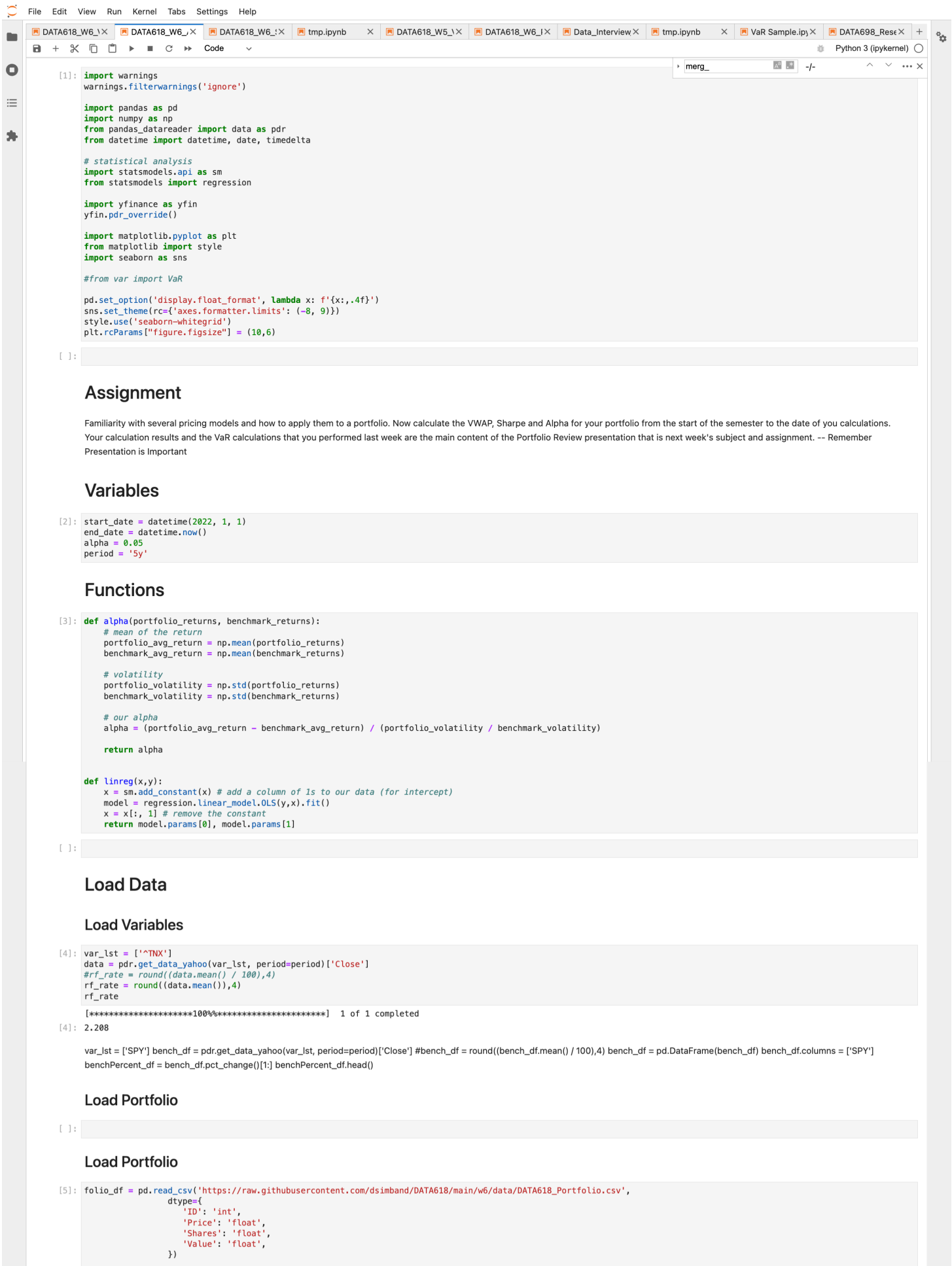


File Edit View Run Kernel Tabs Settings Help

DATA618_W6_\ | DATA618_W6_ | DATA618_W6_ | tmp.ipynb | DATA618_W5_\ | DATA618_W6_I | Data_Interview | tmp.ipynb | VaR Sample.ip\ | DATA698_Rese

Code | Python 3 (ipykernel)

merg_ -/-

[1]:
```
import warnings
warnings.filterwarnings('ignore')

import pandas as pd
import numpy as np
from pandas_datareader import data as pdr
from datetime import datetime, date, timedelta

# statistical analysis
import statsmodels.api as sm
from statsmodels import regression

import yfinance as yfin
yfin.pdr_override()

import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns

#from var import VaR

pd.set_option('display.float_format', lambda x: f'{x:,.4f}')
sns.set_theme(rc={'axes.formatter.limits': (-8, 9)})
style.use('seaborn-whitegrid')
plt.rcParams["figure.figsize"] = (10,6)
```

[ ]:

Assignment

Familiarity with several pricing models and how to apply them to a portfolio. Now calculate the VWAP, Sharpe and Alpha for your portfolio from the start of the semester to the date of you calculations. Your calculation results and the VaR calculations that you performed last week are the main content of the Portfolio Review presentation that is next week's subject and assignment. -- Remember Presentation is Important

Variables

[2]:
```
start_date = datetime(2022, 1, 1)
end_date = datetime.now()
alpha = 0.05
period = '5y'
```

Functions

[3]:
```
def alpha(portfolio_returns, benchmark_returns):
    # mean of the return
    portfolio_avg_return = np.mean(portfolio_returns)
    benchmark_avg_return = np.mean(benchmark_returns)

    # volatility
    portfolio_volatility = np.std(portfolio_returns)
    benchmark_volatility = np.std(benchmark_returns)

    # our alpha
    alpha = (portfolio_avg_return - benchmark_avg_return) / (portfolio_volatility / benchmark_volatility)

    return alpha


def linreg(x,y):
    x = sm.add_constant(x) # add a column of 1s to our data (for intercept)
    model = regression.linear_model.OLS(y,x).fit()
    x = x[:, 1] # remove the constant
    return model.params[0], model.params[1]
```

[ ]:

Load Data

Load Variables

[4]:
```
var_lst = ['^TNX']
data = pdr.get_data_yahoo(var_lst, period=period)['Close']
#rf_rate = round((data.mean() / 100),4)
rf_rate = round((data.mean()),4)
rf_rate
```

```
[**********100%**********]  1 of 1 completed
```

[4]:
```
2.208
```

var_lst = ['SPY'] bench_df = pdr.get_data_yahoo(var_lst, period=period)['Close'] #bench_df = round((bench_df.mean() / 100),4) bench_df = pd.DataFrame(bench_df) bench_df.columns = ['SPY'] benchPercent_df = bench_df.pct_change()[1:] benchPercent_df.head()

Load Portfolio

[ ]:

Load Portfolio

[5]:
```
folio_df = pd.read_csv('https://raw.githubusercontent.com/dsimband/DATA618/main/w6/data/DATA618_Portfolio.csv',
                       dtype={
                          'ID': 'int',
                          'Price': 'float',
                          'Shares': 'float',
                          'Value': 'float',
                       })

folio_df = folio_df[folio_df['Shares'] > 0]
```

```
folio_df = folio_df[folio_df['Shares'] > 0]
folio_df = folio_df.groupby(['Ticker','BondName','Class','Sub_Class'])[['Shares','Value']].sum()
folio_df.reset_index(inplace=True)

# calculate portfilio percentage
portfolio_total = folio_df['Value'].sum()
folio_df['port_percent'] = folio_df['Value'] / portfolio_total

# Class Portfolios
folio_econ_df = folio_df[folio_df['Class'] == 'Economically Sensitive']
folio_int_df = folio_df[folio_df['Class'] == 'Interest Rate Sensitive']

folio_df
```

[5]:

	Ticker	BondName	Class	Sub_Class	Shares	Value	port_percent
0	ALTVX	AB Municipal Income National Advisor	Interest Rate Sensitive	Very-High-Quality Municipal Bonds	106,383.0000	1,000.0000	0.0119
1	BLUEX	AMG Veritas Global Real Return I	Economically Sensitive	Long/Short Equities	14,201.0000	500.0000	0.0060
2	BPLSX	Boston Partners Long/Short Equity Instl	Economically Sensitive	Long/Short Equities	49,967.0000	750.0000	0.0090
3	C_A_S_H	Cash	Cash	Cash	4,469,000.0000	4,469.0000	0.0534
4	DFAR	Dimensional US Real Estate ETF	Economically Sensitive	Publicly Traded REITs	108,467.0000	2,250.0000	0.0269
5	FREL	Fidelity MSCI Real Estate ETF	Economically Sensitive	Publicly Traded REITs	115,479.0000	2,800.0000	0.0334
6	FSMD	Fidelity Small-Mid Multifactor ETF	Economically Sensitive	U.S. Equities	120,705.0000	4,000.0000	0.0478
7	GARIX	Gotham Absolute Return Institutional	Economically Sensitive	Long/Short Equities	25,278.0000	500.0000	0.0060
8	GCHDX	Gotham Hedged Core Institutional	Economically Sensitive	Long/Short Equities	46,211.0000	500.0000	0.0060
9	ICBFX	Invesco Corporate Bond R6	Interest Rate Sensitive	Very-High-Quality Corporate Bonds	166,389.0000	1,000.0000	0.0119
10	IGMWX	Voya GNMA Income W	Interest Rate Sensitive	U.S. Government Bonds	763,889.0000	5,500.0000	0.0657
11	IGRO	iShares International Dividend Gr ETF	Economically Sensitive	International Equities	32,610.0000	2,000.0000	0.0239
12	IMCB	iShares Morningstar Mid-Cap ETF	Economically Sensitive	U.S. Equities	64,494.0000	4,000.0000	0.0478
13	ISCF	iShares MSCI Intl Small-Cap Mltfct ETF	Economically Sensitive	International Equities	98,007.0000	3,000.0000	0.0358
14	IVV	iShares Core S&P 500 ETF	Economically Sensitive	U.S. Equities	8,912.0000	4,000.0000	0.0478
15	JPEM	JPMorgan Diversified Return EMkts Eq ETF	Economically Sensitive	Emerging Market Equities	78,561.0000	4,000.0000	0.0478
16	JPRE	JPMorgan Realty Income ETF	Economically Sensitive	Publicly Traded REITs	60,695.0000	2,500.0000	0.0299
17	MEAR	iShares Short Maturity Municipal Bd ETF	Interest Rate Sensitive	Very-High-Quality Municipal Bonds	20,060.0000	1,000.0000	0.0119
18	MMIN	IQ MacKay Municipal Insured ETF	Interest Rate Sensitive	Very-High-Quality Municipal Bonds	42,553.0000	1,000.0000	0.0119
19	MMIT	IQ MacKay Municipal Intermediate ETF	Interest Rate Sensitive	Very-High-Quality Municipal Bonds	41,771.0000	1,000.0000	0.0119
20	PEX	ProShares Global Listed Private Equity	Economically Sensitive	Private Equity	150,754.0000	4,200.0000	0.0502
21	PFRAX	PIMCO International Bond (USD-Hdg) Adm	Interest Rate Sensitive	Very-High-Quality Corporate Bonds	105,820.0000	1,000.0000	0.0119
22	PSAIX	PIMCO Global Advantage Strategy Bd Instl	Interest Rate Sensitive	Very-High-Quality Corporate Bonds	103,413.0000	1,000.0000	0.0119
23	SCHD	Schwab US Dividend Equity ETF™	Economically Sensitive	U.S. Equities	41,027.0000	3,000.0000	0.0358
24	SNDPX	AB Diversified Municipal	Interest Rate Sensitive	Very-High-Quality Municipal Bonds	73,529.0000	1,000.0000	0.0119
25	SWRSX	Schwab Treasury Infl Protected Secs Idx	Interest Rate Sensitive	U.S. Government Bonds	592,300.0000	6,000.0000	0.0717
26	VBR	Vanguard Small-Cap Value ETF	Economically Sensitive	U.S. Equities	6,120.0000	1,000.0000	0.0119
27	VFIUX	Vanguard Interm-Term Treasury Adm	Interest Rate Sensitive	U.S. Government Bonds	566,426.0000	5,500.0000	0.0657
28	VGCAX	Vanguard Global Credit Bond Admiral	Interest Rate Sensitive	Very-High-Quality Corporate Bonds	54,437.0000	1,000.0000	0.0119
29	VICSX	Vanguard Interm-Term Corp Bd Idx Admiral	Interest Rate Sensitive	Very-High-Quality Corporate Bonds	47,962.0000	2,000.0000	0.0239
30	VIGI	Vanguard Intl Div Apprec ETF	Economically Sensitive	International Equities	40,481.0000	3,000.0000	0.0358
31	VNQ	Vanguard Real Estate ETF	Economically Sensitive	Publicly Traded REITs	40,221.0000	3,250.0000	0.0388
32	VO	Vanguard Mid-Cap ETF	Economically Sensitive	U.S. Equities	9,234.0000	2,000.0000	0.0239
33	VONG	Vanguard Russell 1000 Growth ETF	Economically Sensitive	U.S. Equities	41,501.0000	3,000.0000	0.0358
34	VTEB	Vanguard Tax-Exempt Bond ETF	Interest Rate Sensitive	Very-High-Quality Municipal Bonds	20,333.0000	1,000.0000	0.0119

[6]:
```
folio_df.groupby(['Class'])[['Value']].sum()
```

[6]:

	Value
Class	
Cash	4,469.0000
Economically Sensitive	50,250.0000
Interest Rate Sensitive	29,000.0000

[7]:
```
initial_investment = folio_df['Value'].sum() * 1000
initial_investment
```

[7]:
```
83719000.0
```

Portfolio

Load Pricing Time Series

[8]:
```
# ticker symbols
ticker_lst = list(folio_df['Ticker'])
print('ticker #:', len(ticker_lst))


# portfolio weights
weight_lst = (folio_df['port_percent'].values)
print('price #:', len(weight_lst))


#Download closing prices
price_df = pdr.get_data_yahoo(ticker_lst, period=period)['Close']

price_df['C_A_S_H'] = 1
print('price_df #:', price_df.shape)

#From the closing prices, calculate periodic returns
return_df = price_df.pct_change()
print('return_df #:', len(return_df.columns))
```

```
ticker #: 35
price #: 35
```

```
[**********100%**********]  35 of 35 completed

1 Failed download:
['C_A_S_H']: Exception('%ticker%: No data found, symbol may be delisted')

price_df #: (1258, 35)
return_df #: 35
```

Portfolio Returns

[9]:
```
port_ret_weighted= return_df.mul(weight_lst, axis=1)
print('port_ret_weighted #:', port_ret_weighted.shape)

return_df['Portfolio'] = port_ret_weighted.sum(axis=1)
print('return_df #:', return_df.shape)
```

```
port_ret_weighted #: (1258, 35)
return_df #: (1258, 36)
```

Calculate Portfolio Value

[10]:
```
shares_df = folio_df[['Ticker','Shares']]
shares_df.set_index('Ticker', drop=True, inplace=True)
#shares_df[['Shares']].round(0)
shares_df.head()
```

[10]:

Ticker	Shares
ALTVX	106,383.0000
BLUEX	14,201.0000
BPLSX	49,967.0000
C_A_S_H	4,469,000.0000
DFAR	108,467.0000

[11]:
```
m_df = price_df.copy()
m_df.reset_index(inplace=True)
m_df = m_df.melt(id_vars=['Date'])
m_df.columns = ['Date','Ticker','Price']
m_df.head()
```

[11]:

	Date	Ticker	Price
0	2018-10-08	ALTVX	9.9800
1	2018-10-09	ALTVX	9.9600
2	2018-10-10	ALTVX	9.9400
3	2018-10-11	ALTVX	9.9500
4	2018-10-12	ALTVX	9.9500

[12]:
```
l_df = folio_df[['Ticker','Class','Sub_Class']] #.drop_duplicates()
print('l_df: ',l_df.shape)

#l_df.head()
```

```
l_df:  (35, 3)
```

[13]:
```
merge_df = m_df.merge(l_df, how='left', left_on='Ticker', right_on='Ticker').merge(shares_df,how='left' ,left_on='Ticker', right_on='Ticker')
merge_df['share_value'] = merge_df['Price'] * merge_df['Shares']
merge_df['perc_share_value'] = merge_df.groupby(['Ticker'])[['share_value']].pct_change()

print('merge_df: ',merge_df.shape)
#merge_df.head()
```

```
merge_df:  (44030, 8)
```

[14]:
```
folioValue_df = merge_df.groupby(['Date'])[['share_value']].sum().reset_index()
folioValue_df['pct_change'] = folioValue_df['share_value'].pct_change()
folioValue_df = folioValue_df[1:]

print('folioValue_df: ',folioValue_df.shape)
```

```
folioValue_df:  (1257, 3)
```

[15]:
```
folioValue_df[folioValue_df['Date'] == '2023-09-11']
```

[15]:

	Date	share_value	pct_change
1238	2023-09-11	83,004,840.7077	0.0025

[16]:
```
folioValue_df[folioValue_df['Date'] == '2023-10-06']
```

[16]:

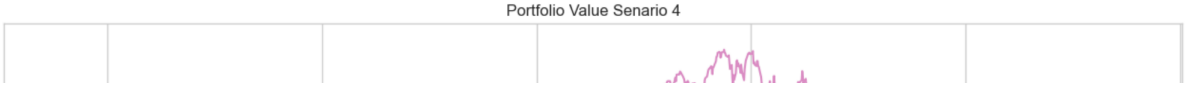
	Date	share_value	pct_change
1257	2023-10-06	79,376,544.6729	0.0037

Graph Porfolio Values

[17]:
```
t_df = merge_df.groupby(['Date','Class','Sub_Class'])[['share_value']].sum().reset_index()
print('t_df: ',t_df.shape)
```

```
t_df:  (12580, 4)
```

[18]:
```
# Graph
fig, ax = plt.subplots(figsize=(16,8))
plt.ticklabel_format(style='plain', axis='y')

# Plot
sns.set_style("white")
g = sns.lineplot(data=t_df, x="Date", y="share_value", hue='Sub_Class')
g.set_yticklabels(['{:,.1f}'.format(x) + ' M' for x in g.get_yticks()/1000000])

ax.set(title='Portfolio Value Senario 4');
```



Alpha Calculations

Economically Sensitive

[19]:
```
tickers='^DJA'
#tickers = 'FHNFX'
bench_data = pdr.get_data_yahoo(tickers=tickers, period=period)
bench_econ_df= pd.DataFrame(bench_data['Close'])
bench_econ_df.columns = ['bench']
bench_econ_df['precent_bench']= bench_econ_df['bench'].pct_change(1)

print('bench_econ_df:', bench_econ_df.shape)
#bench_econ_df
```

```
[**********100%**********]  1 of 1 completed
bench_econ_df: (1258, 2)
```

[20]:
```
ticker_lst = folio_econ_df['Ticker']
print('ticker_lst:', len(ticker_lst))

#value_econ_df = merge_df[merge_df['Ticker'].isin(ticker_lst)]
#print('value_econ_df:', value_econ_df.shape)
```

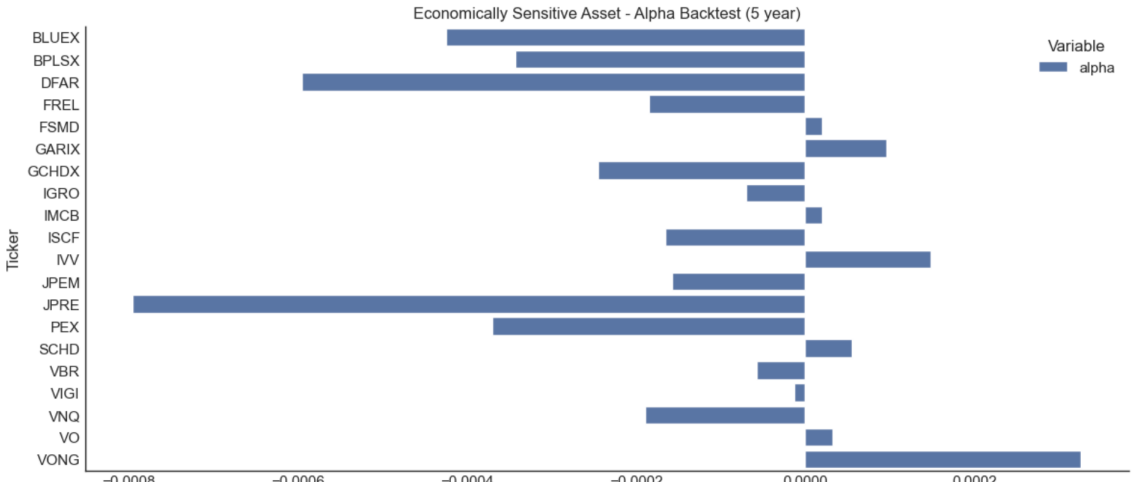
```
ticker_lst: 20
```

[21]:
```
alpha_econ_df = pd.DataFrame(columns=['Ticker','alpha','beta'])

for t in ticker_lst:
    X = bench_econ_df['precent_bench'].values
    y = merge_df[merge_df['Ticker'] == t]['perc_share_value'].values

    t_df = pd.DataFrame({'X':X,'y':y})
    t_df = t_df[np.isfinite(t_df).all(1)]

    alpha, beta = linreg(t_df.X.values,t_df.y.values)
    alpha_econ_df.loc[len(alpha_econ_df.index)] = [t, alpha, beta]

print('alpha_econ_df:', alpha_econ_df.shape)
```

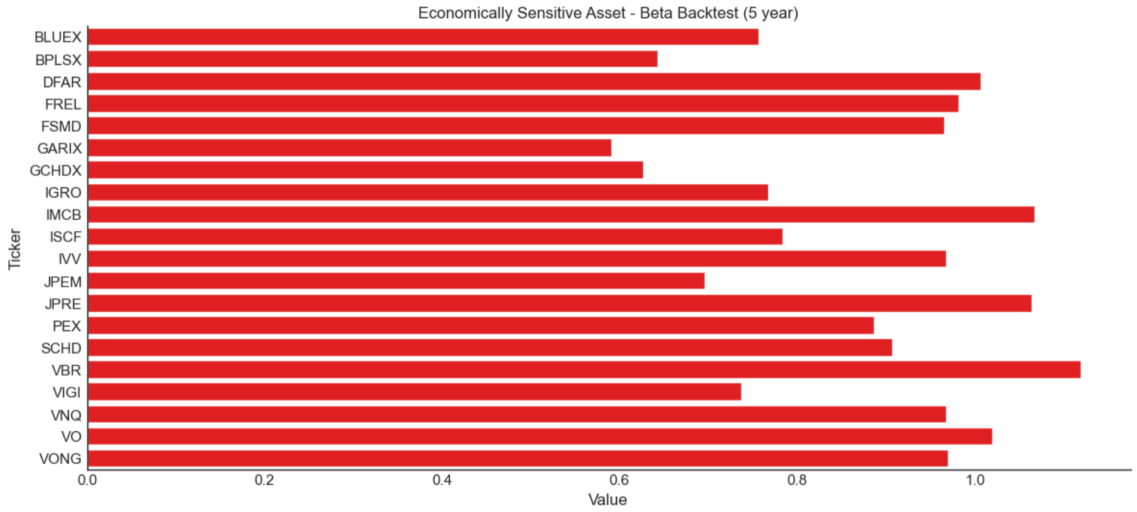
```
alpha_econ_df: (20, 3)
```

Graph

[22]:
```
fig, ax = plt.subplots(figsize=(14,6))
t_df = alpha_econ_df.melt(id_vars='Ticker').rename(columns=str.title)
t_df = t_df[t_df['Variable'] == 'alpha']
g = sns.barplot(x='Value', y='Ticker', hue='Variable',data=t_df, ax=ax)
sns.despine(fig)
ax.set(title='Economically Sensitive Asset - Alpha Backtest (5 year)');
```



[23]:
```
fig, ax = plt.subplots(figsize=(14,6))
t_df = alpha_econ_df.melt(id_vars='Ticker').rename(columns=str.title)
t_df = t_df[t_df['Variable'] == 'beta']
g = sns.barplot(x='Value', y='Ticker', color='red', data=t_df, ax=ax)
sns.despine(fig)
ax.set(title='Economically Sensitive Asset - Beta Backtest (5 year)');
```



[ ]:
```

```

Interest Rate Sensitive

[24]:
```
#tickers='^DJA'
tickers = 'FHNFX'
bench_data = pdr.get_data_yahoo(tickers=tickers, period=period)
bench_int_df= pd.DataFrame(bench_data['Close'])
bench_int_df.columns = ['bench']
bench_int_df['precent_bench']= bench_int_df['bench'].pct_change(1)

print('bench_int_df #:', bench_int_df.shape)
#bench_int_df
```

```
[**********100%**********]  1 of 1 completed
bench_int_df #: (1258, 2)
```

[25]:
```
ticker_lst = folio_int_df['Ticker']
print('ticker_lst:', len(ticker_lst))
```

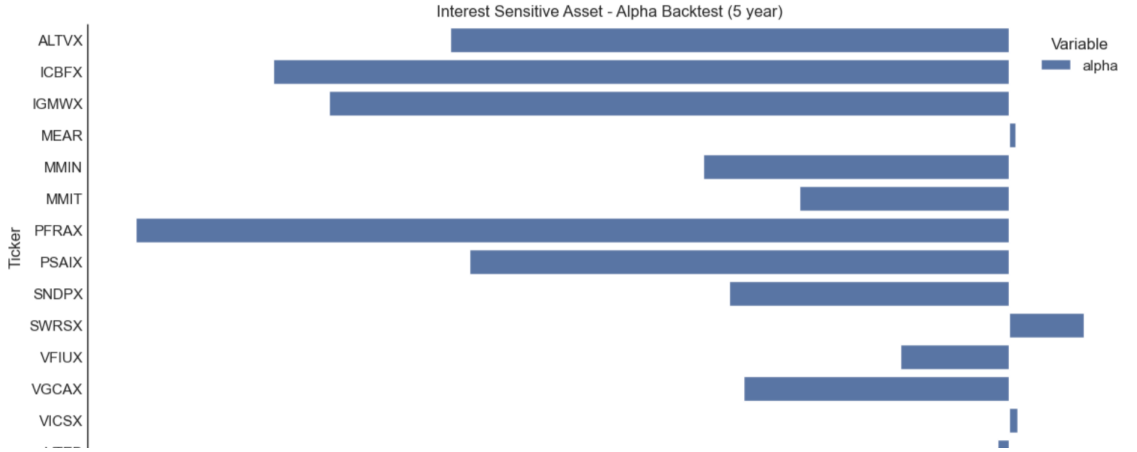
```
ticker_lst: 14
```

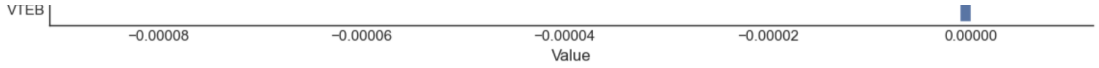
[26]:
```
alpha_int_df = pd.DataFrame(columns=['Ticker','alpha','beta'])

for t in ticker_lst:
    X = bench_int_df['precent_bench'].values
    y = merge_df[merge_df['Ticker'] == t]['perc_share_value'].values

    t_df = pd.DataFrame({'X':X,'y':y})
    t_df = t_df[np.isfinite(t_df).all(1)]

    alpha, beta = linreg(t_df.X.values,t_df.y.values)
    alpha_int_df.loc[len(alpha_int_df.index)] = [t, alpha, beta]

print('alpha_int_df:', alpha_int_df.shape)
```

```
alpha_int_df: (14, 3)
```

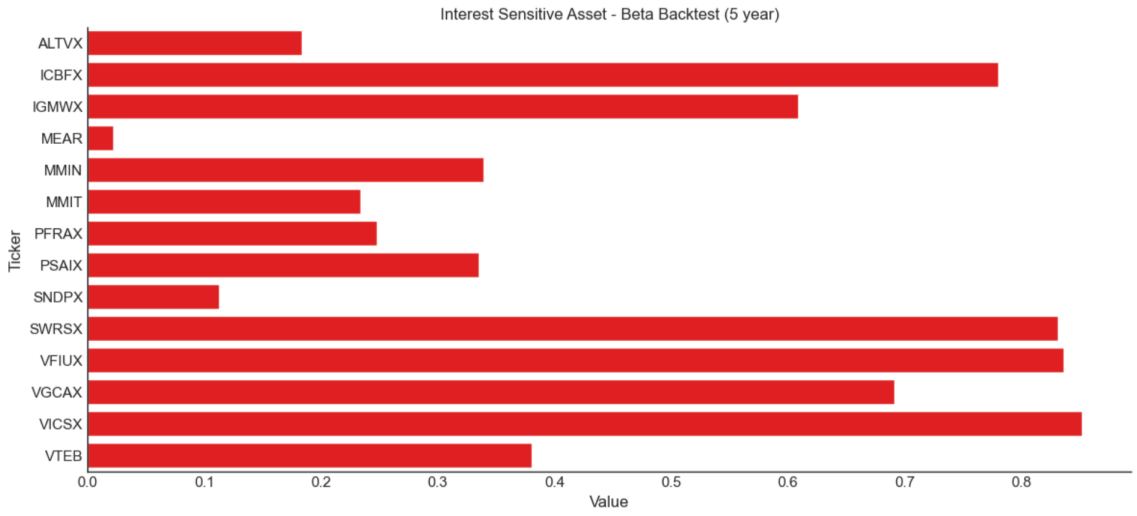
Graph

[27]:
```
fig, ax = plt.subplots(figsize=(14,6))
t_df = alpha_int_df.melt(id_vars='Ticker').rename(columns=str.title)
t_df = t_df[t_df['Variable'] == 'alpha']
g = sns.barplot(x='Value', y='Ticker', hue='Variable',data=t_df, ax=ax)
sns.despine(fig)
ax.set(title='Interest Sensitive Asset - Alpha Backtest (5 year)');
```





[28]:
```
fig, ax = plt.subplots(figsize=(14,6))
t_df = alpha_int_df.melt(id_vars='Ticker').rename(columns=str.title)
t_df = t_df[t_df['Variable'] == 'beta']
g = sns.barplot(x='Value', y='Ticker', color='red',data=t_df, ax=ax)
sns.despine(fig)
ax.set(title='Interest Sensitive Asset - Beta Backtest (5 year)');
```



[ ]:

Simple 0 13 Python 3 (ipykernel) | Idle Mode: Command Ln 1, Col 1 DATA618_W6_Alpha.ipynb 1