

# Quantitative Investing

---

MATTHEW ADNER

UNION COLLEGE STUDENT INVESTMENT FUND

# What is it

---

~~“Life is relationships; the rest is just details.”~~ –Gary Smalley

“Make money; the rest is just details.” ~~–someone~~  
trading<sup>^</sup> securities                      –me

# What is it

---

- Quantitative models instead of fundamental analysis
- Data driven

# But why

---

- Unbiased
- Backtesting
  - Short Feedback Loop

# Why not?

---

- Overfitting
- Can be a black box

# Examples (Strategies)

---

- Pairs Trading
- Sentiment Analysis
- High Frequency Trading (HFT)
- Market Making
- Mean Reversion
- Momentum
- Factor Investing

# Examples (Firms)

---



hudson river trading



# Books/Podcasts

---

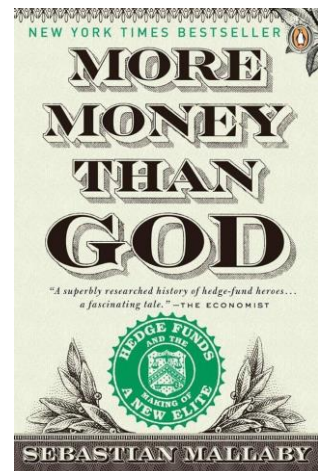
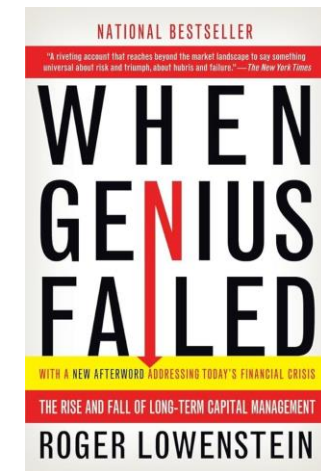
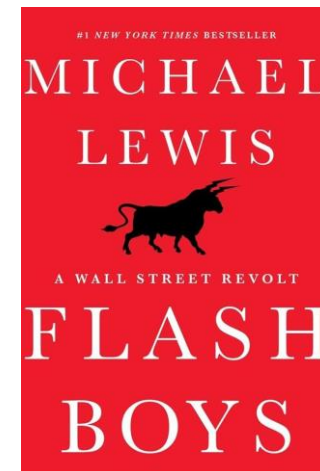
Flash Boys - Michael Lewis

When Genius Failed: The Rise and Fall of Long-Term Capital Management - Roger Lowenstein

More Money Than God: Hedge Funds and the Making of a New Elite - Sebastian Mallaby

Acquired Podcast episode on Renaissance Technologies/Jim Simmons

Flirting with Models

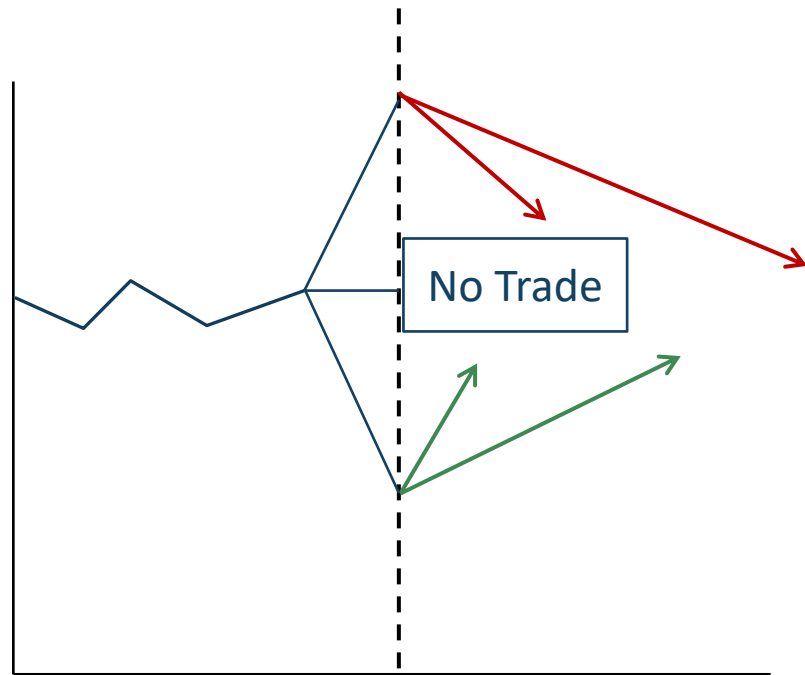




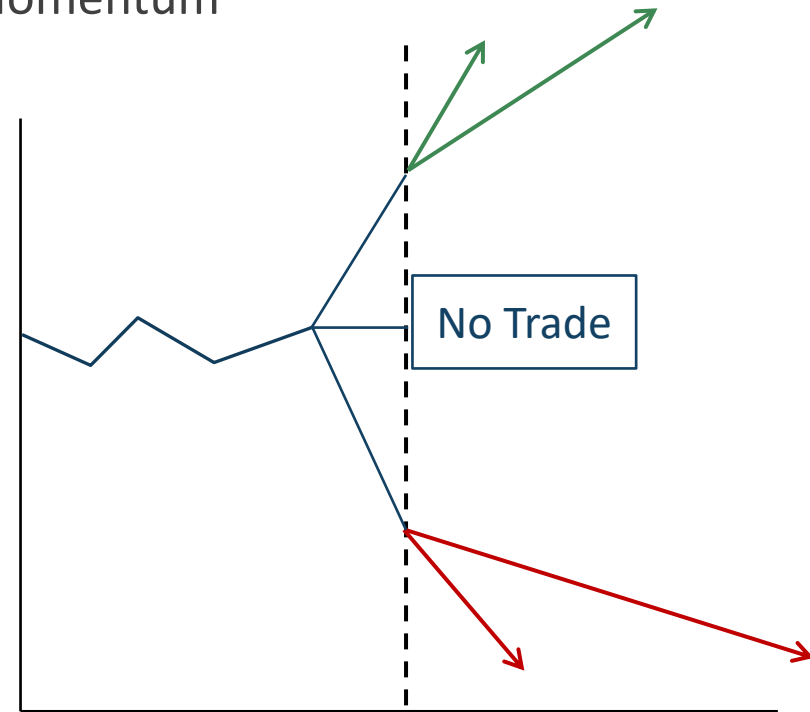
# My Strategy

---

Mean Reversion



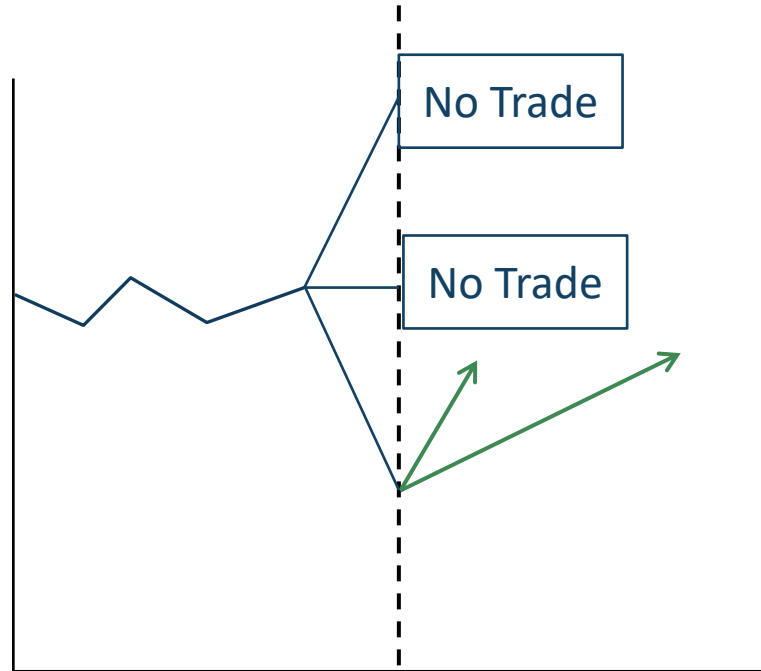
Momentum



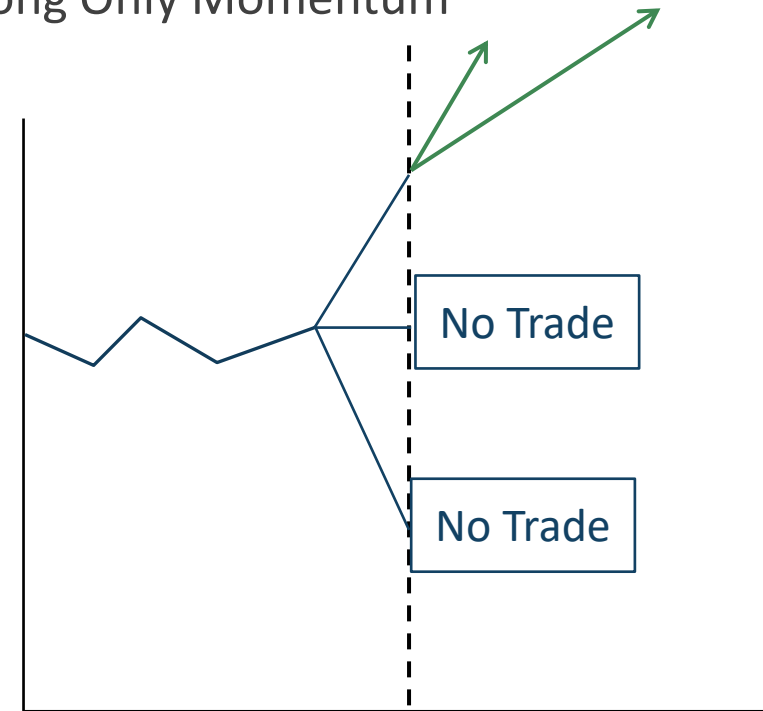
# My Strategy

---

Long Only Mean Reversion



Long Only Momentum



# How

Ticker	Name	Last_Sale	Net_Change	Percent_Change	Market_Cap	Country	IPO_Year	Volume	Sector	Industry	Exchange
A	Agilent Technologies Inc. Common Stock	\$140.09	-1.860	-1.31%	4.025173e+10	United States	1999.0	2612065	Industrials	Biotechnology: Laboratory Analytical Instruments	NYSE
AA	Alcoa Corporation Common Stock	\$33.83	-1.090	-3.121%	8.739647e+09	United States	2016.0	9044364	Industrials	Aluminum	NYSE
AACT	Ares Acquisition Corporation II Class A Ordina...	\$10.81	-0.005	-0.046%	0.000000e+00	NaN	2023.0	564337	Finance	Blank Checks	NYSE
AAM	AA Mission Acquisition Corp. Class A Ordinary ...	\$9.99	0.010	0.10%	0.000000e+00	NaN	2024.0	212652	NaN	NaN	NYSE
AAN	Aarons Holdings Company Inc. Common Stock	\$10.02	-0.020	-0.199%	3.077420e+08	United States	2020.0	1573979	Consumer Discretionary	Diversified Commercial Services	NYSE
...	...	...	...	...	...	...	...	...	...	...	...
ZURA	Zura Bio Limited Class A Ordinary Shares	\$4.44	0.140	3.256%	2.983995e+08	Cayman Islands	NaN	4563183	Health Care	Biotechnology: Biological Products (No Diagnos...	NASDAQ
ZVRA	Zevra Therapeutics Inc. Common Stock	\$8.06	0.040	0.499%	4.240994e+08	United States	NaN	5159085	Health Care	Biotechnology: Pharmaceutical Preparations	NASDAQ
ZVSA	ZyVersa Therapeutics Inc. Common Stock	\$2.20	-0.070	-3.084%	1.668449e+06	United States	2022.0	41342	Health Care	Biotechnology: Pharmaceutical Preparations	NASDAQ
ZYME	Zymeworks Inc. Common Stock	\$13.09	0.040	0.307%	9.300103e+08	United States	NaN	1576190	Health Care	Biotechnology: Pharmaceutical Preparations	NASDAQ
ZYXI	Zynex Inc. Common Stock	\$8.12	-0.210	-2.521%	2.577493e+08	United States	NaN	322447	Health Care	Biotechnology: Electromedical & Electrotherape...	NASDAQ

6726 rows × 11 columns

# How

	Open	High	Low	Close	Adj Close	Volume
Date						
2015-01-02	27.847500	27.860001	26.837500	27.332500	24.373957	212818400
2015-01-05	27.072500	27.162500	26.352501	26.562500	23.687304	257142000
2015-01-06	26.635000	26.857500	26.157499	26.565001	23.689533	263188400
2015-01-07	26.799999	27.049999	26.674999	26.937500	24.021713	160423600
2015-01-08	27.307501	28.037500	27.174999	27.972500	24.944683	237458000
...	...	...	...	...	...	...
2024-09-04	221.660004	221.779999	217.479996	220.850006	220.850006	43840200
2024-09-05	221.630005	225.479996	221.520004	222.380005	222.380005	36615400
2024-09-06	223.949997	225.240005	219.770004	220.820007	220.820007	48423000
2024-09-09	220.820007	221.270004	216.710007	220.910004	220.910004	67180000
2024-09-10	218.919998	221.479996	216.729996	220.110001	220.110001	51591000

2438 rows × 6 columns

	Dates_Numeric	Adj_Close	Size_Category
Date			
2015-01-02	5480	24.819241	mid
2015-01-05	5483	24.120045	mid
2015-01-06	5484	24.122320	mid
2015-01-07	5485	24.460564	mid
2015-01-08	5486	25.400398	mid
...	...	...	...
2024-09-04	9013	220.850006	mega
2024-09-05	9014	222.380005	mega
2024-09-06	9015	220.820007	mega
2024-09-09	9018	220.910004	mega
2024-09-10	9019	220.110001	mega

2438 rows × 3 columns

# How

	Dates_Numeric	Adj_Close	Size_Category	Intercept_60	Dates_Numeric_Coeff_60	Std_Dev_30
Date						
2015-01-02	5480	24.819241	mid	13.937909	0.002000	0.680853
2015-01-05	5483	24.120045	mid	13.959455	0.002003	0.698723
2015-01-06	5484	24.122320	mid	13.970087	0.002005	0.755726
2015-01-07	5485	24.460564	mid	13.980274	0.002006	0.794332
2015-01-08	5486	25.400398	mid	13.994239	0.002009	0.805481
...	...	...	...	...	...	...
2024-09-04	9013	220.850006	mega	95.707011	0.013823	5.893780
2024-09-05	9014	222.380005	mega	95.870873	0.013847	5.852028
2024-09-06	9015	220.820007	mega	96.073247	0.013877	5.836777
2024-09-09	9018	220.910004	mega	96.163029	0.013890	5.796304
2024-09-10	9019	220.110001	mega	96.210564	0.013897	5.763205

2438 rows × 6 columns

# How

	Dates_Numeric	Adj_Close	Intercept_60	Dates_Numeric_Coeff_60	Std_Dev_30	Size_Category	Theo_Value	Theo_Diff
Date								
2015-01-02	5480	24.819241	13.937909	0.002000	0.680853	mid	24.899550	0.117954
2015-01-05	5483	24.120045	13.959455	0.002003	0.698723	mid	24.944296	1.179654
2015-01-06	5484	24.122320	13.970087	0.002005	0.755726	mid	24.965501	1.115723
2015-01-07	5485	24.460564	13.980274	0.002006	0.794332	mid	24.985909	0.661368
2015-01-08	5486	25.400398	13.994239	0.002009	0.805481	mid	25.013085	-0.480848
...	...	...	...	...	...	...	...	...
2024-09-04	9013	220.850006	95.707011	0.013823	5.893780	mega	220.295666	-0.094055
2024-09-05	9014	222.380005	95.870873	0.013847	5.852028	mega	220.688331	-0.289075
2024-09-06	9015	220.820007	96.073247	0.013877	5.836777	mega	221.169907	0.059947
2024-09-09	9018	220.910004	96.163029	0.013890	5.796304	mega	221.419225	0.087853
2024-09-10	9019	220.110001	96.210564	0.013897	5.763205	mega	221.543128	0.248669

2438 rows × 8 columns

```
aapl.loc[:, 'Theo_Value'] = aapl['Intercept_60'] + aapl['Dates_Numeric'] * aapl['Dates_Numeric_Coeff_60']
aapl.loc[:, 'Theo_Diff'] = (aapl['Theo_Value'] - aapl['Adj_Close']) / aapl['Std_Dev_30']
display(aapl)
```

✓ 0.0s

Python

# How

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 7883 entries, 1993-06-02 to 2024-09-20
Columns: 86391 entries, ('Dates_Numeric_Coeff_30', 'BWA') to ('Volume_Value', 'ZYXI')
dtypes: float64(79992), int64(30), object(6369)
memory usage: 5.1+ GB
```

None

	Dates_Numeric_Coeff_30	Volume	Adj_Close	Market_Cap		Intercept_60
	BWA	BDJ	G	ALSN	SRE	BLBX
Date						
1993-06-02		NaN	NaN	NaN	NaN	NaN
1993-06-03		NaN	NaN	NaN	NaN	NaN
1993-06-04		NaN	NaN	NaN	NaN	NaN
1993-06-07		NaN	NaN	NaN	NaN	NaN
1993-06-08		NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...
2024-09-16	0.002723	391600.0	39.000000	7.679461e+09	5.293099e+10	100.169420
2024-09-17	0.002720	383600.0	38.549999	7.708217e+09	5.315259e+10	100.161822
2024-09-18	0.002722	452000.0	38.330002	7.717802e+09	5.251312e+10	100.158849
2024-09-19	0.002724	524400.0	38.480000	7.928676e+09	5.224087e+10	100.162960
2024-09-20	0.002731	332157.0	38.680000	7.934775e+09	5.266507e+10	100.155678

7883 rows × 86391 columns

```

class Position:
    def __init__(self, date_opened, ticker, shares):
        self.date_opened = date_opened
        self.ticker = ticker
        self.shares = shares
        self.cost_basis = shares*stock_data.at[date_opened, ('Adj_Close', ticker)]
        self.current_value = shares*stock_data.at[date_opened, ('Adj_Close', ticker)]
        self.current_theo = shares*stock_data.at[date_opened, ('Adj_Close', ticker)]
        self.current_std_dev = shares*stock_data.at[date_opened, ('std_dev', ticker)]
        self.current_price_diff = stock_data.at[date_opened, ('price_diff', ticker)]
        self.last_date_checked = date_opened

    def __repr__(self):
        return ("Position(date_opened={self.date_opened}, ticker='{self.ticker}', "
                f"{'shares={self.shares}, cost_basis={self.cost_basis}, ' "
                f"{'current_value={self.current_value}, ' "
                f"{'current_theo={self.current_theo}, ' "
                f"{'current_std_dev={self.current_std_dev}, ' "
                f"{'current_price_diff={self.current_price_diff}}',"
                f"{'last_date_checked={self.last_date_checked}}")

    def get_ticker(self):
        return(self.ticker)

    def days_old(self, date:str):
        """returns the number of days old a position is

        Args:
            date (str): date formatted 'YYYY-MM-DD'

        """
        if type(self.date_opened) == str:
            opened = datetime.datetime.strptime(self.date_opened, '%Y-%m-%d').date()
        else:
            opened = self.date_opened
        if type(date)==str:
            current = datetime.datetime.strptime(date, '%Y-%m-%d').date()
        else:
            current = date
        return((current-opened).days)

    def __refresh__(self, current_date):
        self.last_date_checked = current_date
        self.current_value = self.shares*stock_data.at[current_date, ('Adj_Close', self.ticker)]
        self.current_theo = stock_data.at[current_date, ('theo', self.ticker)]
        self.current_std_dev = stock_data.at[current_date, ('std_dev', self.ticker)]
        self.current_price_diff = stock_data.at[current_date, ('price_diff', self.ticker)]
        # recalculate current_value as well as all other values that change over time,
        pass

    def get_current_value(self, current_date):
        self.__refresh__(current_date-current_date)
        return(self.current_value)

    def get_current_theo(self, current_date):
        self.__refresh__(current_date-current_date)
        return(self.current_theo)

    def get_current_std_dev(self, current_date):
        self.__refresh__(current_date-current_date)
        return(self.current_std_dev)

    def get_current_price_diff(self, current_date):
        self.__refresh__(current_date-current_date)
        return(self.current_price_diff)

```

```

class Portfolio:
    def __init__(self, cash: float, date, trading_cost: float, n_assets: int):
        """create Portfolio object"""

        Args:
            cash (float): starting amount of cash in account
            date (TypeDate): starting date
            trading_cost (float, optional): cost of trading- each time we transact, we lose this amount. Defaults to 0.005.
            n_assets (int, optional): number of assets to trade. Defaults to 1.

        """
        self.position_df = pd.DataFrame(columns=['Position', 'Exposure', 'Value', 'Date_Opened', 'Days_Old'])
        self.position_df.index.name = 'ticker'
        # Ensure 'Value' is explicitly set to float type
        self.position_df.loc['cash', 'position'] = ['N/A', 'N/A', float(cash), date, 0] # Cash cost to float
        self.position_df = self.position_df.astype({'Value': 'float64'}) # Ensure Value column remains float
        self._last_date_checked = date
        self.trading_cost = trading_cost
        # log portfolio opening
        # logger.info(f"Portfolio opened on {date}")

    def _repr_(self):
        # return string of "Current Portfolio Values (self.get_portfolio_value(self._last_date_checked)) in"
        returnString = f"Current Portfolio Values (self.get_portfolio_value(self._last_date_checked)) in"
        # for pos in self.position_df['Position']:
        #     returnString += repr(pos)
        return returnString

    def get_cash(self):
        """returns size of cash_position"""
        return(self.position_df.at['cash', 'position', 'Value'])

    def get_exposure(self):
        exposure = self.position_df['Value'].iloc[1:].sum()
        return(exposure)

    def position_count(self):
        return(len(self.position_df.index)-1)

    def open_position(self, position):
        # Assuming position has attributes 'cost_basis', 'get_ticker()', and 'date_opened'
        if position.cost_basis != self.position_df.loc['cash', 'position', 'Value']:
            print("Position not opened; too expensive")
            return
        # Explicit case to ensure correct dtype when modifying Value
        self.position_df.loc['cash', 'position', 'Value'] = float(self.position_df.loc['cash', 'position', 'Value']) - (position.cost_basis)
        # new row for the position
        self.position_df.loc[position.get_ticker()] = [position, position.shares, position.cost_basis, position.date_opened, 0]
        # log purchase

    def Close_position(self, ticker: str, current_date: str):
        """Close a currently open position"""
        self._last_date_checked = current_date
        if ticker == 'cash_position':
            print("Cannot sell cash position")
        else:
            self.position_df = self.position_df.loc[ticker, 'Position'].get_current_value(current_date)
            self.position_df.loc['cash', 'position', 'Value'] = (position.value + (1-self.trading_cost))
            self.position_df.drop(index=ticker, inplace=True)
            # log sale

    def get_portfolio_value(self, date):
        """gets the value of all the positions in the portfolio on the given date"""
        self._refresh_position_df(date)
        self._last_date_checked = date
        value = 0
        for ticker in self.position_df.index:
            value += self.position_df.loc[ticker, 'Value']
        return value

    def refresh_position_df(self, date):
        self._last_date_checked = date
        for ticker in self.position_df.index[1:]: # skipping the cash position
            self.position_df.loc[ticker, 'Value'] = self.position_df.loc[ticker, 'Position'].get_current_value(date)
            self.position_df.loc[ticker, 'Days_Old'] = self.position_df.loc[ticker, 'Position'].days_ald(date)

```

```
best_on_date(stock_data[pd.DataFrame(data=ret, index=tickers, columns=['max', 'min', 'close', 'open', 'high', 'low', 'volume', 'vwap', 'adj_close', 'adj_open', 'adj_high', 'adj_low', 'adj_volume', 'adj_vwap', 'adj_adj_close', 'adj_adj_open', 'adj_adj_high', 'adj_adj_low', 'adj_adj_volume', 'adj_adj_vwap'])], max_or_min = "max", exclude_extremes=True, min_trading_volume=10000, min_trading_value=10000, size_categories=["mega", "large", "mid", "small", "micro"], adj_market_cap=1e+09)

# gets the ticker(s) of the best stock or stocks on a specific date according to the numerical value of either a specific metric, or the absolute value of a specific metric. If there is not enough tickers with data then the return value will be None and a warning will be printed and logged

args:
stock_data (pd.DataFrame): the dataframe to use for reference
date (str): date in the format of "year-month-day"
metric (str): the name of the column to be used for evaluation
abs_val (bool, optional): do you want the highest values in the column, or the values furthest from 0. Defaults to True.
how_many (int, optional): top ... ticker(s). Defaults to 1.

...

# takes a slice of dataframe for current date with .as(date)
# unstack if because .as puts it in long format
# transpose so tickers are the index column
date_slice = stock_data.as(date).unstack().transpose()

# numeric cols = date_slice.select_dtypes(include=["float", "int"]).columns
# date_slice[numerics] = date_slice[numerics].apply(pd.to_numeric, errors='coerce')
# date_slice[metrics] = pd.to_numeric(date_slice[metrics], errors='coerce')
with warnings.catch_warnings():
    warnings.simplefilter("ignore", category=UserWarning)
    date_slice[metrics] = pd.to_numeric(date_slice[metrics], errors='coerce')

# identify to sort relevant data and drop na columns so we don't return useless data
date_slice = date_slice[date_slice['market_cap']>min_market_cap]
date_slice = date_slice[date_slice['size_category'].isin(size_categories)]
date_slice = date_slice[date_slice['volume']>min_trading_volume]
date_slice = date_slice[date_slice['volume_value']>min_trading_volume_value]
date_slice = date_slice[date_slice['close']>min_close, 'volume', metrics].dropna()
# date_slice = date_slice.query(abs(metrics)<10000 exclude extremes)
date_slice = date_slice.dropna(subset=[metrics])

# If there is not enough data to return the number of tickers wanted, returns
if len(date_slice.index)<how_many:
    warnings(f'Parnings not enough data on {date} to run best_on_date. There is only {len(date_slice.index)} non-na6 values in {metrics} but you want the top {how_many}. Returning {None}')
    # logger.warning(warningString)
    return(None)

else:
    if max_or_min == "max":
        [abs_val]
        # gets the id (tickers) of the 2 highest absolute value for price_diff_var
        abs_tickers = abs(date_slice).nlargest(how_many, metrics).index.tolist()
    else:
        # gets the id (tickers) of the 2 highest value for price_diff_var
        abs_tickers = date_slice.nlargest(how_many, metrics).index.tolist()
    return(abs_tickers)
if max_or_min == "min":
    [abs_val]
    # gets the id (tickers) of the 2 highest absolute value for price_diff_var
    min_tickers = abs(date_slice).smallest(how_many, metrics).index.tolist()
    # gets the id (tickers) of the 2 highest value for price_diff_var
    min_tickers = date_slice.smallest(how_many, metrics).index.tolist()
    return(min_tickers)
```



# Results

## Long Only Mid Cap Mean Reversion/ Micro Cap Momentum Portfolio vs SP500



### Portfolio Stats

Annualized Return (CAGR): 12.810%  
Volatility (Standard Deviation): 25.456%  
Max Drawdown: -62.379%  
Alpha: 0.104  
Beta: 0.837  
Sharpe Ratio: 0.807  
Sortino Ratio: 1.319

### SPY Stats

Annualized Return (CAGR): 7.343%  
Volatility (Standard Deviation): 19.083%  
Max Drawdown: -55.189%  
Alpha: 0.000  
Beta: 1.000  
Sharpe Ratio: 0.634  
Sortino Ratio: 0.901

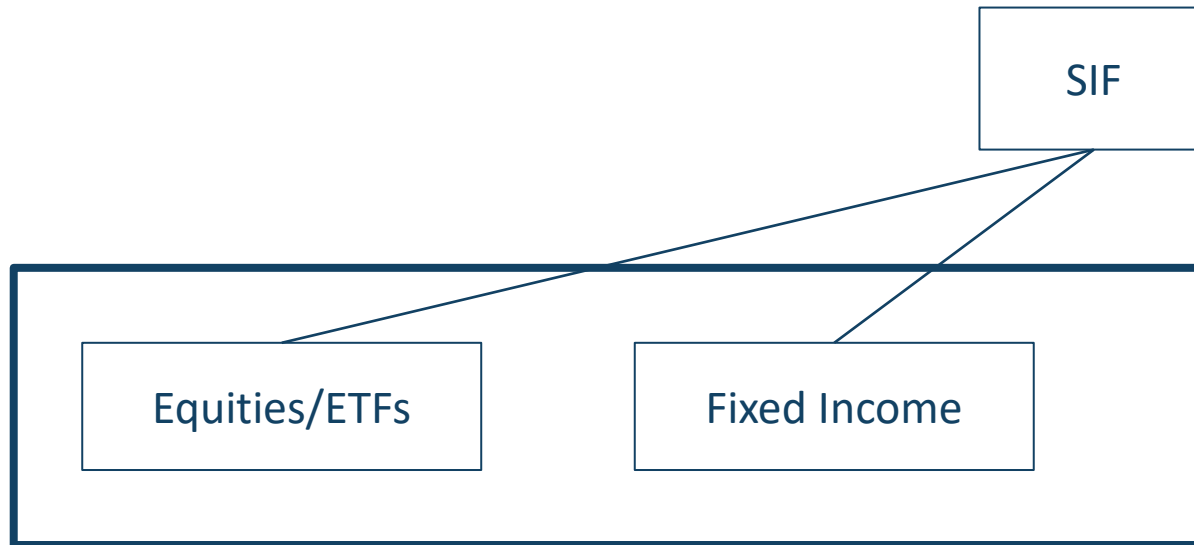
# My Strategy— What's Missing

---

- Dynamic Position Entry/Exit
- More Accurate Trade Cost Estimation
- Trade Analyzer

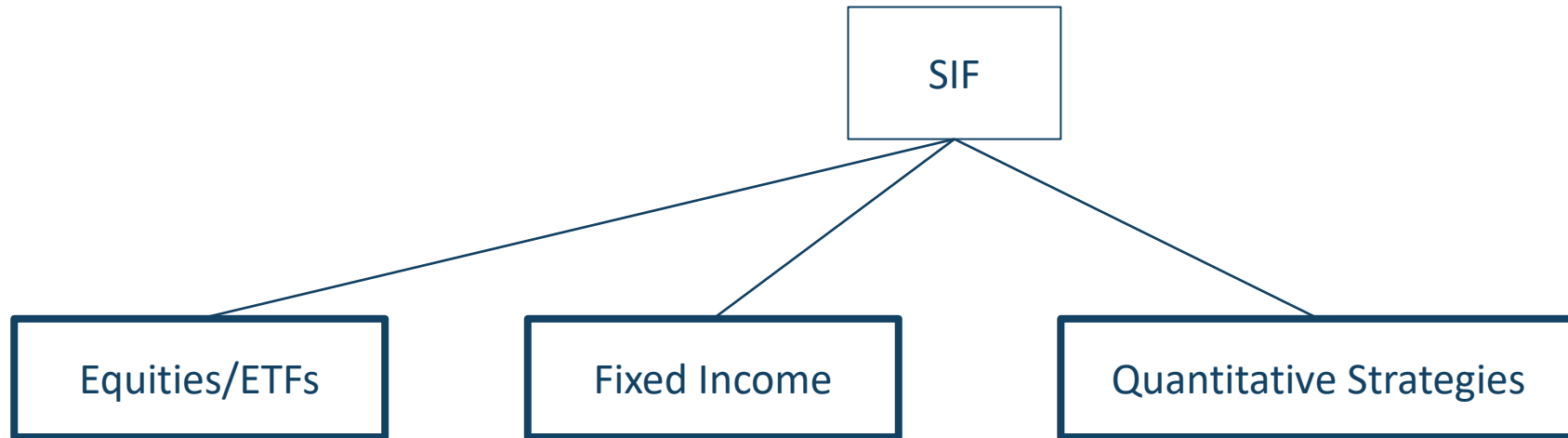
# The Pitch

---



# The Pitch

---



# Questions?

---

LinkedIn  
Let's Connect!



# Sources

---

- <https://www.investopedia.com/terms/p/pairstrade.asp>
- <https://www.schwab.com/learn/story/using-sentiment-analysis-tools-your-trading>
- <https://www.investopedia.com/terms/h/high-frequency-trading.asp>
- <https://www.investopedia.com/terms/m/marketmaker.asp>
- <https://www.investopedia.com/terms/m/meanreversion.asp>
- [https://www.investopedia.com/terms/m/momentum\\_investing.asp](https://www.investopedia.com/terms/m/momentum_investing.asp)
- <https://www.investopedia.com/terms/f/factor-investing.asp>