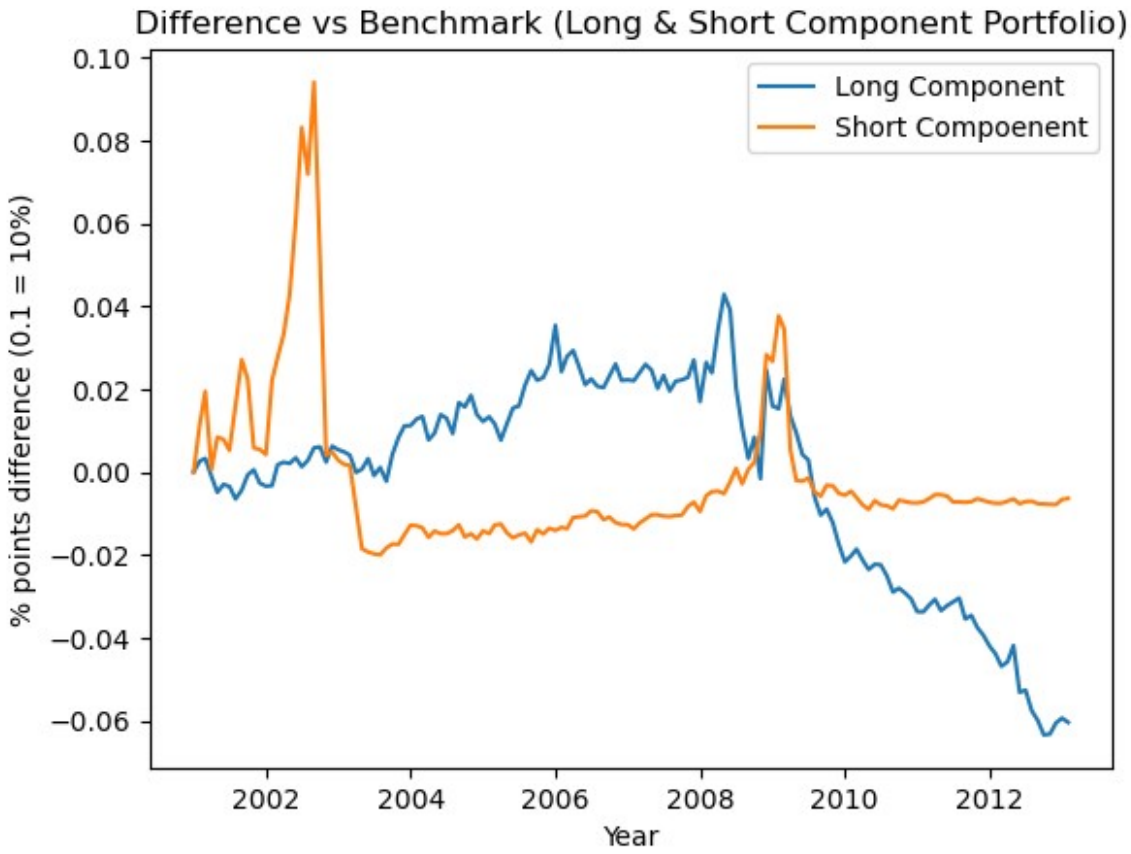
Here we see that this approach largely did not work from a long-term perspective.

Individually from the long component, there is some additional information between early on, but by 2006, it has not provided alpha, and has since completely disappeared, if not negative. Again, supporting the theory that markets have become more efficient.

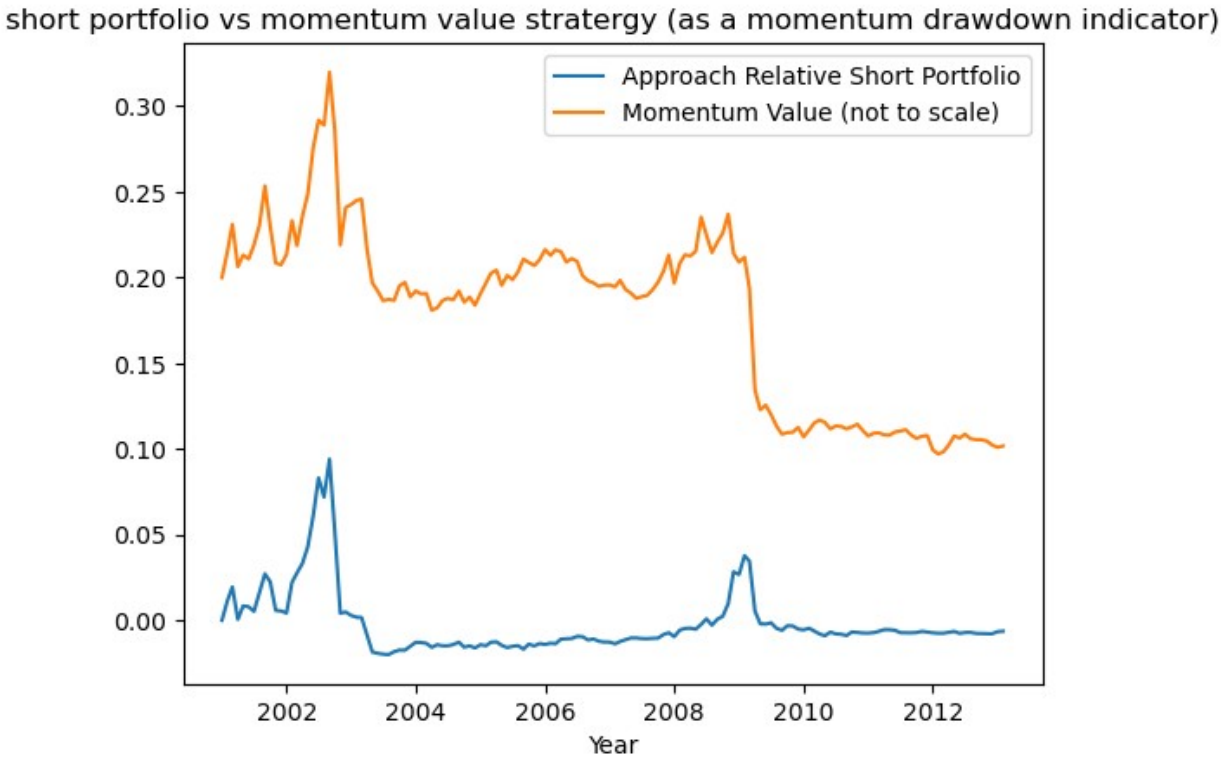
However, very interestingly, from its short component, it had provided a potential leading indicator to both momentum drawdown events where the short portfolio produced increasing positive relative returns (captured information) prior to drawdowns, which can be very useful in a market timing context to avoid momentum drawdowns.

This could make some fundamental sense as the short component (largely from the negative shock case) effectively captures the degree investors are willing to overlook large negative surprises, a measure of market uneasiness

```
port_ejc.quick_plt_ls(port_momval)
```




```
_, ejc_tcret= port_ejc.get_port_ret( port_ejc.sw, bps=10)
_, momval_tcret= port_momval.get_port_ret( port_momval.sw, bps=10)

_, momval_tcret2= port_momval.get_port_ret( port_momval.lsw, bps=10)

plt.plot((ejc_tcret-momval_tcret))
plt.plot((momval_tcret2)/5)
plt.title("short portfolio vs momentum value stratergy (as a momentum
drawdown indicator)")
plt.legend(["Approach Relative Short Portfolio", "Momentum Value (not
to scale)"])
plt.xlabel("Year")

Text(0.5, 0, 'Year')
```



3.4 Method 3 Linear Regression Enabled Method

Lastly, we also try to investigate whether linear regression techniques can improve factor performance by selecting better factor weights.

Here we applied simple least squares regression based on the conditional weights approach in method 3. First, we compile the factor scores into 3 data sets again by earnings surprise percentage threshold. The compiled data set is then used to predict next period returns.

```
sup = data.get("SUP").copy()
ret = data.get("ret").copy().shift(-1)
data.append("RET-ML",ret)

segment = tmbool_train

key = pd.DataFrame(False,index=segment.index,columns=segment.columns)

cuttoff=0.6

key_pos = key.copy()
key_pos[sup>=cuttoff]=True
mldf_pos = data.gen_mldf(["MOM","VAL","RET-ML"],key_pos)

key_neg = key.copy()
key_neg[sup<=-cuttoff]=True
mldf_neg = data.gen_mldf(["MOM","VAL","RET-ML"],key_neg)
```

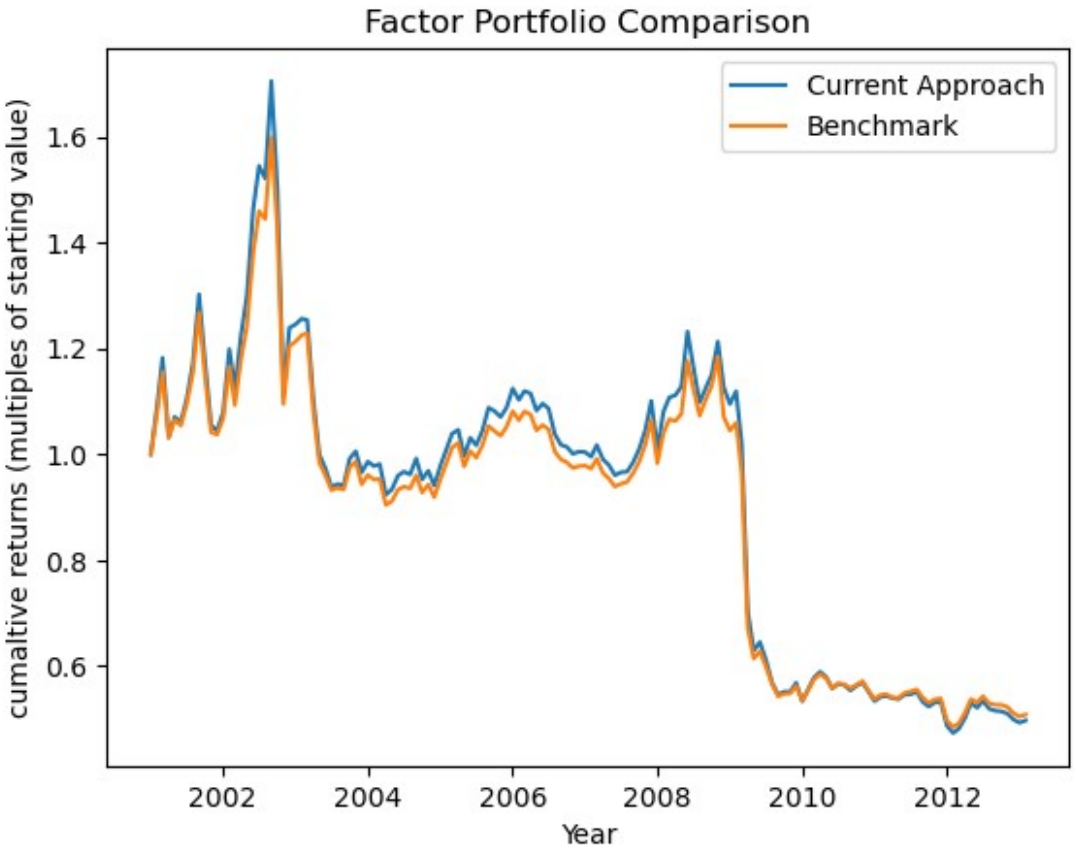
```
key_base = key.copy()
key_base[(sup<cuttoff)&(sup>-cuttoff)]=True
mldf_base = data.gen_mldf(["MOM","VAL","RET-ML"],key_base)


model_pos, res_pos = cb.run_ml(mldf_pos)
model_neg, res_neg = cb.run_ml(mldf_neg)
model_base, res_base = cb.run_ml(mldf_base)
```

Shock Cases

Here we isolate the positive and negative shock cases, by fixing the base case to the simple momentum stratergy to see the effectiveness of the linear regression approach on the earnings shock cases alone

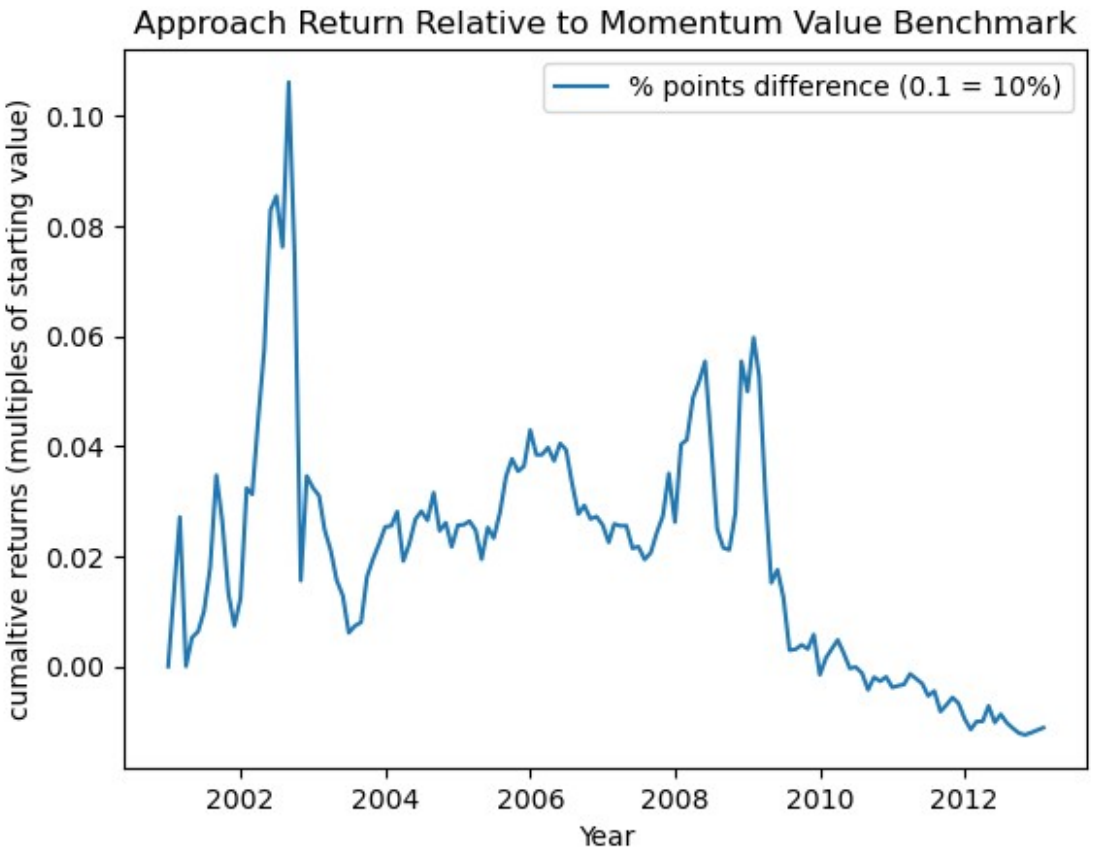
```
try:
    data.data_dict.pop("EJC4")
except:
    pass

sup = data.get("SUP").copy()
mom = data.get("MOM").copy()
val = data.get("VAL").copy()

mom.fillna(0,inplace = True)
val.fillna(0,inplace = True)

fmb = 0.5
fvb = 0.5
fcb = 0.0

fmp = res_pos.params[0]
fvp = res_pos.params[1]
fcp = res_pos.params[2]

fmn = res_neg.params[0]
fvn = res_neg.params[1]
fcn = res_neg.params[2]

# basecase
ejustc4 = fmb*mom + fvb*val + fcb

# positive
ejustc4[sup>=cuttoff] = fmp*mom[sup>=cuttoff] + fvp*val[sup>=cuttoff]
+ fcp

# neg
ejustc4[sup<= -cuttoff] = fmn*mom[sup<= -cuttoff] + fvn*val[sup<= -
```

```
cuttoff] + fcn

data.append("EJC4",ejustc4)
port_ejc4 = data.to_port("EJC4", tmbool_train)
port_ejc4.gen_weights_from_score(0.3)
port_ejc4.quick_plt_diff(port_momval)
```



	Current Approach	Benchmark
Annualised Information Ratio	0.061	nan
Annualised Sharpe Ratio (rf = 3%)	-0.329	-0.365
Max Drawdown	0.722	0.696
Annualised Volatility	0.200	0.184
Annualised Average Return	-0.036	-0.037

```
c:\Users\ChunMing\OneDrive - EL Accounting\Documents\VS Code\MSFE HFS-Project N\code_base.py:90: RuntimeWarning: invalid value encountered in double_scalars
  ratio = error.mean(skipna=True)/(error.std(skipna=True))
```

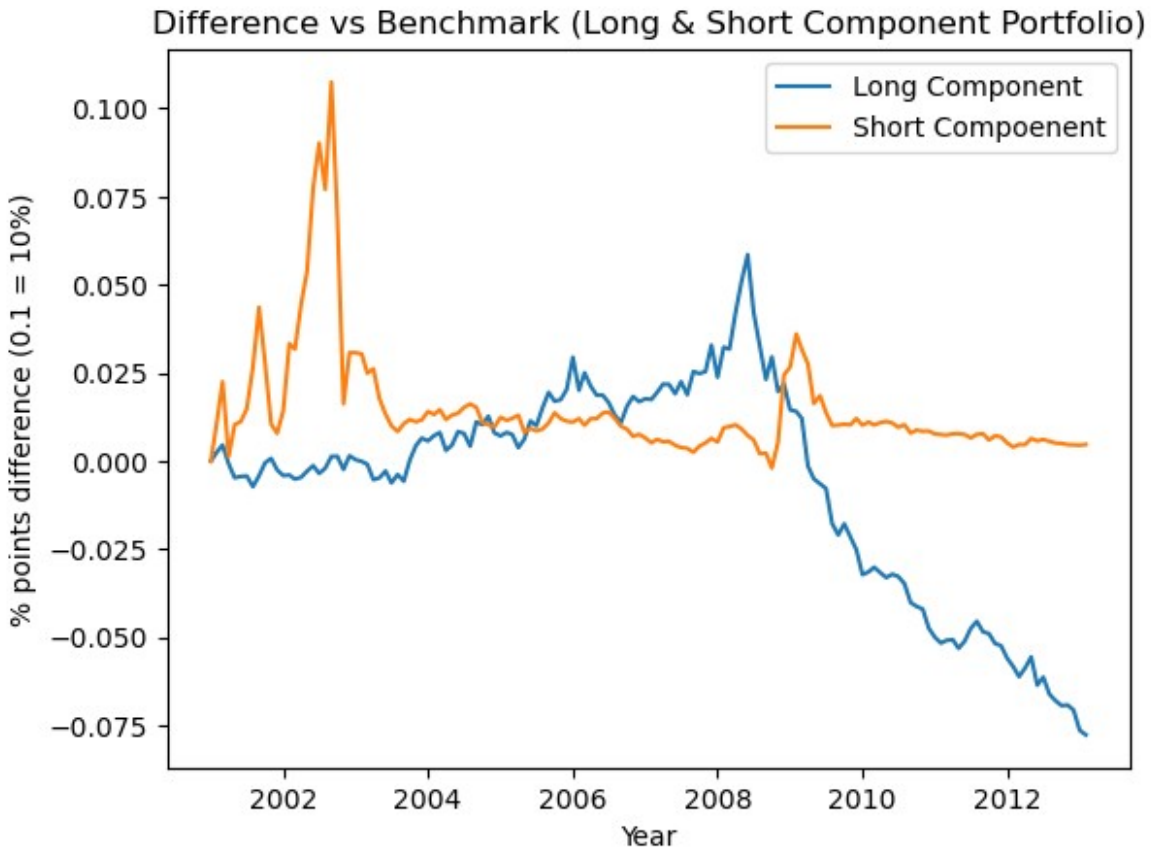


Shock Case Evaluation

Here, whilst we saw a positive information ratio, it is very small, whilst neither the aggregate nor the long short components displayed any noticeable trends or patterns, rendering this approach **largely ineffective**

This is likely due to the highly non-linear nature of the return relationship with scores and a limited number of data points, making a simple linear regression inappropriate without strong fundamental priors

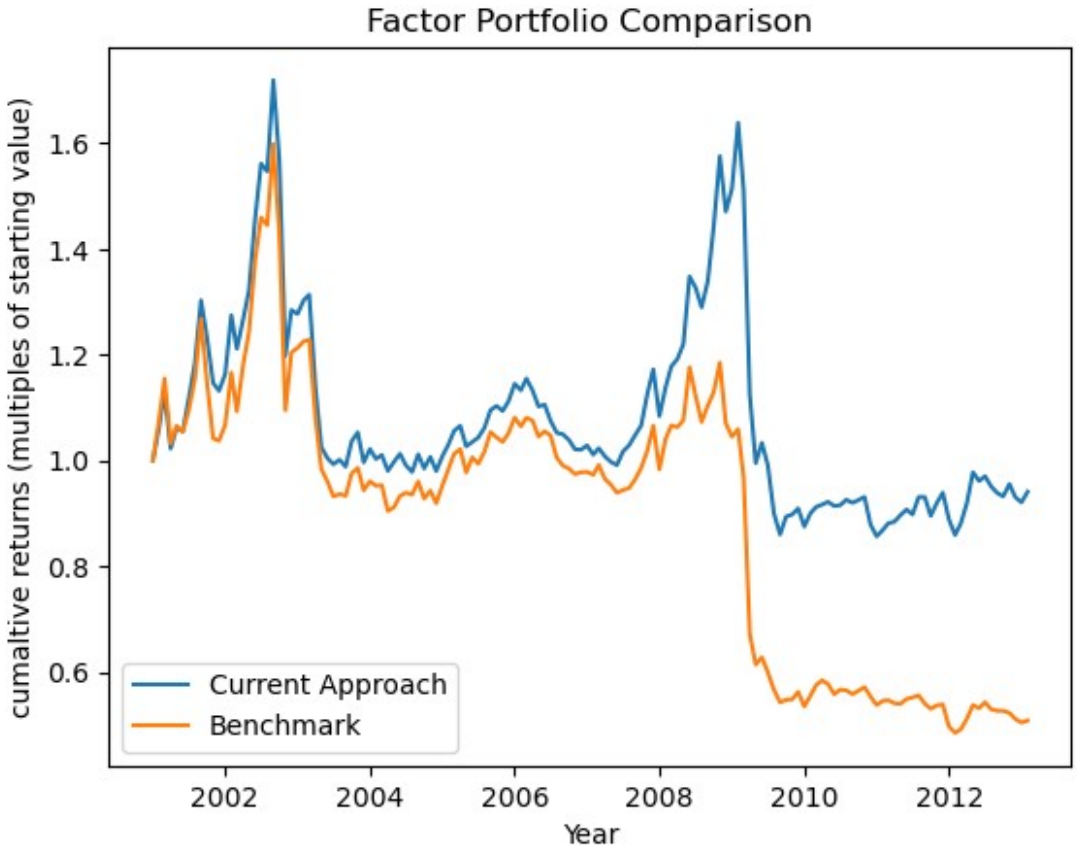
```
port_ejc4.quick_plt_ls(port_momval)
```



perhaps interestingly, we do see linear regresion to select a central base case with more negative value exposure

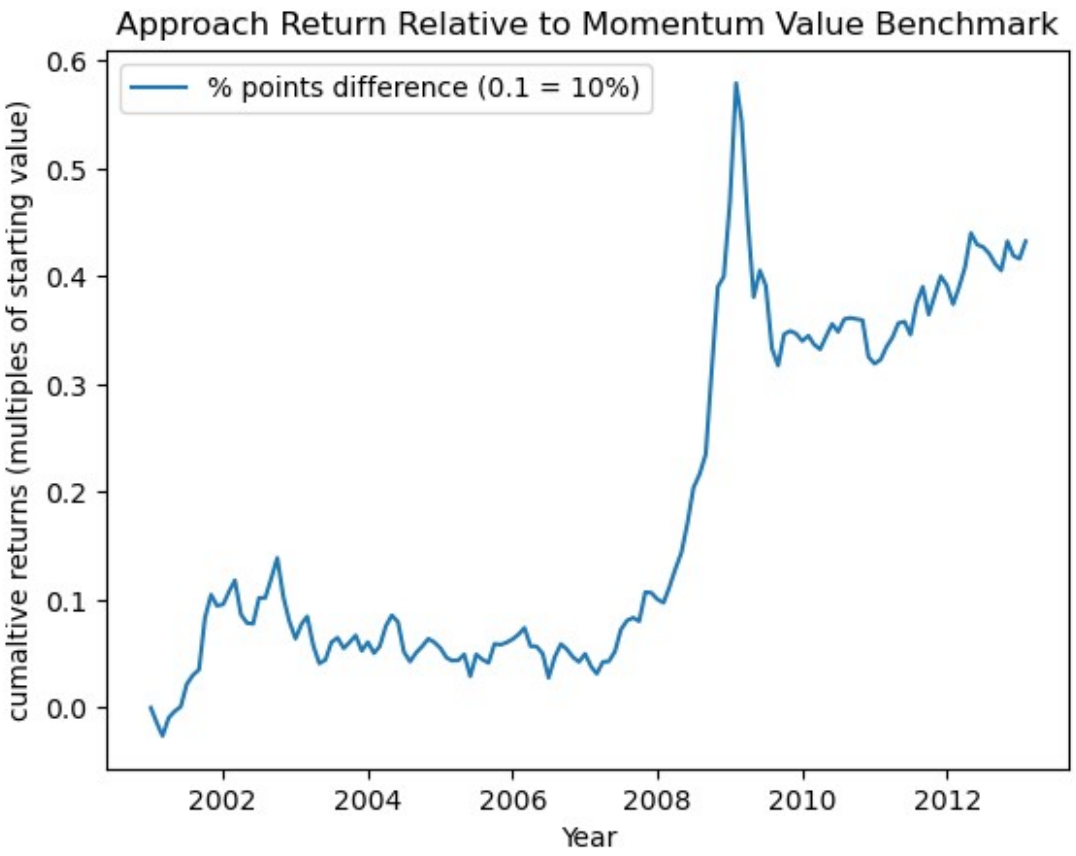
```
try:
    data.data_dict.pop("EJC4-B")
except:
    pass

sup = data.get("SUP").copy()
mom = data.get("MOM").copy()
val = data.get("VAL").copy()

mom.fillna(0,inplace = True)
val.fillna(0,inplace = True)

fmb = res_base.params[0]
fvb = res_base.params[1]
fcb = res_base.params[2]

fmp = 0
fvp = 0
fcp = 0
fmn = 0
```

```
fvn = 0
fcn = 0

# basecase
ejustc4 = fmb*mom + fvb*val + fcb

# positive
ejustc4[sup>=cuttoff] = fmp*mom[sup>=cuttoff] + fvp*val[sup>=cuttoff]
+ fcp

# neg
ejustc4[sup<= -cuttoff] = fmn*mom[sup<= -cuttoff] + fvn*val[sup<= -
cuttoff] + fcn

data.append("EJC4-B",ejustc4)
port_ejc4b = data.to_port("EJC4-B", tmbool_train)
port_ejc4b.gen_weights_from_score(0.3)
port_ejc4b.quick_plt_diff(port_momval)
```



	Current Approach	Benchmark
Annualised Information Ratio	0.849	nan
Annualised Sharpe Ratio (rf = 3%)	-0.100	-0.365
Max Drawdown	0.502	0.696

```
Annualised Volatility                          0.179     0.184
Annualised Average Return                      0.012    -0.037

c:\Users\ChunMing\OneDrive - EL Accounting\Documents\VS Code\MSFE HFS-
Project N\code_base.py:90: RuntimeWarning: invalid value encountered
in double_scalars
  ratio = error.mean(skipna=True)/(error.std(skipna=True))
```



3.5 Out of Sample Evaluation

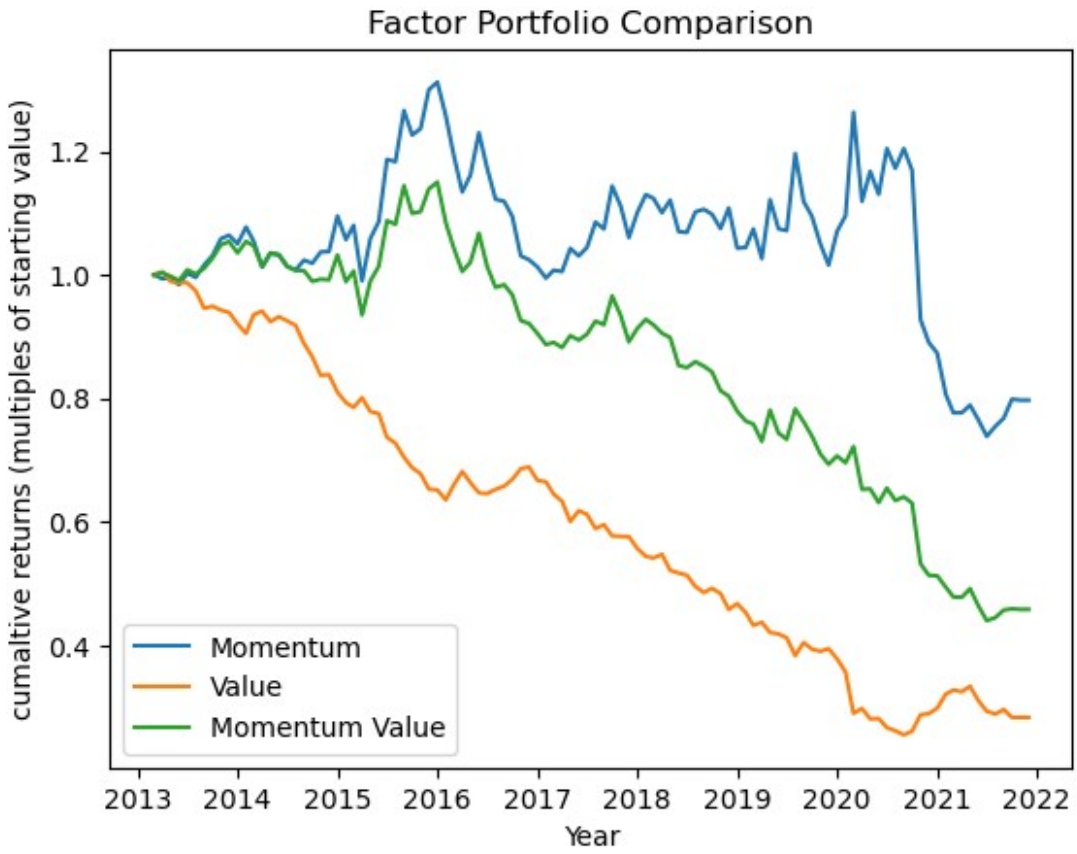
So far, we have investigated three different approaches. Now we will consider their out-of-sample performance, before drawing final conclusions

Benchmark

For our Long Short Momentum Value Portfolio, we once again see that the significantly underperformed the market-cap weighted index, as well as the momentum factor. This can be largely attributable to the very negative characteristics of the value factor as observed and discussed before.

Now different from before all factors underperfrom even for our their Long Portfolios

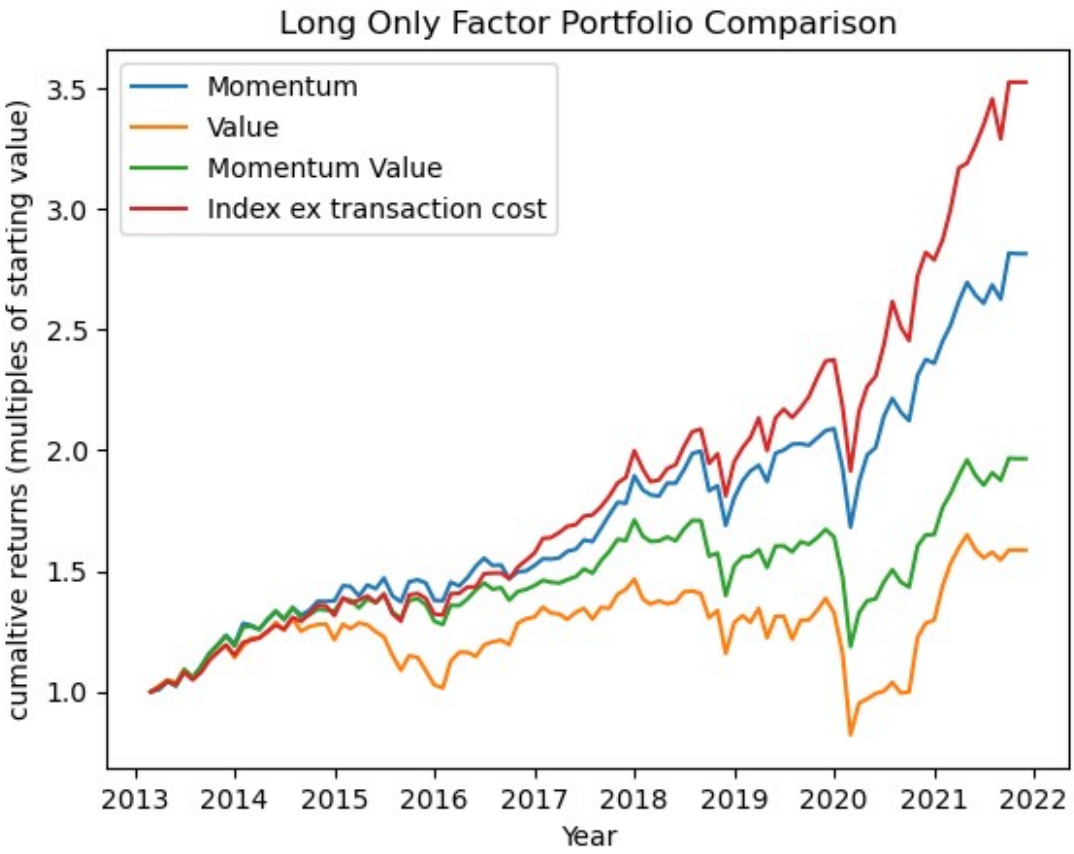
```
test_port_mom = data.to_port("MOM",tmbool=tmbool_test)
test_port_mom.gen_weights_from_score(0.3)
mom_test_rest, mom_test_crest =
test_port_mom.get_port_ret(weight=test_port_mom.lsw,bps=10)

test_port_val = data.to_port("VAL",tmbool=tmbool_test)
test_port_val.gen_weights_from_score(0.3)
val_test_rest, val_test_crest =
test_port_val.get_port_ret(weight=test_port_val.lsw,bps=10)

test_port_momval = data.to_port("MOM_VAL",tmbool=tmbool_test)
test_port_momval.gen_weights_from_score(0.3)
momval_test_rest, momval_test_crest =
test_port_momval.get_port_ret(weight=test_port_momval.lsw,bps=10)

plt.plot(mom_test_crest)
plt.plot(val_test_crest)
plt.plot(momval_test_crest)


plt.legend(["Momentum","Value","Momentum Value"])
plt.xlabel("Year")
plt.ylabel("cumaltive returns (multiples of starting value)")
plt.title("Factor Portfolio Comparison")


factor_ret = pd.DataFrame(index = tmbool.index, columns =
["MOM","VAL"])
factor_ret.loc[tmbool_train.index,"MOM"] = mom_ret
factor_ret.loc[tmbool_train.index,"VAL"] = val_ret
factor_ret.loc[tmbool_test.index,"MOM"] = mom_test_rest
factor_ret.loc[tmbool_test.index,"VAL"] = val_test_rest
factor_ret["Intercept"] = 1

mkt_test_ret, mkt_test_cret  =
test_port_mom.get_port_ret(mcap_weight_test,0)
```




```
# ploting Long only portfolios

mom_test_rest, mom_test_crest =
test_port_mom.get_port_ret(weight=test_port_mom.lw,bps=10)

val_test_rest, val_test_crest =
test_port_val.get_port_ret(weight=test_port_val.lw,bps=10)

momval_test_rest, momval_test_crest =
test_port_momval.get_port_ret(weight=test_port_momval.lw,bps=10)

plt.plot(mom_test_crest)
plt.plot(val_test_crest)
plt.plot(momval_test_crest)
plt.plot(mkt_test_cret)

plt.legend(["Momentum","Value","Momentum Value", "Index ex transaction
cost"])
plt.xlabel("Year")
plt.ylabel("cumaltive returns (multiples of starting value)")
plt.title("Long Only Factor Portfolio Comparison")
```

```
res =
port_momval.calc_evals(mom_test_rest,mkt_test_ret,Feild_Name="Momentum
")
res =
port_momval.calc_evals(val_test_rest,mkt_test_ret,res,Feild_Name="Valu
e")
res =
port_momval.calc_evals(momval_test_rest,mkt_test_ret,res,Feild_Name="M
omentum Value")
res =
port_momval.calc_evals(mkt_test_ret,mkt_test_ret,res,Feild_Name="Index
")

print(res)
```

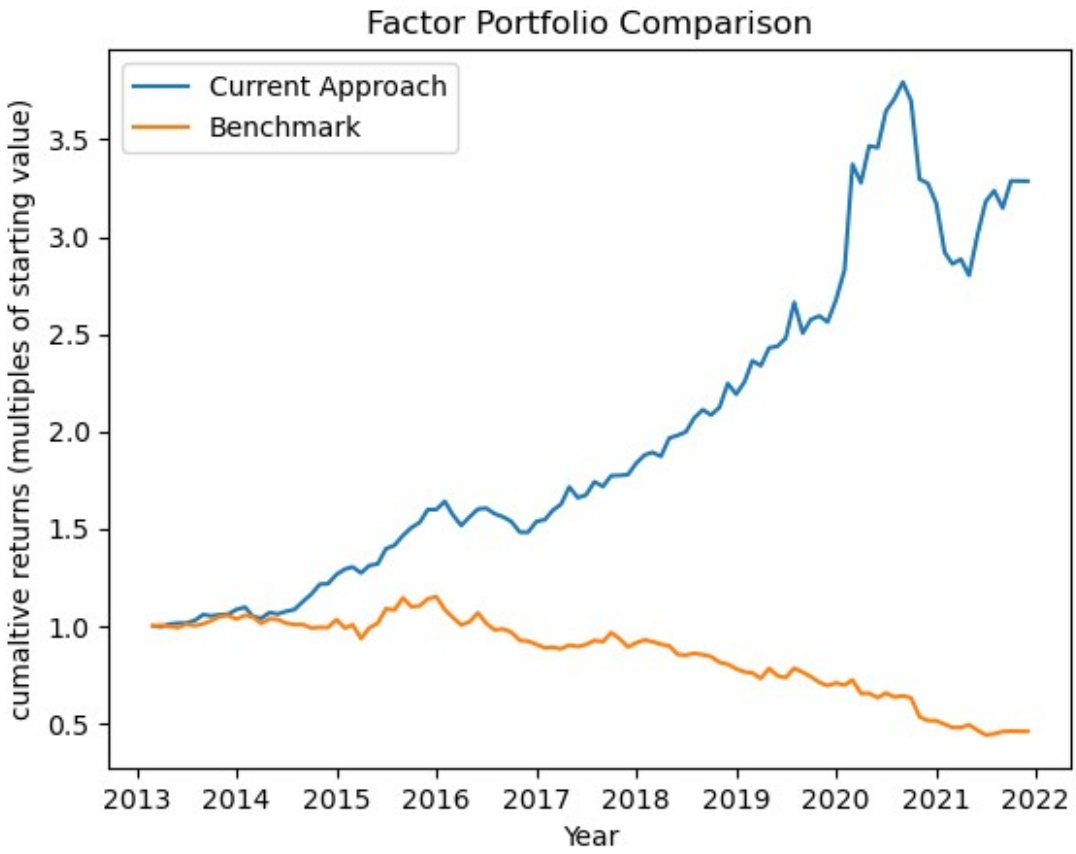
```
                                   Momentum   Value Momentum Value
Index
Annualised Information Ratio         -0.515  -0.749         -1.126
nan
Annualised Sharpe Ratio (rf = 3%)     0.756   0.215          0.398
0.919
Max Drawdown                          0.195   0.440          0.307
0.194
Annualised Volatility                 0.127   0.198          0.144
0.133
Annualised Average Return             0.126   0.073          0.087
0.152

c:\Users\ChunMing\OneDrive - EL Accounting\Documents\VS Code\MSFE HFS-
Project N\code_base.py:90: RuntimeWarning: invalid value encountered
in double_scalars
  ratio = error.mean(skipna=True)/(error.std(skipna=True))
```



Inverse Momentum

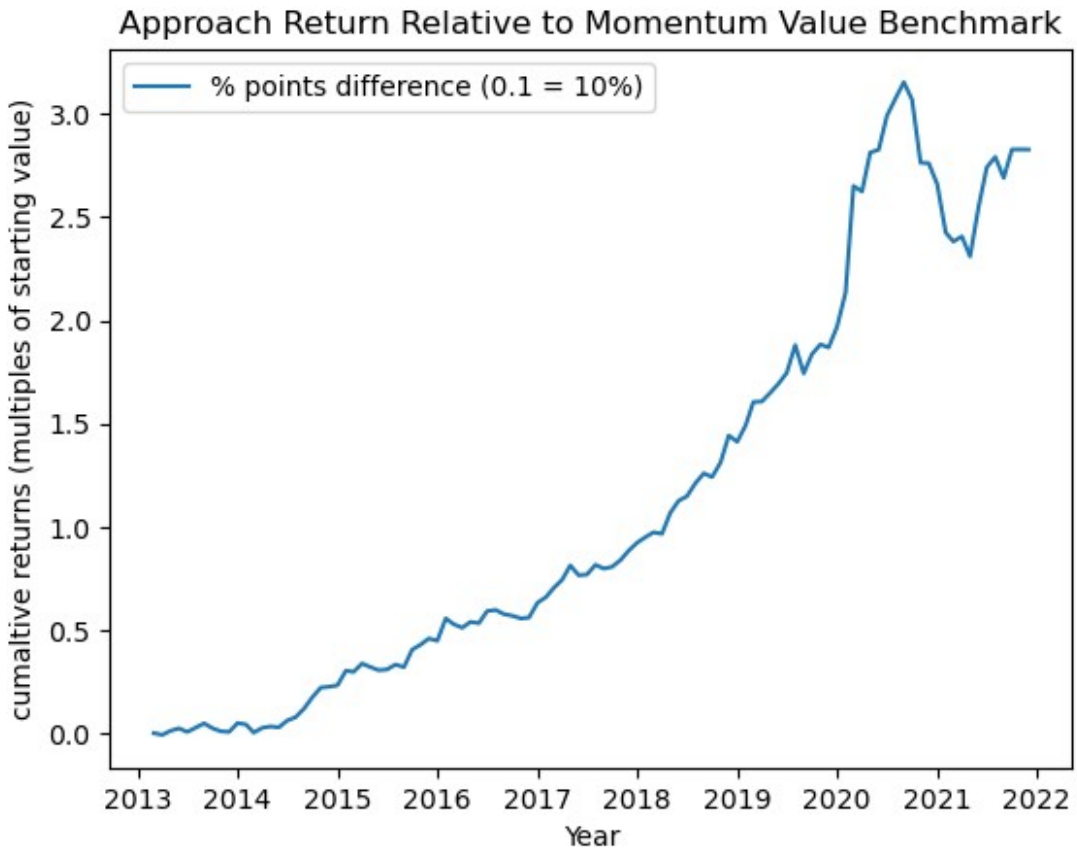
inverse momentum is again by far the most dominant especially in the out of sample period, for both the long and short weights, and long only, being on par (and signifcantly better for long only) on a Sharpe Ratio basis

```
test_port_inv = data.to_port("INV-Z",tmbool_test)
test_port_inv.gen_weights_from_score(0.3)
test_port_inv.quick_plt_diff(test_port_momval,type="lsw")
```

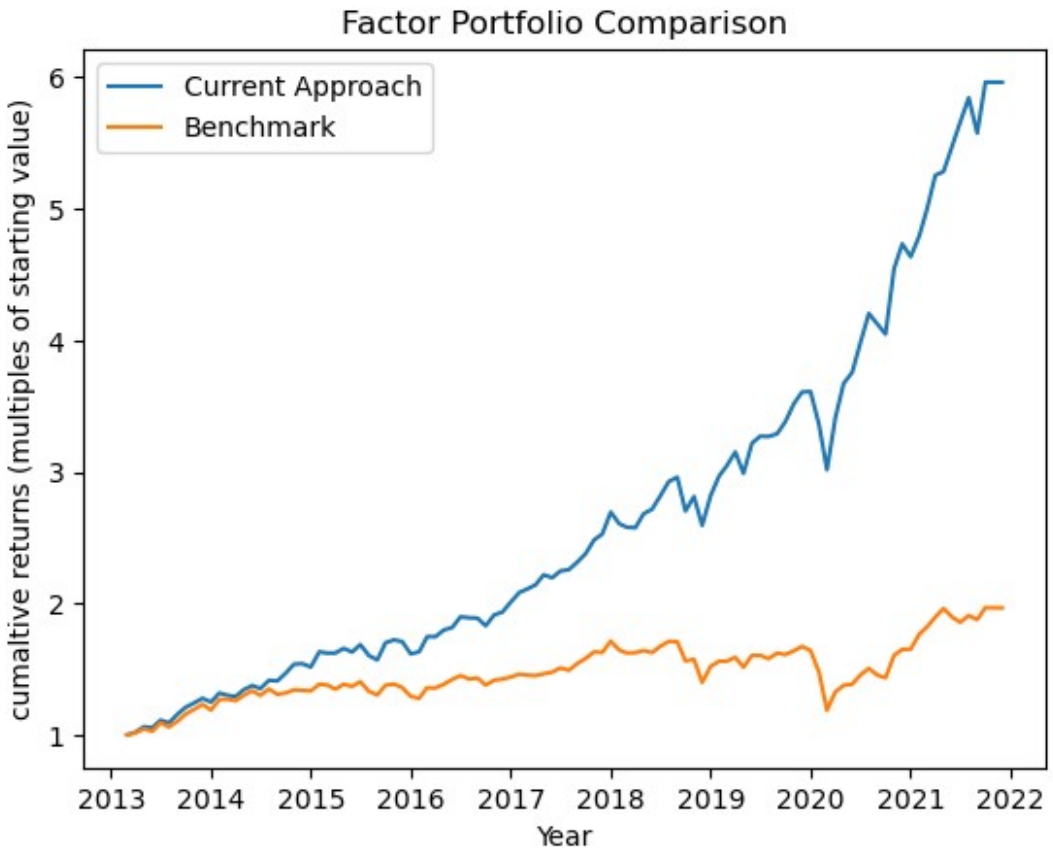



```
c:\Users\ChunMing\OneDrive - EL Accounting\Documents\VS Code\MSFE HFS-
Project N\code_base.py:90: RuntimeWarning: invalid value encountered
in double_scalars
  ratio = error.mean(skipna=True)/(error.std(skipna=True))
```

	Current Approach	Benchmark
Annualised Information Ratio	1.838	nan
Annualised Sharpe Ratio (rf = 3%)	0.919	-0.950
Max Drawdown	0.262	0.617
Annualised Volatility	0.123	0.117
Annualised Average Return	0.143	-0.081

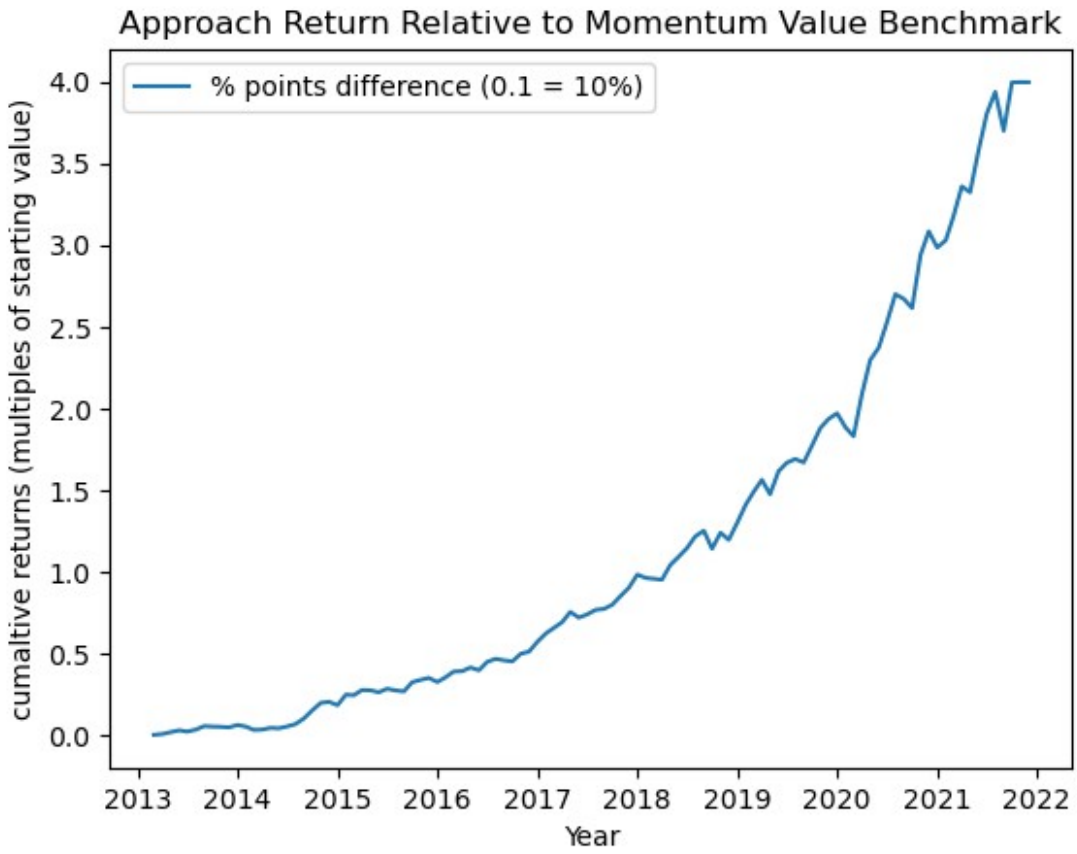



```
# Long Only
test_port_inv.quick_plt_diff(test_port_momval,type="lw")
```




```
c:\Users\ChunMing\OneDrive - EL Accounting\Documents\VS Code\MSFE HFS-
Project N\code_base.py:90: RuntimeWarning: invalid value encountered
in double_scalars
  ratio = error.mean(skipna=True)/(error.std(skipna=True))
```

	Current Approach	Benchmark
Annualised Information Ratio	1.962	nan
Annualised Sharpe Ratio (rf = 3%)	1.344	0.398
Max Drawdown	0.165	0.307
Annualised Volatility	0.136	0.144
Annualised Average Return	0.213	0.087

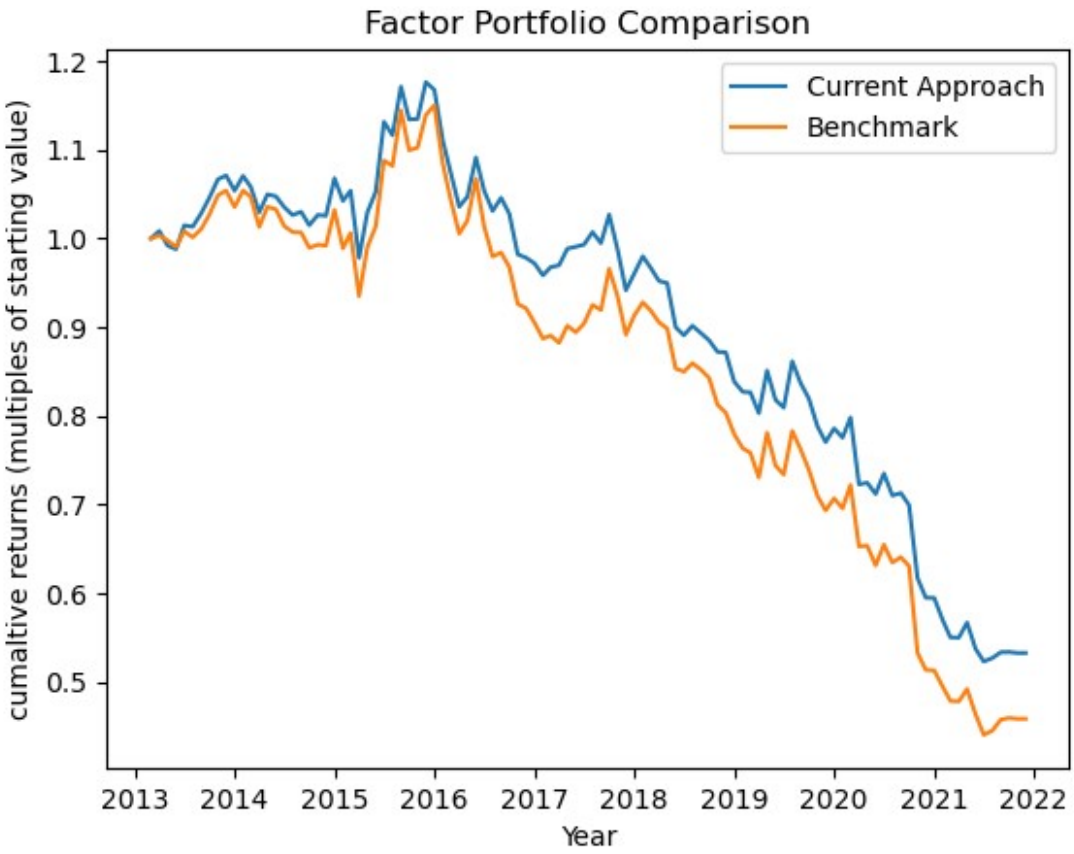


Here, once again, we see that the inverse momentum portfolio has been one of the most dominant factors for the S&P 5000

Earnings Information: Composite Score

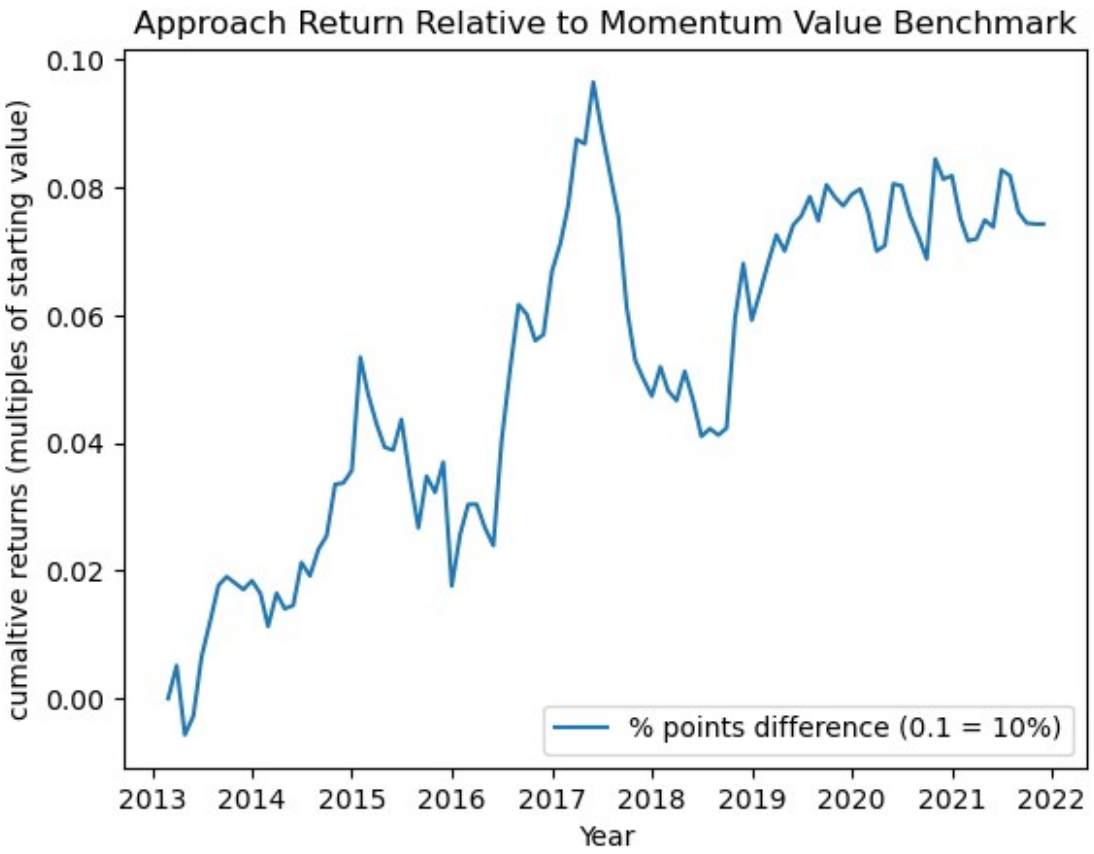
Here, we see that the introduction of earnings surprise information improves the momentum value benchmark of the sample. The Information ratio has actually increased from the in-sample value of 0.352, and we do see a small increase in alpha from by ~0.6-1% annualised

```
test_port_com = data.to_port("COM",tmbool=tmbool_test)
test_port_com.gen_weights_from_score(0.3)
test_port_com.quick_plt_diff(test_port_momval)
```

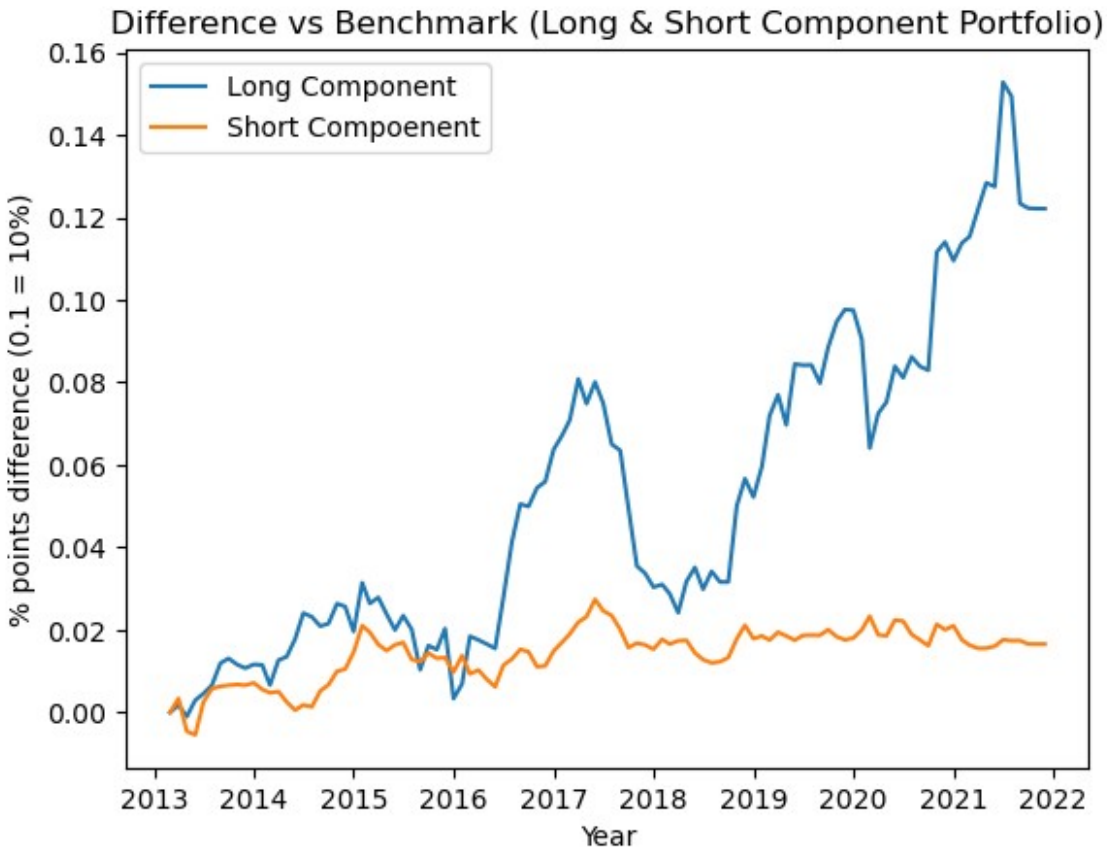


	Current Approach	Benchmark
Annualised Information Ratio	0.554	nan
Annualised Sharpe Ratio (rf = 3%)	-0.920	-0.950
Max Drawdown	0.555	0.617
Annualised Volatility	0.104	0.117
Annualised Average Return	-0.066	-0.081

```
c:\Users\ChunMing\OneDrive - EL Accounting\Documents\VS Code\MSFE HFS-
Project N\code_base.py:90: RuntimeWarning: invalid value encountered
in double_scalars
  ratio = error.mean(skipna=True)/(error.std(skipna=True))
```

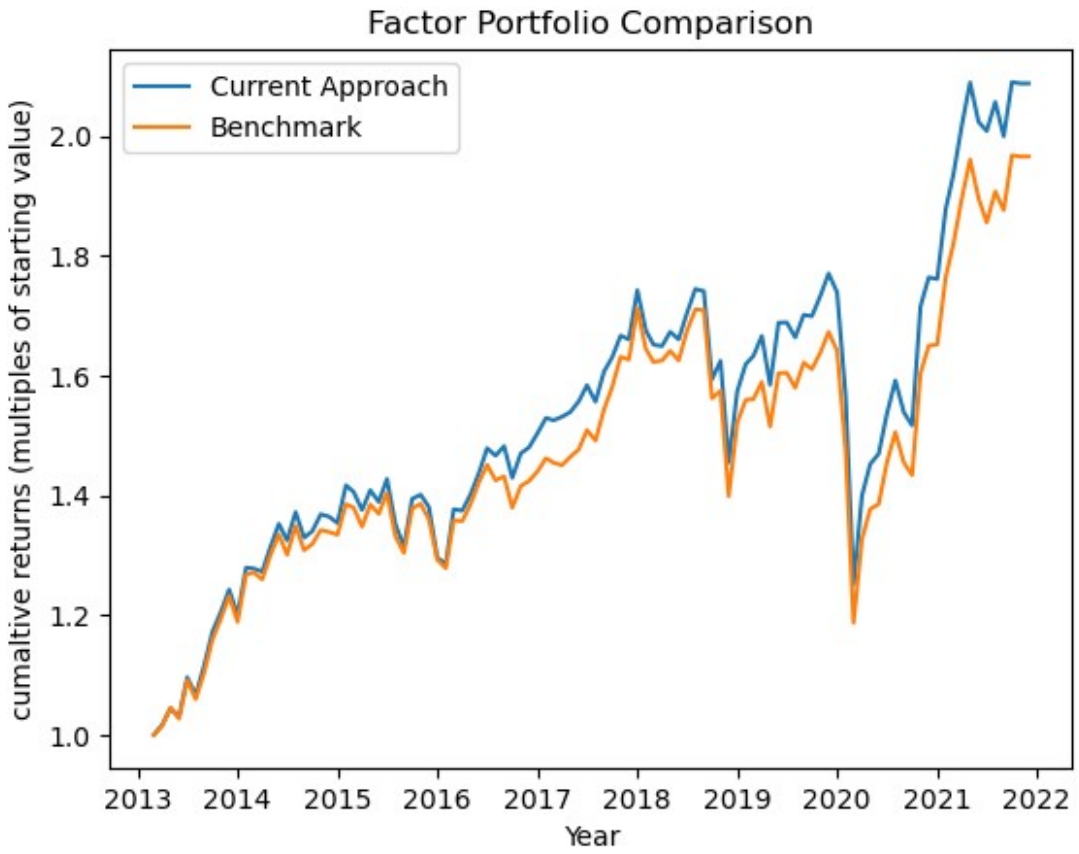



```
test_port_com.quick_plt_ls(test_port_momval)
```



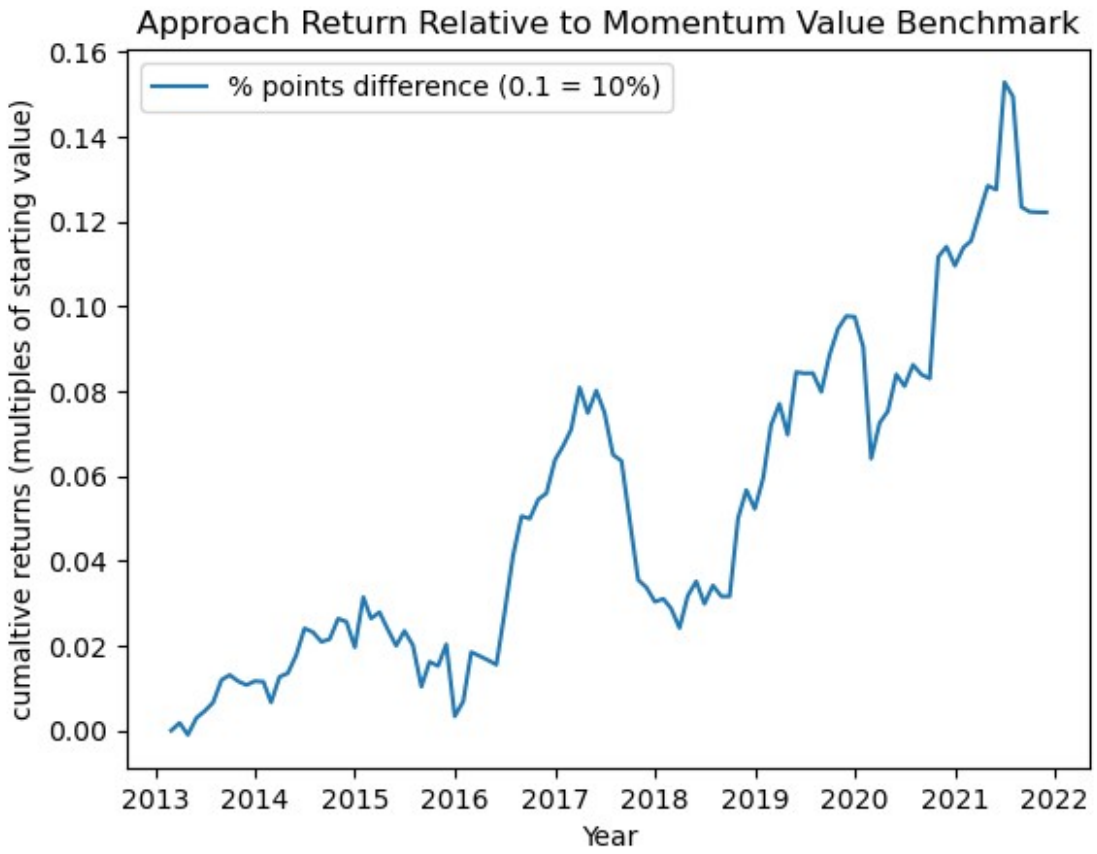
Here, the long portfolio also impoved over momentum value, however it remained insuffienct to beat the market weighted index, and underperformed the momentum only portfolio

```
# Long only
test_port_com.quick_plt_diff(test_port_momval,type="lw")
```




```
c:\Users\ChunMing\OneDrive - EL Accounting\Documents\VS Code\MSFE HFS-
Project N\code_base.py:90: RuntimeWarning: invalid value encountered
in double_scalars
  ratio = error.mean(skipna=True)/(error.std(skipna=True))
```

	Current Approach	Benchmark
Annualised Information Ratio	0.453	nan
Annualised Sharpe Ratio (rf = 3%)	0.442	0.398
Max Drawdown	0.293	0.307
Annualised Volatility	0.146	0.144
Annualised Average Return	0.095	0.087



We see that the returns generated largely from its long portfolio, which is consistent with in-sample observations.

Regression Analyis

we ran a regression the following regression betwen, and observe whether if there is a change $\hat{\alpha}$ due to our stratergy

$$R_t^{[\text{MOM + VAL}]} = \alpha + \beta_{\text{MOM}} R_t^{\text{MOM}} + \beta_{\text{VAL}} R_t^{\text{VAL}} + \varepsilon_t$$

$$R_t^{[\text{Approach}]} = \hat{\alpha} + \hat{\beta}_{\text{MOM}} R_t^{\text{MOM}} + \hat{\beta}_{\text{VAL}} R_t^{\text{VAL}} + \varepsilon_t$$

we saw that here $\hat \alpha $ has increased from -0.0009% to -0.0004% monthly, furhtermore alpha has also became statistically insignifiant, suggesting that the earnings suprise factor do provide some information

```
# alpha regression
total_port_com = data.to_port("COM",tmbool=tmbool)
total_port_com.gen_weights_from_score(0.3)
rets,_ = total_port_com.get_port_ret(bps=10)

total_port_mom_val = data.to_port("MOM_VAL",tmbool=tmbool)
total_port_mom_val.gen_weights_from_score(0.3)
bench_rets,_ = total_port_mom_val.get_port_ret(bps=10)
```

```python
X = factor_ret.copy()
X["Y"] = bench_rets
# factor_ret.
X.dropna(inplace=True)
Y = X.pop("Y")

X = X.astype(float)
ols_mod = sm.OLS(Y,X)
res = ols_mod.fit()
print(res.summary())

X = factor_ret.copy()
X["Y"] = rets
# factor_ret.
X.dropna(inplace=True)
Y = X.pop("Y")

X = X.astype(float)
ols_mod = sm.OLS(Y,X)
res2 = ols_mod.fit()
```

```
                            OLS Regression Results

==============================================================================
========
Dep. Variable:                          Y   R-squared:
0.981
Model:                                OLS   Adj. R-squared:
0.981
Method:                     Least Squares   F-statistic:
6516.
Date:                   Tue, 13 May 2025   Prob (F-statistic):
9.57e-216
Time:                           10:13:08   Log-Likelihood:
894.22
No. Observations:                    252   AIC:
-1782.
Df Residuals:                        249   BIC:
-1772.
Df Model:                              2

Covariance Type:             nonrobust

==============================================================================
========
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
--------
```

```
MOM             0.9372      0.008    111.497      0.000       0.921
0.954
VAL             0.3494      0.016     22.431      0.000       0.319
0.380
Intercept      -0.0012      0.000     -2.517      0.012      -0.002
-0.000
==============================================================================
========
Omnibus:                       76.210   Durbin-Watson:
1.682
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
338.760
Skew:                          -1.151   Prob(JB):
2.75e-74
Kurtosis:                       8.192   Cond. No.
36.4
==============================================================================
========

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
```

```
# Factor Regression with Earnings Factor
print(res2.summary())
```

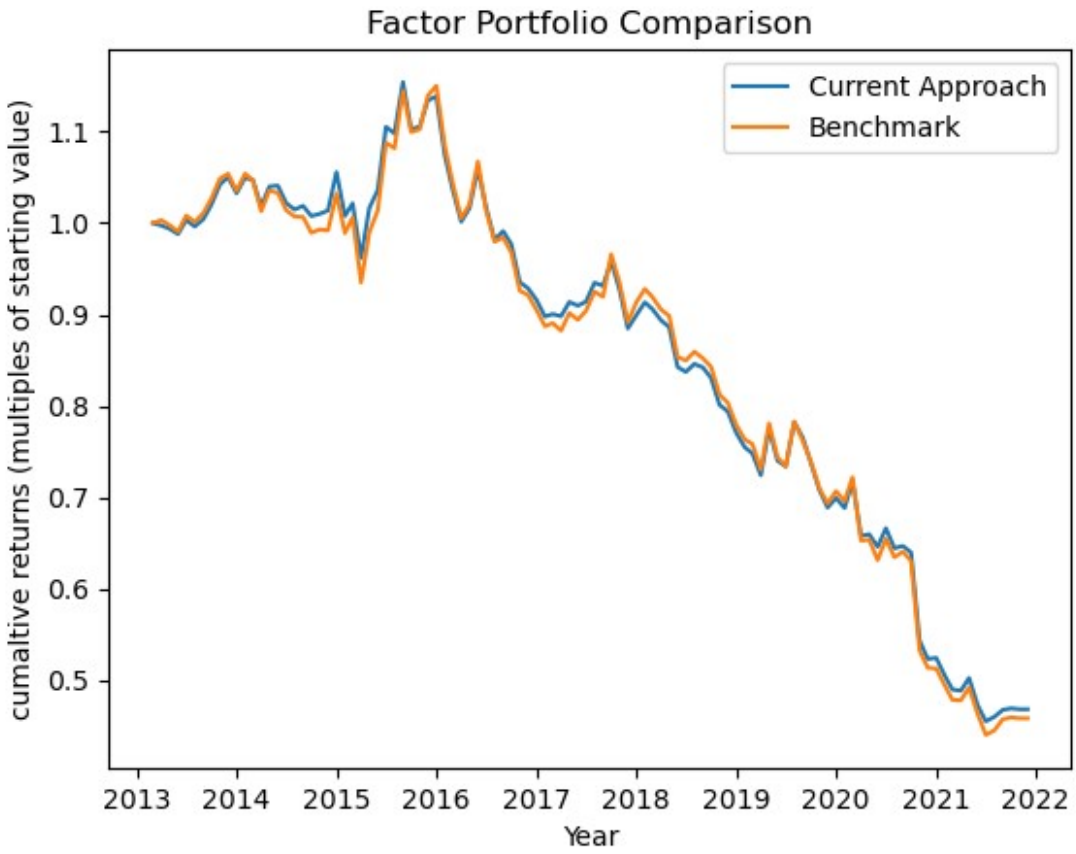
```
                            OLS Regression Results

==============================================================================
========
Dep. Variable:                      Y   R-squared:
0.943
Model:                            OLS   Adj. R-squared:
0.943
Method:                 Least Squares   F-statistic:
2077.
Date:                Tue, 13 May 2025   Prob (F-statistic):
4.85e-156
Time:                        10:13:08   Log-Likelihood:
786.25
No. Observations:                 252   AIC:
-1567.
Df Residuals:                     249   BIC:
-1556.
Df Model:                           2

Covariance Type:            nonrobust

==============================================================================
========
```

```
                  coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
--------
MOM            0.8086      0.013     62.675      0.000       0.783
0.834
VAL            0.2747      0.024     11.487      0.000       0.228
0.322
Intercept     -0.0004      0.001     -0.637      0.525      -0.002
0.001
==============================================================================
========
Omnibus:                       76.328   Durbin-Watson:
1.763
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
589.091
Skew:                           0.956   Prob(JB):
1.20e-128
Kurtosis:                      10.242   Cond. No.
36.4
==============================================================================
========

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
```

Earnings Information: Conditional Weights

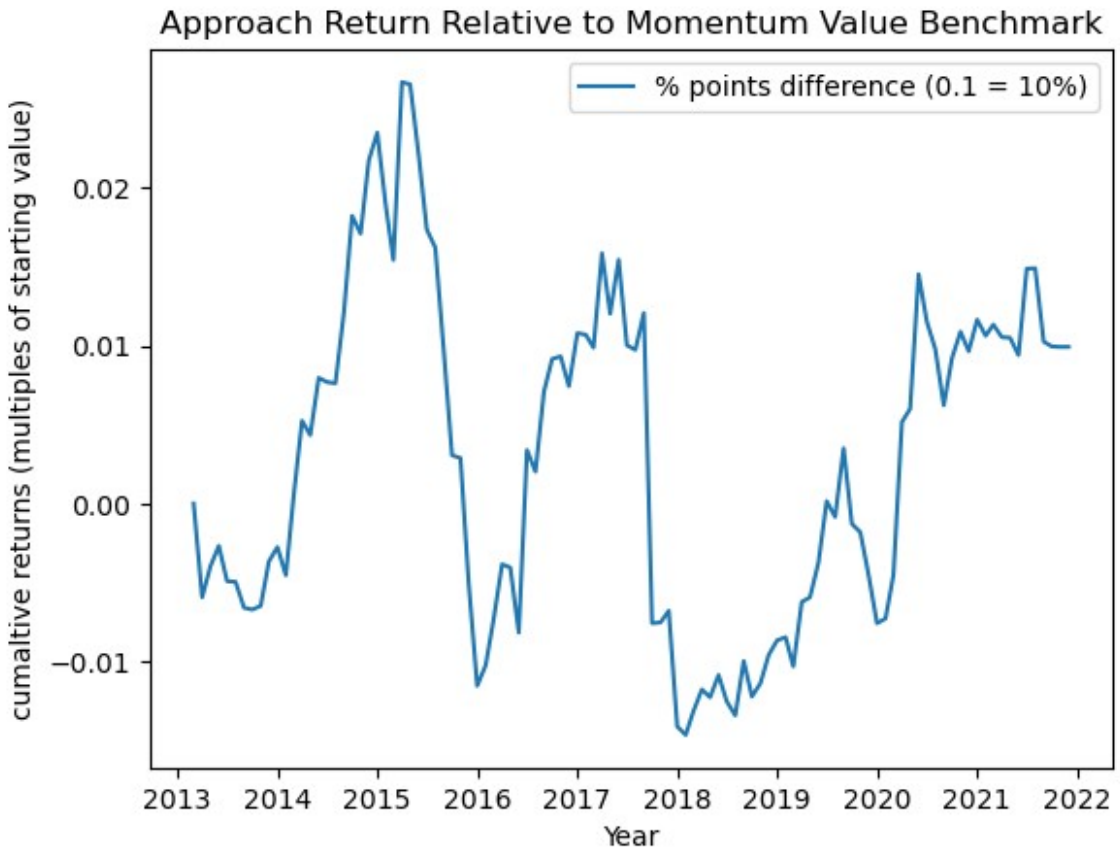
There have been little to no additional returns observed from this approach, largely similar to before. The use of its short component return as a potential indicator out of sample has been inconclusive, as the signal showcased different characteristics.

```
test_port_ejc = data.to_port("EJC",tmbool=tmbool_test)
test_port_ejc.gen_weights_from_score(0.3)
test_port_ejc.quick_plt_diff(test_port_momval)
```

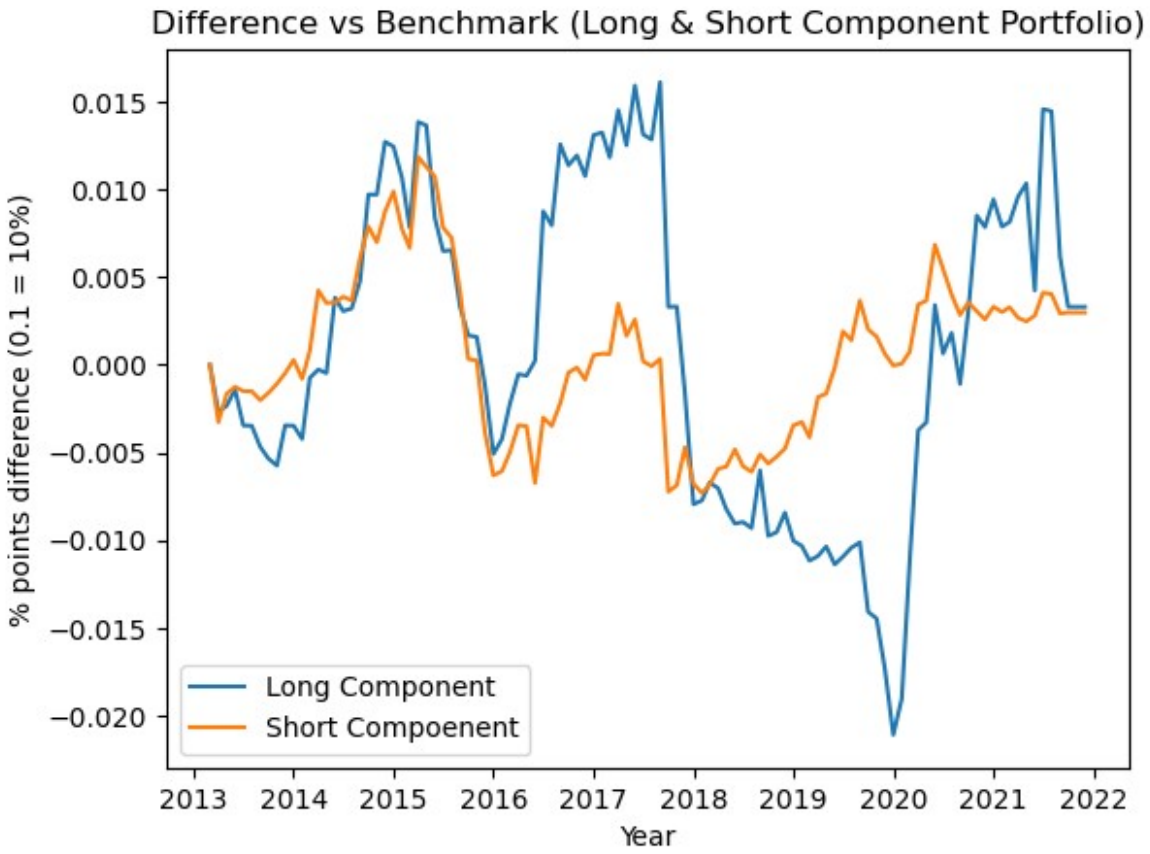



```
                                  Current Approach Benchmark
Annualised Information Ratio                 0.092       nan
Annualised Sharpe Ratio (rf = 3%)           -1.001    -0.950
Max Drawdown                                 0.606     0.617
Annualised Volatility                        0.109     0.117
Annualised Average Return                   -0.080    -0.081
```

```
c:\Users\ChunMing\OneDrive - EL Accounting\Documents\VS Code\MSFE HFS-
Project N\code_base.py:90: RuntimeWarning: invalid value encountered
in double_scalars
  ratio = error.mean(skipna=True)/(error.std(skipna=True))
```



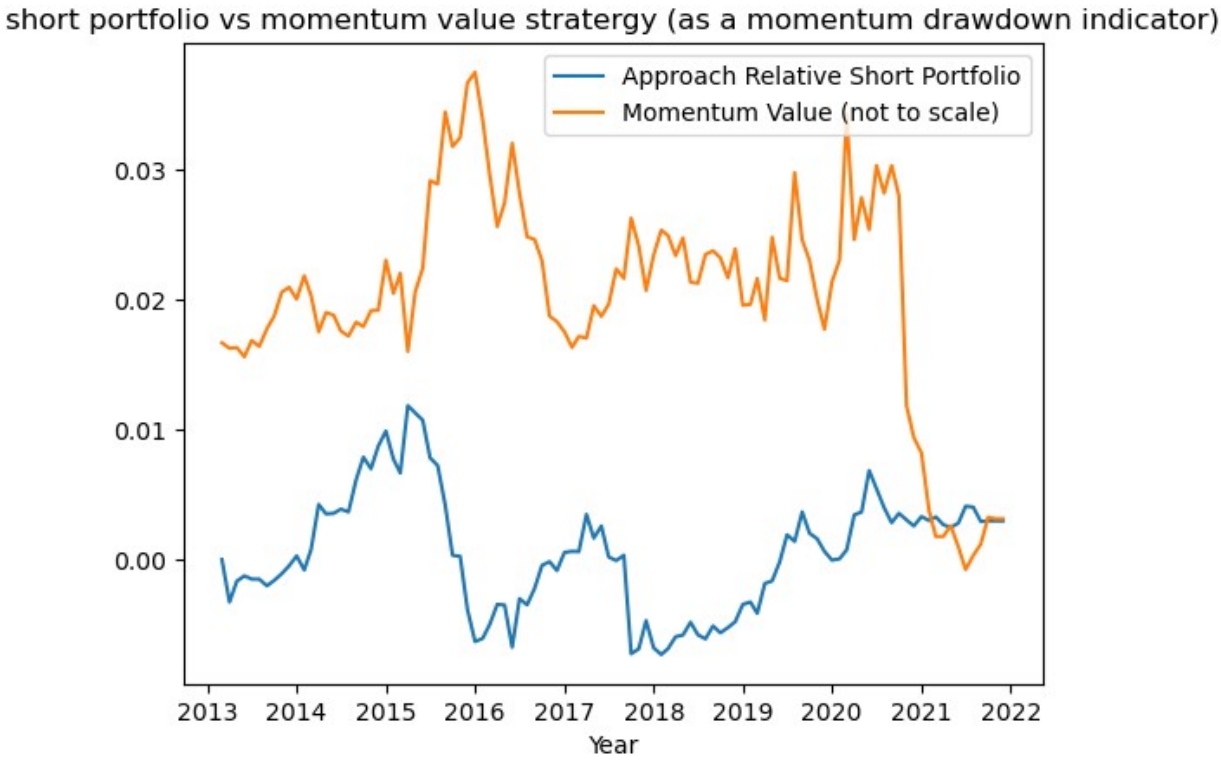

```
test_port_ejc.quick_plt_ls(test_port_momval)
```



Again, long component has failed to capture additional information. Instead, we are most interested in its short components, as a potential indicator for momentum drawdowns.

(Figure below) Whilst there is still an early indication from the signal, we saw the signal drawdown leading both the momentum returns and also the momentum drawdown significantly, potentially degrading its usefulness

```
_, tcret = test_port_ejc.get_port_ret(test_port_ejc.sw,bps=10)
_, bench_tcret =
test_port_momval.get_port_ret(test_port_momval.sw,bps=10)
_, mom_tcret= test_port_momval.get_port_ret( test_port_mom.lsw,
bps=10)

plt.plot(tcret-bench_tcret)
plt.plot(mom_tcret/15-0.05)

plt.title("short portfolio vs momentum value stratergy (as a momentum
drawdown indicator)")
plt.legend(["Approach Relative Short Portfolio", "Momentum Value (not
to scale)"])
plt.xlabel("Year")

Text(0.5, 0, 'Year')
```




```
# alpha regression
total_port_com = data.to_port("EJC",tmbool=tmbool)
total_port_com.gen_weights_from_score(0.3)
rets,_ = total_port_com.get_port_ret(bps=10)

X = factor_ret.copy()
X["Y"] = rets
# factor_ret.
X.dropna(inplace=True)
Y = X.pop("Y")

X = X.astype(float)
ols_mod = sm.OLS(Y,X)
res2 = ols_mod.fit()
print(res2.summary())
```

```
                            OLS Regression Results

==============================================================================
========
Dep. Variable:                          Y   R-squared:
0.970
Model:                                OLS   Adj. R-squared:
0.970
Method:                     Least Squares   F-statistic:
4004.
```

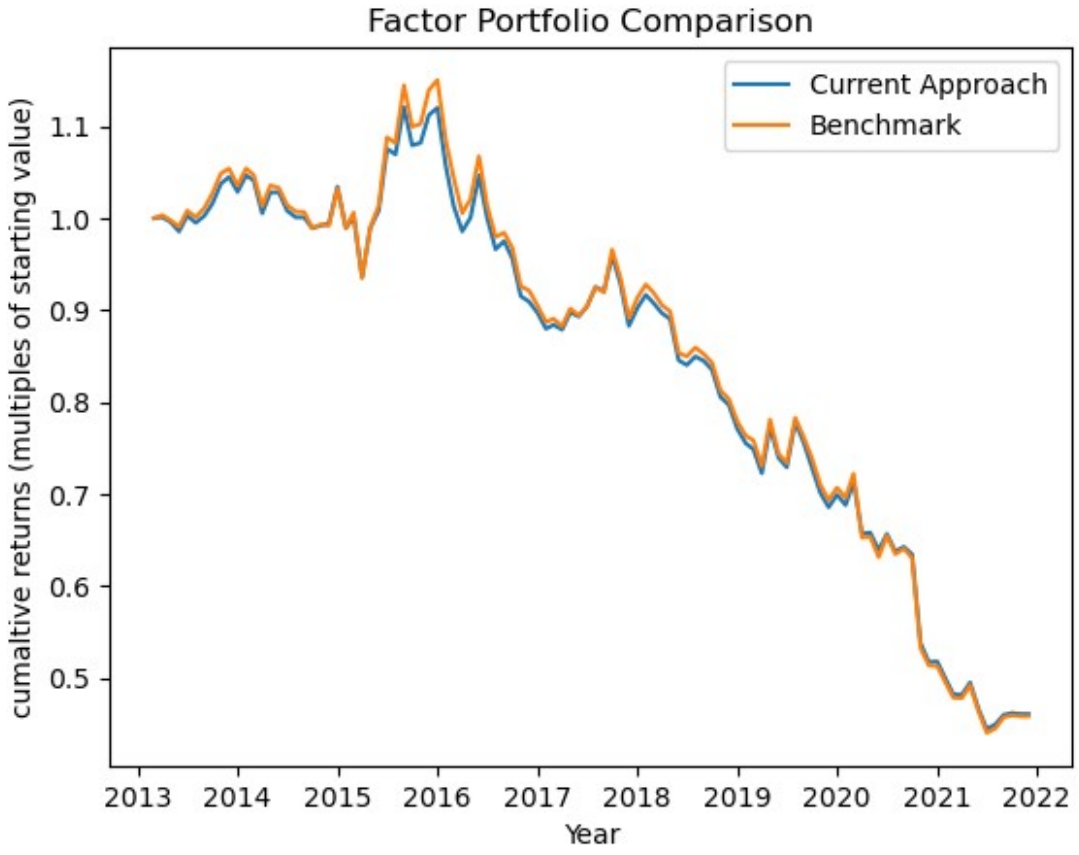
```
Date:                Tue, 13 May 2025   Prob (F-statistic):
4.76e-190
Time:                        10:13:12   Log-Likelihood:
845.70
No. Observations:                 252   AIC:
-1685.
Df Residuals:                     249   BIC:
-1675.
Df Model:                           2

Covariance Type:            nonrobust

==============================================================================
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
MOM            0.8898      0.010     87.311      0.000       0.870
0.910
VAL            0.3243      0.019     17.170      0.000       0.287
0.361
Intercept     -0.0013      0.001     -2.307      0.022      -0.002
-0.000
==============================================================================
Omnibus:                       35.886   Durbin-Watson:
2.025
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
264.682
Skew:                           0.078   Prob(JB):
3.35e-58
Kurtosis:                       8.018   Cond. No.
36.4
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
```

did not produce much alpha improvement, instead it lowered the p value of alpha estimate

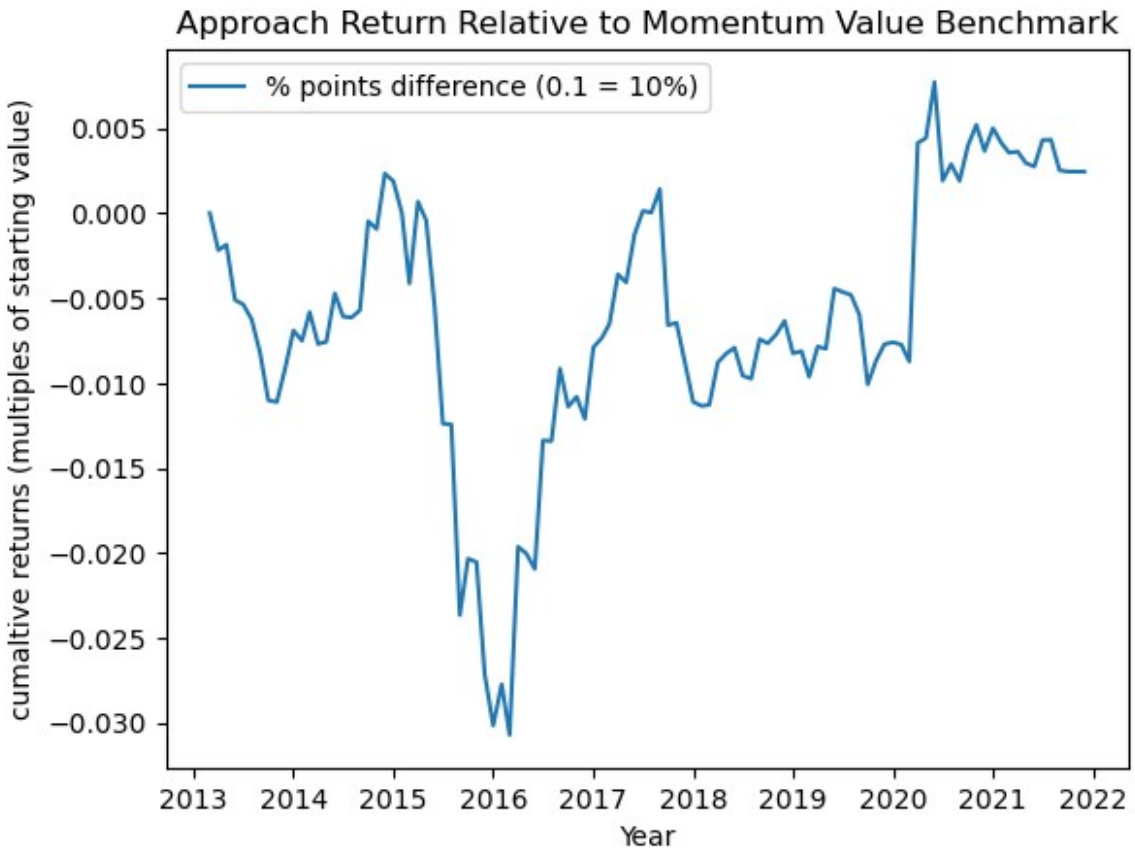
Earnings Information: Linear Regression Enabled

Here, the observation and conclusion are the same as before, where the use of linear regression to select the switch case weights has been ineffective.

```
test_port_ejc4 = data.to_port("EJC4",tmbool=tmbool_test)
test_port_ejc4.gen_weights_from_score(0.3)
test_port_ejc4.quick_plt_diff(test_port_momval)
```




```
                                  Current Approach Benchmark
Annualised Information Ratio                -0.002       nan
Annualised Sharpe Ratio (rf = 3%)           -0.995    -0.950
Max Drawdown                                 0.603     0.617
Annualised Volatility                        0.112     0.117
Annualised Average Return                   -0.081    -0.081

c:\Users\ChunMing\OneDrive - EL Accounting\Documents\VS Code\MSFE HFS-
Project N\code_base.py:90: RuntimeWarning: invalid value encountered
in double_scalars
  ratio = error.mean(skipna=True)/(error.std(skipna=True))
```




```
# alpha regression
total_port_com = data.to_port("EJC4",tmbool=tmbool)
total_port_com.gen_weights_from_score(0.3)
rets,_ = total_port_com.get_port_ret(bps=10)


X = factor_ret.copy()
X["Y"] = rets
# factor_ret.
X.dropna(inplace=True)
Y = X.pop("Y")

X = X.astype(float)
ols_mod = sm.OLS(Y,X)
res2 = ols_mod.fit()
print(res2.summary())
```

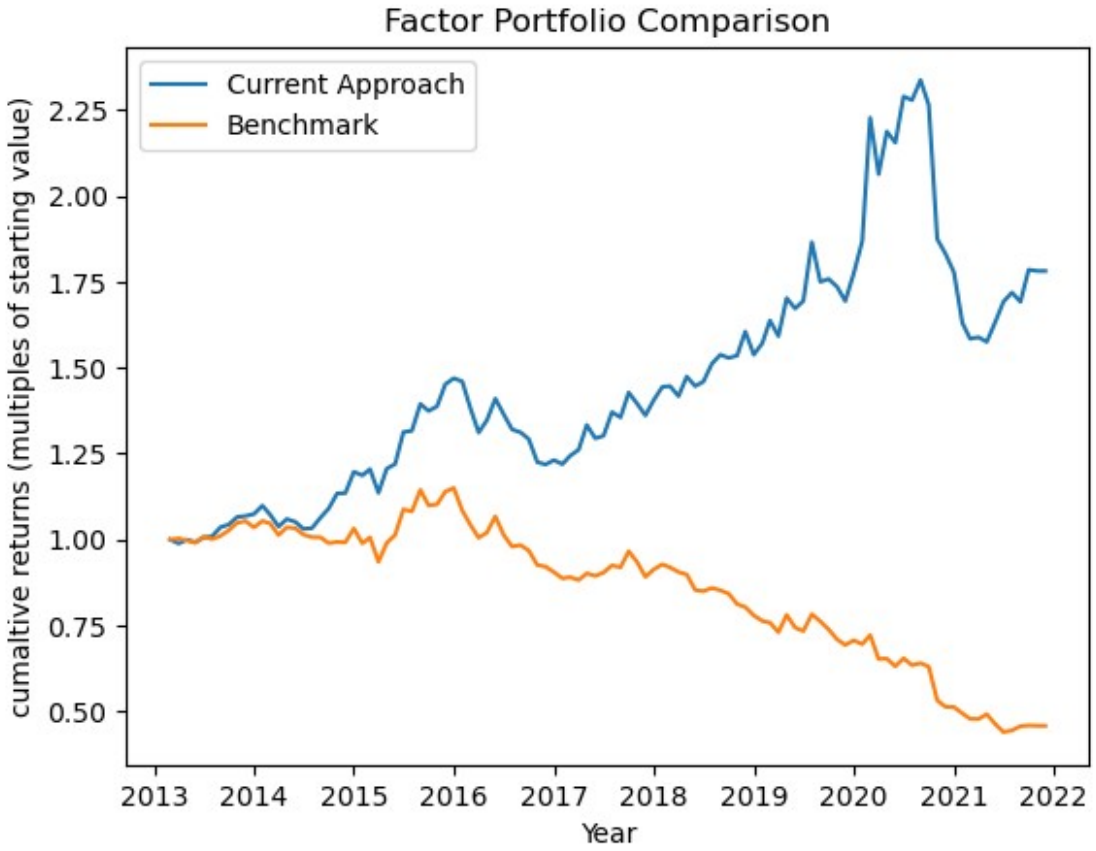
```
                            OLS Regression Results

==============================================================================
========
Dep. Variable:                          Y   R-squared:
0.980
```

```
Model:                            OLS   Adj. R-squared:
0.980
Method:                 Least Squares   F-statistic:
6057.
Date:                Tue, 13 May 2025   Prob (F-statistic):
7.14e-212
Time:                        10:13:15   Log-Likelihood:
897.11
No. Observations:                 252   AIC:
-1788.
Df Residuals:                     249   BIC:
-1778.
Df Model:                           2

Covariance Type:            nonrobust

==============================================================================
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
MOM            0.8951      0.008    107.706      0.000       0.879
0.911
VAL            0.3480      0.015     22.595      0.000       0.318
0.378
Intercept     -0.0010      0.000     -2.152      0.032      -0.002   -
8.23e-05
==============================================================================
Omnibus:                       33.206   Durbin-Watson:
1.971
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
123.085
Skew:                          -0.447   Prob(JB):
1.87e-27
Kurtosis:                       6.305   Cond. No.
36.4
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
```

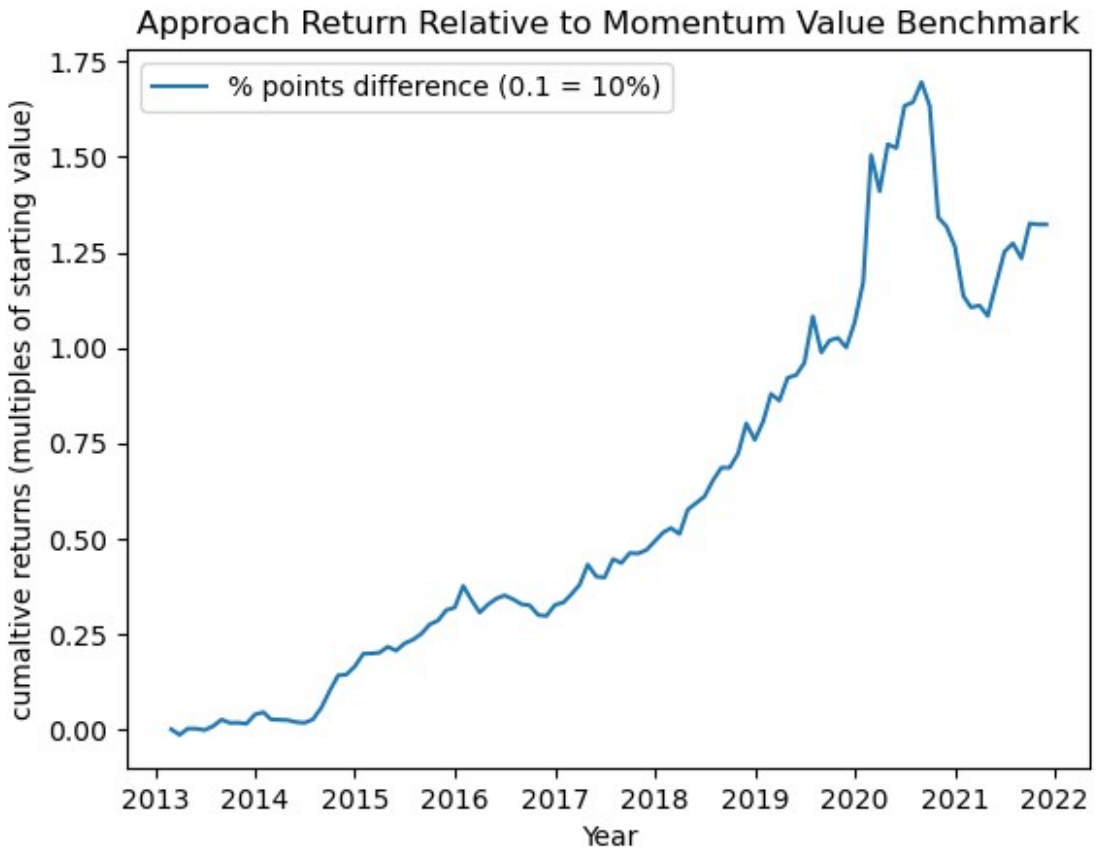
some alpha is observed, however it is both small and hard to qualitatively interpret where it comes from, whithout much observable stratergy outperformance patterns making it impractical in reality

Lastly we would like to point out that under its central case (no earnings shock) the linear regerssion was able to select good factor weights, especially for Long only portfolios

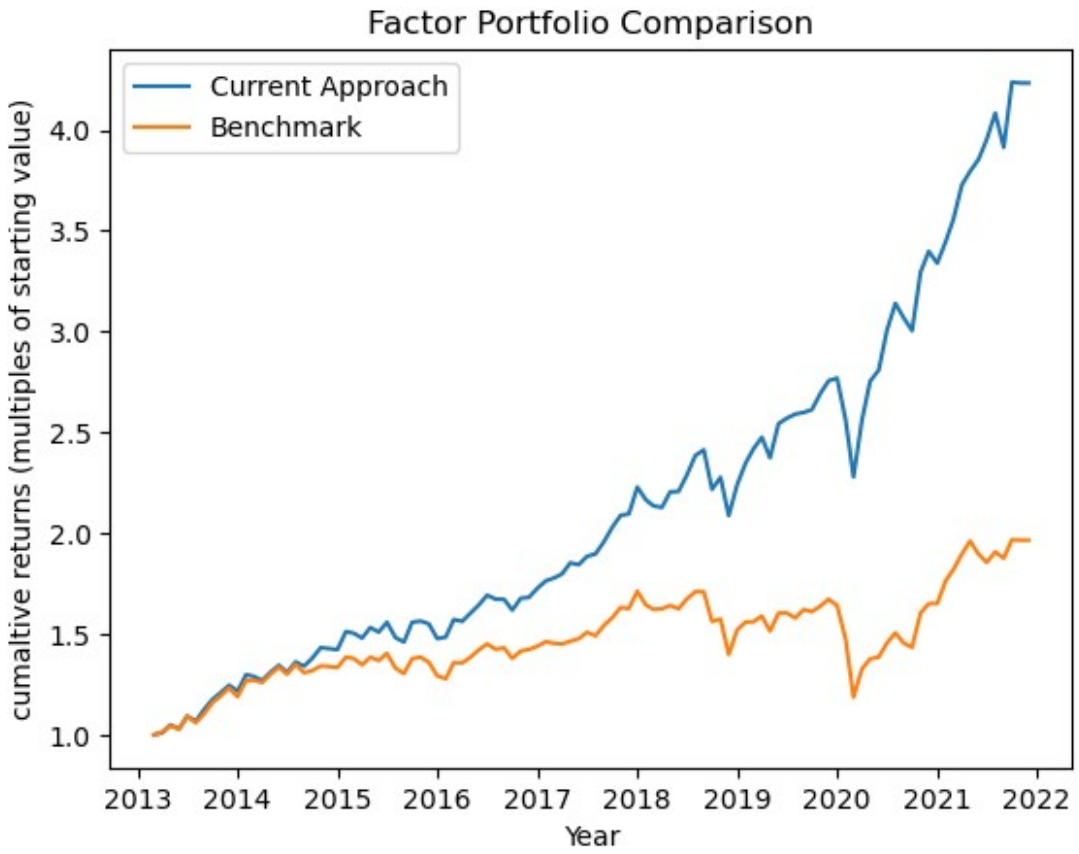
```
test_port_ejc4b = data.to_port("EJC4-B",tmbool=tmbool_test)
test_port_ejc4b.gen_weights_from_score(0.3)
test_port_ejc4b.quick_plt_diff(test_port_momval)
```




```
                                  Current Approach Benchmark
Annualised Information Ratio                 1.638       nan
Annualised Sharpe Ratio (rf = 3%)            0.316    -0.950
Max Drawdown                                 0.326     0.617
Annualised Volatility                        0.146     0.117
Annualised Average Return                    0.076    -0.081

c:\Users\ChunMing\OneDrive - EL Accounting\Documents\VS Code\MSFE HFS-
Project N\code_base.py:90: RuntimeWarning: invalid value encountered
in double_scalars
  ratio = error.mean(skipna=True)/(error.std(skipna=True))
```

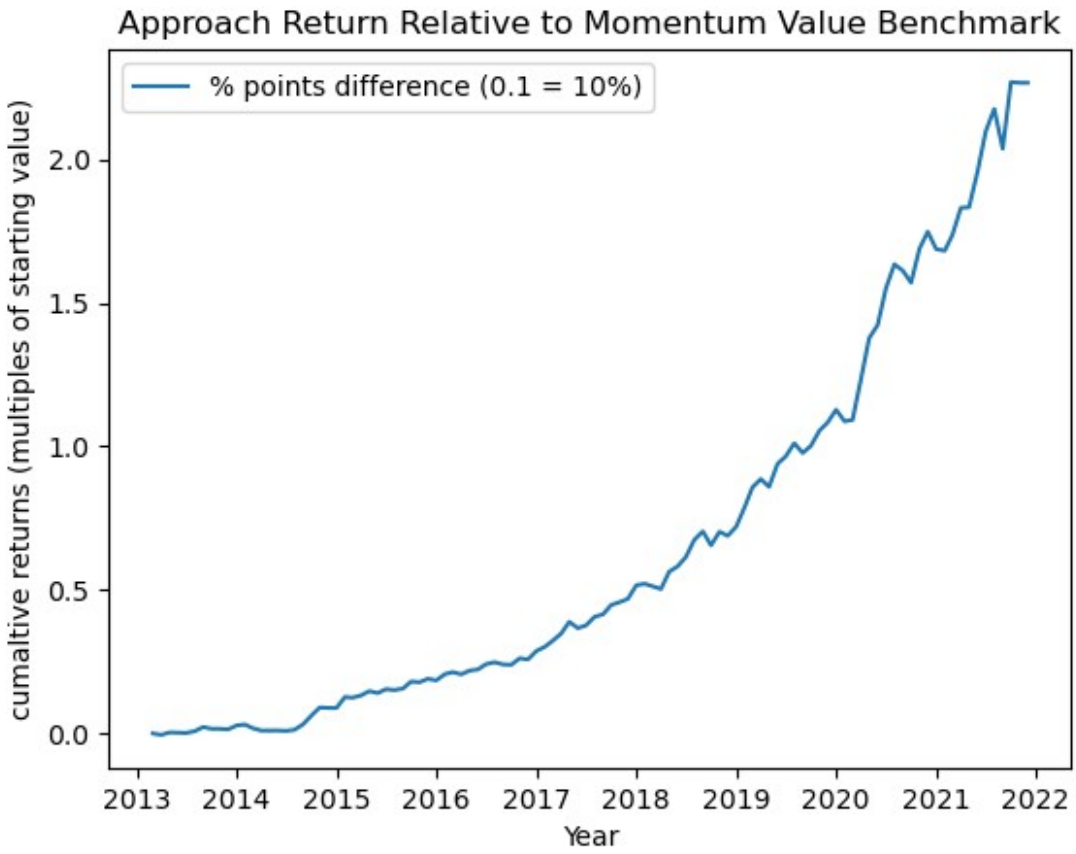



```
# Long Only, Regression Enabled - No Suprise Regime (all Suprise cases have been discarded)

test_port_ejc4b.quick_plt_diff(test_port_momval,10,"lw")
```



	Current Approach	Benchmark
Annualised Information Ratio	1.580	nan
Annualised Sharpe Ratio (rf = 3%)	1.098	0.398
Max Drawdown	0.177	0.307
Annualised Volatility	0.130	0.144
Annualised Average Return	0.173	0.087

```
c:\Users\ChunMing\OneDrive - EL Accounting\Documents\VS Code\MSFE HFS-Project N\code_base.py:90: RuntimeWarning: invalid value encountered in double_scalars
  ratio = error.mean(skipna=True)/(error.std(skipna=True))
```



4 Conclusions

we observed that the true long short momentum value portfolio has **severely underperformed** both the underlying universe and the momentum portfolio in the backtest period between 2001 and 2021. This was due to a very dominant "inverse value" effect, where the most expensive stocks consistently delivered high risk-adjusted returns, and a very consistent underperformance of any factor short portoflios. If we consider Long portfolios only, we did observed that factor portflios do out perform in sample, have all underperformed the inedx out of sample, since 2013.

In terms of a tradable stratergy, inverse momentum has been by far the most dominate in our backtest sample (both long only, and long short) this is contrary to what risk based conventional theory would suggest; and is unclear if it will continue for the future.

We then attempted to investigate whether a momentum value portfolio can improve by incorporating earnings information, using the following three approaches.

- **Composite Score**: calculate a composite score from the weighted sum of factor scores
- **Conditional Weights**: introducing conditional shock scenarios, and applying heuristic adjustments
- **Linear Regression Enabled**: use of linear regression to estimate factor weights under different surprise scenarios

We found that the **composite Score method** has been the most effective both in and out of sample cases. Whilst the more complex latter two cases have been ineffective. we have found that incorporating earning announcement information do provide additonal information in a combined momentum value stratergies; Although, the overall effects have been small (information ratio of only ~0.5 and relative "alpha" of 0.6% annualised (long short) compared to our baseline), with return profiles very similar to the underlying momentum value stratergy.

However due to the unique inverse value dominated nature of the S&P 500, further work would be required to explore whether our findings would translate to other data sets where momentum value premiums are positive, and whether other earning factors can drive improvement.

lastly we would also like highlight the use of simple linear regression methods to select factor weights, and the potential use of our negative shock heuristic conditional weight relative performance as a momentum drawdown indicator as other potential research directions.

Bibiography

1. Beaver, W.H. (1968) The Information Content of Annual Earnings Announcements. Journal of Accounting Research, 6, 67-92.https://doi.org/10.2307/2490070
2. Ball, R., & Brown, P. (1968). An Empirical Evaluation of Accounting Income Numbers. Journal of Accounting Research, 6(2), 159–178. https://doi.org/10.2307/2490232
3. Brown, S. J., & Warner, J. B. (1985). Using Daily Stock Returns: The Case of Event Studies. Journal of Financial Economics, 14(1), 3–31. https://doi.org/10.1016/0304-405X(85)90042-X
4. Bernard, V. L., & Thomas, J. K. (1989). Post-Earnings-Announcement Drift: Delayed Price Response or Risk Premium? Journal of Accounting Research, 27, 1–36. https://doi.org/10.2307/2491062
5. Chan, L.K.C., Jegadeesh, N. and Lakonishok, J. (1996), Momentum Strategies. The Journal of Finance, 51: 1681-1713. https://doi.org/10.1111/j.1540-6261.1996.tb05222.x
6. Asness, C.S., Moskowitz, T.J. and Pedersen, L.H. (2013), Value and Momentum Everywhere. The Journal of Finance, 68: 929-985. https://doi.org/10.1111/jofi.12021
7. Charles M.C Lee. Earnings news and small traders: An intraday analysis. Journal of Accounting and Economics, 15(2):265–302, 1992. https://doi.org/10.1016/0165-4101(92)90021-S
8. Fama, E.F. and French, K.R. (1992), The Cross-Section of Expected Stock Returns. The Journal of Finance, 47: 427-465. https://doi.org/10.1111/j.1540-6261.1992.tb04398.x
9. Barber, B. M., De George, E.T., Lehavy, R, and Trueman, B. (2013). "The earnings announcement premium around the globe," Journal of Financial Economics, Elsevier, vol. 108(1), pages 118-138. https://doi.org/10.1016/j.jfineco.2012.10.006