Momentum-Based Strategy

FINA 6333 for Spring 2025

Richard Herron

### Momentum-Based Strategy

#### Executive Summary

##### Objective

This project explores a momentum-based long-short strategy using the monthly industry return dataset. Momentum investing operates on a simple rule that assets which have done well recently will continue to do in the near future, while poor performers will keep underperforming. Our approach systematically ranks industries into deciles based on their past returns and constructs a portfolio that aims to profit from this trend.

* Formation Period: We compute cumulative returns over the past 12 months (excluding the most recent month) to rank the chosen industries
* Ranking & Grouping: Industries are sorted into deciles. We go long on the top 3 deciles (“winners”) and short the bottom decile
* Portfolio Construction:
  + The winners are equally weighted with a total long allocation of 130%
  + The losers are equally weighted with a total short allocation of -30%
* The rebalancing was done monthly, and we tracked how the combined portfolio performed over time.

This creates a dollar-neutral strategy with more emphasis on long positions, which helps manage volatility while exploiting momentum signals.

##### Why This Strategy Is Competitive

While the full-sample Sharpe ratio of ~0.74 may appear moderate, it does not tell the full story. Figure 1 shows that this strategy steadily compounds wealth over the long run, and if you had invested 1,000 dollars at the beginning of the period, it would’ve grown into over $110 million by the end. Below are the reasons why this strategy remains a strong competitor in both absolute and risk-adjusted terms:

* Fully Invested, Always Active: Unlike breakout strategies that may hold cash when no signal is present, our momentum strategy is always allocated — either long or short. This ensures consistent exposure to return-generating opportunities.
* Balanced Weights: By allocating 130% to longs and 30% to shorts, we capitalize more on winners while limiting risk from the shorts.
* Diversified Returns: Since returns are diversified across many different industries, the strategy benefits from the diversification effect, meaning that it reduces the risk that one bad pick will ruin the portfolio, creating more stable momentum signals.
* Cumulative: Even though the loser portfolio drags performance during crises (e.g., dot-com, 2008 financial crisis), the compounding from long positions more than compensates, creating strong cumulative growth.

##### Performance Summary

* Total Return: As shown in Figure 1, a 1000 dollar investment in this strategy grows to over $110 million. This explosive compounding is what makes the strategy shine from a total-return perspective.
* CAGR: At 18.43%, the compounded growth rate is strong, especially given the long historical sample from 1929 to 2023.
* Max Drawdown: The worst observed drawdown of -73.4% reflects real-world periods of market stress (e.g., 2008). While high, it’s consistent with trend-following strategies that remain fully invested.
* Sortino Ratio: Our Sortino ratio of 1.02 outperforms our Sharpe ratio, showing that downside risk was better managed than total volatility would suggest. This is a strong indicator that returns came with relatively fewer severe negative months.
* Risk-Adjusted Return: As seen in Figure 3, our 5-year rolling Sharpe ratio peaked at 2.05 in April 1946, suggesting that if the sampling period is more favorable, our strategy could rank near the top in risk-adjusted returns too.

##### Conclusion

Our strategy blends robust long-term performance with disciplined risk exposure. It doesn’t just chase returns — it systematically applies momentum filters, rebalances consistently, and uses leverage smartly. While the Sharpe ratio may seem average over the entire period, our total return and CAGR metrics suggest otherwise.

More importantly, the strategy’s performance improves significantly when viewed through shorter, more favorable windows, where risk-adjusted metrics like Sharpe and Sortino ratios reflect efficiency. This makes our strategy a well-rounded contender that delivers across multiple evaluation criteria.

import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
import matplotlib.ticker as ticker  
import seaborn as sns  
%precision 4  
pd.options.display.float\_format = '{:.4f}'.format  
%config InlineBackend.figure\_format = 'retina'

ind49 = (  
 pd.read\_csv(  
 filepath\_or\_buffer='data\_ind49.csv',  
 parse\_dates=['Date'],  
 na\_values=[-99.99, -999]  
 )  
 .assign(Date=lambda x: x['Date'].dt.to\_period('M'))  
 .set\_index('Date')  
 .sort\_index()  
 .rename\_axis(columns=['Industry'])  
 .div(100)  
)  
ff3 = (  
 pd.read\_csv(  
 filepath\_or\_buffer='data\_ff3.csv',  
 parse\_dates=['Date'],  
 na\_values=[-99.99, -999]  
 )  
 .assign(Date=lambda x: x['Date'].dt.to\_period('M'))  
 .set\_index('Date')  
 .rename\_axis(columns=['Factor'])  
 .div(100)  
)

mom = (  
 pd.read\_csv(  
 filepath\_or\_buffer='data\_mom.csv',  
 parse\_dates=['Date'],  
 na\_values=[-99.99, -999]  
 )  
 .assign(Date=lambda x: x['Date'].dt.to\_period('M'))  
 .set\_index('Date')  
 .rename\_axis(columns=['Factor'])  
 .div(100)  
)  
  
ff5 = (  
 pd.read\_csv(  
 filepath\_or\_buffer='data\_ff5.csv',  
 parse\_dates=['Date'],  
 na\_values=[-99.99, -999]  
 )  
 .assign(Date=lambda x: x['Date'].dt.to\_period('M'))  
 .set\_index('Date')  
 .rename\_axis(columns=['Factor'])  
 .div(100)  
)

cleaned\_data = ind49.dropna(axis=1, how='any')  
past\_11 = cleaned\_data.add(1).rolling(12).apply(np.prod) - 1

def momentum(formation):  
 end\_measurement = (formation - 1).asfreq('M')  
 ret\_12 = past\_11.loc[end\_measurement]  
 ret\_12 = ret\_12.reset\_index()  
 ret\_12['quintile'] = pd.qcut(ret\_12.iloc[:, 1], 10, labels=False)  
 winners = ret\_12[ret\_12.quintile >= 8]  
 losers = ret\_12[ret\_12.quintile <= 1]  
 winners\_rets = cleaned\_data.loc[(formation + 1).asfreq('M'), cleaned\_data.columns.isin(winners['Industry'])]  
 losers\_rets = cleaned\_data.loc[(formation + 1).asfreq('M'),cleaned\_data.columns.isin(losers['Industry'])]  
 w\_long = 1.3 / len(winners\_rets)   
 w\_short = -0.3 / len(losers\_rets)   
 WinnersRets = (winners\_rets \* w\_long).sum()  
 LosersRets = (losers\_rets \* w\_short).sum()  
 MomentumProfit = (winners\_rets \* w\_long).sum() + (losers\_rets \* w\_short).sum()  
 return MomentumProfit, WinnersRets, LosersRets  
  
formation\_dates = cleaned\_data.index[12:-1]  
returns = []  
dates = []  
for date in cleaned\_data.index[12:-1]:  
 mom\_ret, win\_ret, los\_ret = momentum(date)  
 returns.append([mom\_ret, win\_ret, los\_ret])  
 dates.append(date)  
momentumData = pd.DataFrame(returns, columns=['Momentum Returns', 'Winners', 'Losers'], index=dates)

momentumData.add(1).cumprod().mul(1000).plot(  
 figsize=(10, 6),  
 title='Cumulative Returns (Momentum Strategy)',  
 linewidth=2  
)  
plt.yscale('log')  
plt.ylabel('Value ($)')  
plt.xlabel('Date')  
plt.legend(title='Portfolio')  
plt.gca().yaxis.set\_major\_formatter(ticker.FuncFormatter(lambda x, p: format(int(x), ',')))  
plt.tight\_layout()  
plt.show()

|  |
| --- |
| Figure 1: Cumulative Returns of the Momentum Strategy |

As shown in [Figure 1](#fig-cumulative-returns), This chart shows the value of $1,000 invested in winners, losers, and our combined momentum strategy from the early 1930s onward. The blue line represents the momentum strategy, orange line represents the winners and the green line represents the losers portfolio.

def Volatility(x, ann\_fac=np.sqrt(12)):  
 return ann\_fac \* x.std()  
def make\_sharpe(rf\_series):  
 def Sharpe(x):  
 aligned\_rf = rf\_series.reindex(x.index)  
 excess = x - aligned\_rf  
 return np.sqrt(12) \* excess.mean() / excess.std()  
 return Sharpe  
def make\_sortino(rf\_series):  
 def Sortino(x):  
 aligned\_rf = rf\_series.reindex(x.index)  
 excess = x - aligned\_rf  
 downside = excess[excess < 0].std()  
 return np.sqrt(12) \* excess.mean() / downside if downside else np.nan  
 return Sortino  
def Total\_Return(x):  
 return (1 + x).prod() - 1  
def CAGR(x, ann\_fac=12):  
 return (1 + x).prod() \*\* (ann\_fac / x.count()) - 1  
def Drawdown(x):  
 price = (1 + x).cumprod()  
 return (price / price.cummax()).min() - 1

strategy\_stats = (  
 momentumData[['Momentum Returns']]  
 .agg([Total\_Return, CAGR, Volatility,  
 make\_sharpe(ff3['RF']),  
 make\_sortino(ff3['RF']),   
 Drawdown])  
 .rename\_axis(index='Statistic')  
 .round(4)  
 .transpose()  
)  
  
strategy\_stats['Strategy'] = 'Momentum'  
strategy\_stats = strategy\_stats.reset\_index(drop=True)  
strategy\_stats

| Statistic | Total\_Return | CAGR | Volatility | Sharpe | Sortino | Drawdown | Strategy |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 14351855.5617 | 0.1843 | 0.2185 | 0.7412 | 1.0178 | -0.7304 | Momentum |

filtered\_stats = strategy\_stats.drop(columns=['Total\_Return', 'CAGR', 'Drawdown', 'Volatility'])  
long\_stats\_filtered = filtered\_stats.melt(id\_vars='Strategy', var\_name='Statistic', value\_name='Value')  
side\_stats = strategy\_stats.drop(columns=['Total\_Return', 'Sharpe', 'Sortino'])  
long\_side\_stats = side\_stats.melt(id\_vars='Strategy', var\_name='Statistic', value\_name='Value')  
  
fig, ax = plt.subplots(1, 2, figsize=(5, 5), sharey=True)  
sns.barplot(data=long\_stats\_filtered, x='Statistic', y='Value', hue='Strategy', palette='hls', ax=ax[0])  
ax[0].set\_title('Sharpe & Sortino')  
ax[0].set\_xlabel('Statistic')  
ax[0].set\_ylabel('Value (decimals)')  
ax[0].tick\_params(axis='x', rotation=45)  
  
sns.barplot(data=long\_side\_stats, x='Statistic', y='Value', hue='Strategy', ax=ax[1])  
ax[1].set\_title('Volatility & Drawdown')  
ax[1].set\_xlabel('Metrics')  
ax[1].set\_ylabel('')  
ax[1].tick\_params(axis='x', rotation=45)  
for axis in ax:  
 axis.set\_ylim(-0.8, 1.1)  
  
fig.suptitle('Performance Statistics for Momentum Strategy', fontsize=14)  
ax[0].get\_legend().remove()  
ax[1].get\_legend().remove()  
plt.tight\_layout()  
plt.subplots\_adjust(top=0.85)   
plt.show()

|  |
| --- |
| Figure 2: Performance Metrics Including Sharpe, Sortino, Volatility, CAGR, and Drawdown |

[Figure 2](#fig-performance-stats) summarizes the key risk and return metrics for our strategy. The Sortino ratio stands out as stronger than the Sharpe, highlighting lower downside volatility.

momentum\_rf\_adj = momentumData['Momentum Returns'].sub(ff3['RF'].reindex(momentumData.index))  
rolling\_sharpe = (  
 momentum\_rf\_adj  
 .rolling(window=60)  
 .apply(lambda x: np.sqrt(12) \* x.mean() / x.std())  
)  
  
rolling\_sharpe.index = rolling\_sharpe.index.to\_timestamp()  
max\_sharpe = rolling\_sharpe.max()  
max\_date = rolling\_sharpe.idxmax()  
ax = rolling\_sharpe.plot(title='Rolling 5-Year Sharpe Ratio (Momentum Strategy)', figsize=(8, 4))  
ax.axhline(y=2, color='gray', linestyle='--', alpha=0.5)  
ax.scatter(max\_date, max\_sharpe, color='red', zorder=5)  
ax.text(max\_date, max\_sharpe + 0.01, f'Max: {max\_sharpe:.2f}', color='red', ha='center')  
ax.text(max\_date, max\_sharpe - 0.15, f'@ {max\_date.strftime("%Y-%m")}', color='red', ha='center')  
ax.set\_ylabel('Sharpe Ratio')  
ax.set\_xlabel('Date')  
plt.tight\_layout()  
plt.show()

|  |
| --- |
| Figure 3: 5-Year Rolling Sharpe Ratio of the Momentum Strategy |

[Figure 3](#fig-rolling-sharpe) shows that the strategy’s Sharpe ratio peaked at 2.05 in April 1946. This demonstrates that, depending on the sample window, our strategy could have ranked much higher on risk-adjusted return.

def write\_csv(portfolio\_returns, group\_number):  
 """  
 Write portfolio returns to a CSV file named Group\_XX.csv with Date as YYYY-MM.  
 Parameters:  
 - portfolio\_returns: pandas Series or DataFrame with monthly returns (as decimals) and Date index  
 - group\_number: int, your two-digit group number (e.g., 1 for Group\_01)  
 Output:  
 - Saves Group\_XX.csv with two columns: Date (YYYY-MM) and Group\_XX (returns as decimals)  
 """  
 # Make two-digit group number  
 group\_str = f"{group\_number:02d}"  
 filename = f"Group\_{group\_str}.csv"  
  
 # Convert returns to DataFrame if Series  
 if isinstance(portfolio\_returns, pd.Series):  
 df = portfolio\_returns.to\_frame(name=f"Group\_{group\_str}")  
 else:  
 df = portfolio\_returns.rename(columns={portfolio\_returns.columns[0]: f"Group\_{group\_str}"})  
  
 # Convert index to PeriodIndex with monthly frequency and format as YYYY-MM  
 if isinstance(df.index, pd.PeriodIndex):  
 if df.index.freq != 'ME':  
 df.index = df.index.to\_timestamp().to\_period('M')  
 elif isinstance(df.index, pd.DatetimeIndex):  
 df.index = df.index.to\_period('M')  
 else:  
 raise ValueError("Index must be a datetime or PeriodIndex")  
  
 # Rename index to Date  
 df.index.name = "Date"  
  
 # Write to CSV  
 df.to\_csv(filename, index=True)  
 print(f"Saved portfolio returns to {filename}")  
  
# # Example usage:  
# dates = pd.date\_range(start="1931-07-01", end="1936-06-01", freq="MS")  
# returns = pd.Series([0.0025, 0.01, -0.005] + [0.001] \* (len(dates) - 3), index=dates)  
# write\_csv(returns, 1)  
  
# group\_01 = (  
# pd.read\_csv(  
# filepath\_or\_buffer='Group\_01.csv',  
# parse\_dates=['Date'],  
# na\_values=[-99.99, -999]  
# )  
# .assign(Date=lambda x: x['Date'].dt.to\_period('M'))  
# .set\_index('Date')  
# .rename\_axis(columns=['Team'])  
# .div(100)  
# )  
write\_csv(momentumData['Momentum Returns'], group\_number=24)

Saved portfolio returns to Group\_24.csv