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.ipy | DATA698_Rese

Code | Python 3 (ipykernel)

[55]:
```
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

import yfinance as yfin
yfin.pdr_override()

import matplotlib.pyplot as plt
import seaborn as sns

#from var import VaR

pd.set_option('display.float_format', lambda x: f'{x:,.6f}')
```

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

Sharpe Ratio

Sharpe ratio is a measure for calculating risk-adjusted return. It is the ratio of the excess expected return of investment (over risk-free rate) per unit of volatility or standard deviation.

Variables

[56]:
```
period = '5y'
```

Load Data

Load Variables

[57]:
```
#var_lst = ['^SPX']
#var_lst = ['^TNX']
var_lst = ['TNX']
data = pdr.get_data_yahoo(var_lst, period=period)['Close']
rf_df = data.pct_change()
rf_rate = rf_df.mean()
rf_rate
```

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

[57]:
```
-0.016533451391655823
```

[ ]:

Load Portfolio

[58]:
```
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.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
```

[58]:

	Ticker	BondName	Class	Sub_Class	Shares	Value	port_percent
0	ALTVX	AB Municipal Income National Advisor	Interest Rate Sensitive	Very-High-Quality Municipal Bonds	106,383.000000	1,000.000000	0.011945
1	BLUEX	AMG Veritas Global Real Return I	Economically Sensitive	Long/Short Equities	14,201.000000	500.000000	0.005972
2	BPLSX	Boston Partners Long/Short Equity Instl	Economically Sensitive	Long/Short Equities	49,967.000000	750.000000	0.008959
3	C_A_S_H	Cash	Cash	Cash	4,469,000.000000	4,469.000000	0.053381
4	DFAR	Dimensional US Real Estate ETF	Economically Sensitive	Publicly Traded REITs	108,467.000000	2,250.000000	0.026876
5	FREL	Fidelity MSCI Real Estate ETF	Economically Sensitive	Publicly Traded REITs	115,479.000000	2,800.000000	0.033445
6	FSMD	Fidelity Small-Mid Multifactor ETF	Economically Sensitive	U.S. Equities	120,705.000000	4,000.000000	0.047779
7	GARIX	Gotham Absolute Return Institutional	Economically Sensitive	Long/Short Equities	25,278.000000	500.000000	0.005972
8	GCHDX	Gotham Hedged Core Institutional	Economically Sensitive	Long/Short Equities	46,211.000000	500.000000	0.005972
9	ICBFX	Invesco Corporate Bond R6	Interest Rate Sensitive	Very-High-Quality Corporate Bonds	166,389.000000	1,000.000000	0.011945
10	IGMWX	Voya GNMA Income W	Interest Rate Sensitive	U.S. Government Bonds	763,889.000000	5,500.000000	0.065696
11	IGRO	iShares International Dividend Gr ETF	Economically Sensitive	International Equities	32,610.000000	2,000.000000	0.023889
12	IMCB	iShares Morningstar Mid-Cap ETF	Economically Sensitive	U.S. Equities	64,494.000000	4,000.000000	0.047779
13	ISCF	iShares MSCI Intl Small-Cap Mltfct ETF	Economically Sensitive	International Equities	98,007.000000	3,000.000000	0.035834
14	IVV	iShares Core S&P 500 ETF	Economically Sensitive	U.S. Equities	8,912.000000	4,000.000000	0.047779
15	JPEM	JPMorgan Diversified Return EMkts Eq ETF	Economically Sensitive	Emerging Market Equities	78,561.000000	4,000.000000	0.047779

16	JPRE	JPMorgan Realty Income ETF	Economically Sensitive	Publicly Traded REITs	60,695.000000	2,500.000000	0.029862
17	MEAR	iShares Short Maturity Municipal Bd ETF	Interest Rate Sensitive	Very-High-Quality Municipal Bonds	20,060.000000	1,000.000000	0.011945
18	MMIN	IQ MacKay Municipal Insured ETF	Interest Rate Sensitive	Very-High-Quality Municipal Bonds	42,553.000000	1,000.000000	0.011945
19	MMIT	IQ MacKay Municipal Intermediate ETF	Interest Rate Sensitive	Very-High-Quality Municipal Bonds	41,771.000000	1,000.000000	0.011945
20	PEX	ProShares Global Listed Private Equity	Economically Sensitive	Private Equity	150,754.000000	4,200.000000	0.050168
21	PFRAX	PIMCO International Bond (USD-Hdg) Adm	Interest Rate Sensitive	Very-High-Quality Corporate Bonds	105,820.000000	1,000.000000	0.011945
22	PSAIX	PIMCO Global Advantage Strategy Bd Instl	Interest Rate Sensitive	Very-High-Quality Corporate Bonds	103,413.000000	1,000.000000	0.011945
23	SCHD	Schwab US Dividend Equity ETF™	Economically Sensitive	U.S. Equities	41,027.000000	3,000.000000	0.035834
24	SNDPX	AB Diversified Municipal	Interest Rate Sensitive	Very-High-Quality Municipal Bonds	73,529.000000	1,000.000000	0.011945
25	SWRSX	Schwab Treasury Infl Protected Secs Idx	Interest Rate Sensitive	U.S. Government Bonds	592,300.000000	6,000.000000	0.071668
26	VBR	Vanguard Small-Cap Value ETF	Economically Sensitive	U.S. Equities	6,120.000000	1,000.000000	0.011945
27	VFIUX	Vanguard Interm-Term Treasury Adm	Interest Rate Sensitive	U.S. Government Bonds	566,426.000000	5,500.000000	0.065696
28	VGCAX	Vanguard Global Credit Bond Admiral	Interest Rate Sensitive	Very-High-Quality Corporate Bonds	54,437.000000	1,000.000000	0.011945
29	VICSX	Vanguard Interm-Term Corp Bd Idx Admiral	Interest Rate Sensitive	Very-High-Quality Corporate Bonds	47,962.000000	2,000.000000	0.023889
30	VIGI	Vanguard Intl Div Apprec ETF	Economically Sensitive	International Equities	40,481.000000	3,000.000000	0.035834
31	VNQ	Vanguard Real Estate ETF	Economically Sensitive	Publicly Traded REITs	40,221.000000	3,250.000000	0.038820
32	VO	Vanguard Mid-Cap ETF	Economically Sensitive	U.S. Equities	9,234.000000	2,000.000000	0.023889
33	VONG	Vanguard Russell 1000 Growth ETF	Economically Sensitive	U.S. Equities	41,501.000000	3,000.000000	0.035834
34	VTEB	Vanguard Tax-Exempt Bond ETF	Interest Rate Sensitive	Very-High-Quality Municipal Bonds	20,333.000000	1,000.000000	0.011945

[59]:
```
folio_df.groupby(['Class'])[['Value']].sum()
```

[59]:

Class	Value
Cash	4,469.000000
Economically Sensitive	50,250.000000
Interest Rate Sensitive	29,000.000000

[60]:
```
initial_investment = folio_df['Value'].sum() * 1000
initial_investment
```

[60]:
```
83719000.0
```

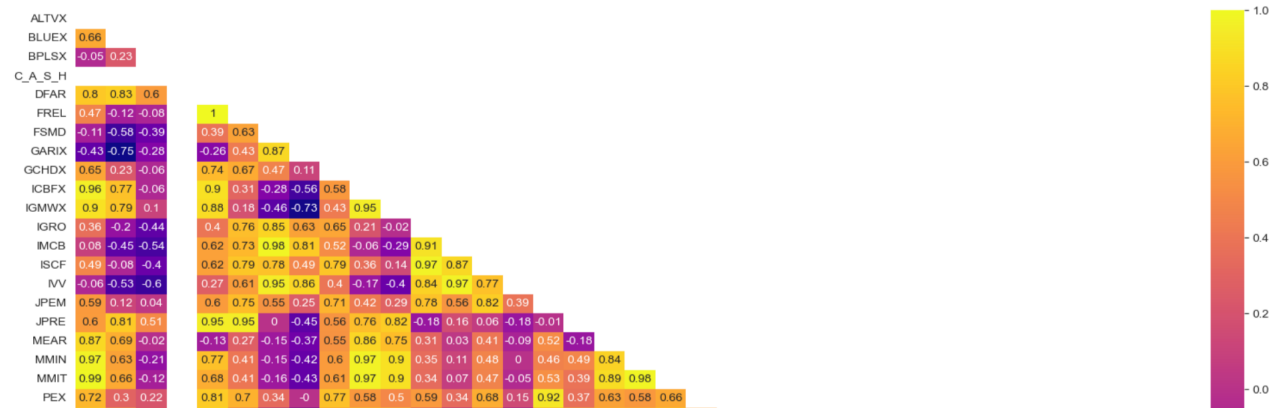
Portfolio

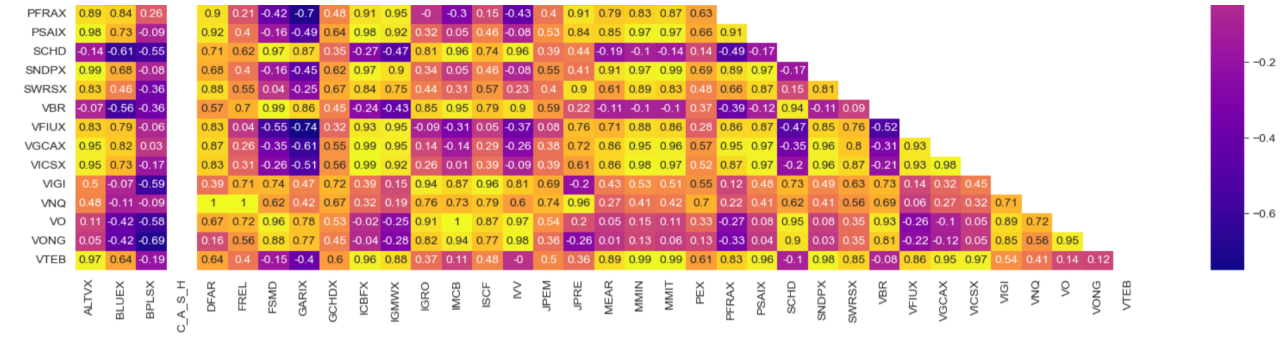
Load Pricing Time Series

[61]:
```
# 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
```

[62]:
```
plt.figure(figsize=(20,10))
mask = np.zeros_like(price_df.corr())
mask[np.triu_indices_from(mask)] = True
sns.set_style("white")
_p = sns.heatmap(price_df.corr().round(2),
                 annot=True, mask=mask,
                 cmap="plasma", annot_kws={"size": 10})
```





Portfolio Returns

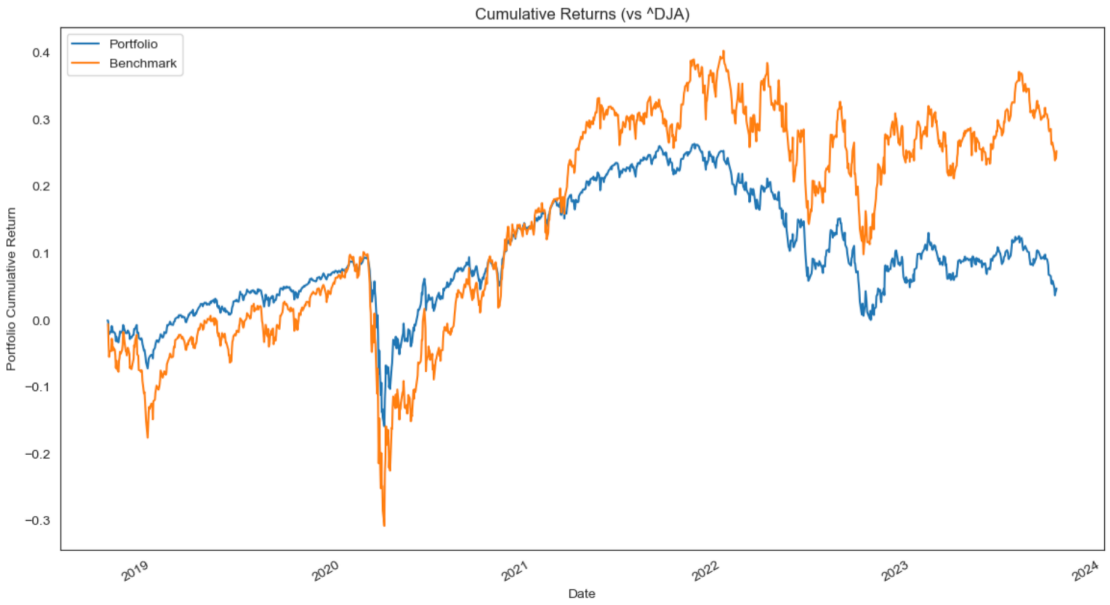
[63]:
```
port_ret_weighted= return_df.mul(weight_lst, axis=1)
print('port_ret_weighted #:', port_ret_weighted.shape)

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

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

[64]:
```
tickers='^DJA'

DJA = pdr.get_data_yahoo(tickers=tickers, period=period)
return_df['Benchmark']= DJA['Close']
return_df['Benchmark']= return_df['Benchmark'].pct_change(1) #.dropna()

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

```
[**********100%**********]  1 of 1 completed
return_df #: (1258, 37)
```

[65]:
```
cum_ret_port = ((1 + return_df[['Portfolio','Benchmark']]).cumprod()-1)
cum_ret_port.plot(label='Cumulative Returns of the Portfolio' ,figsize=(14,8),title='Cumulative Returns (vs ^DJA)')
_=plt.xlabel('Date' )
_=plt.ylabel('Portfolio Cumulative Return' )
```



[66]:
```
port_bench = pd.concat([return_df['Portfolio' ],return_df['Benchmark' ]],axis=1 ).dropna()
port_bench.columns=['Portfolio',' Benchmark' ]
correlation=port_bench.corr()
correlation
```

[66]:

	Portfolio	Benchmark
Portfolio	1.000000	0.931855
Benchmark	0.931855	1.000000

Sharpe Ratio

[67]:
```
return_df['RF Rate']= rf_rate

sharpe_ratio=((return_df['Portfolio'].mean() - return_df['RF Rate'].mean()))/return_df['Portfolio' ].std()
sh = round(sharpe_ratio,4)
print(f"Sharpe Ratio: {sh:.7f} (50% {sh*0.5:.7f})")

avg = return_df['Portfolio'].mean() * 100
print(f"Portfolio Return Mean: {avg:.7f}")

std = return_df['Portfolio' ].std()
print(f"Portfolio Return Standard Deviation: {std:.7f}")
```

```
Sharpe Ratio: 2.2798000 (50% 1.1399000)
Portfolio Return Mean: 0.0063732
Portfolio Return Standard Deviation: 0.0072800
```

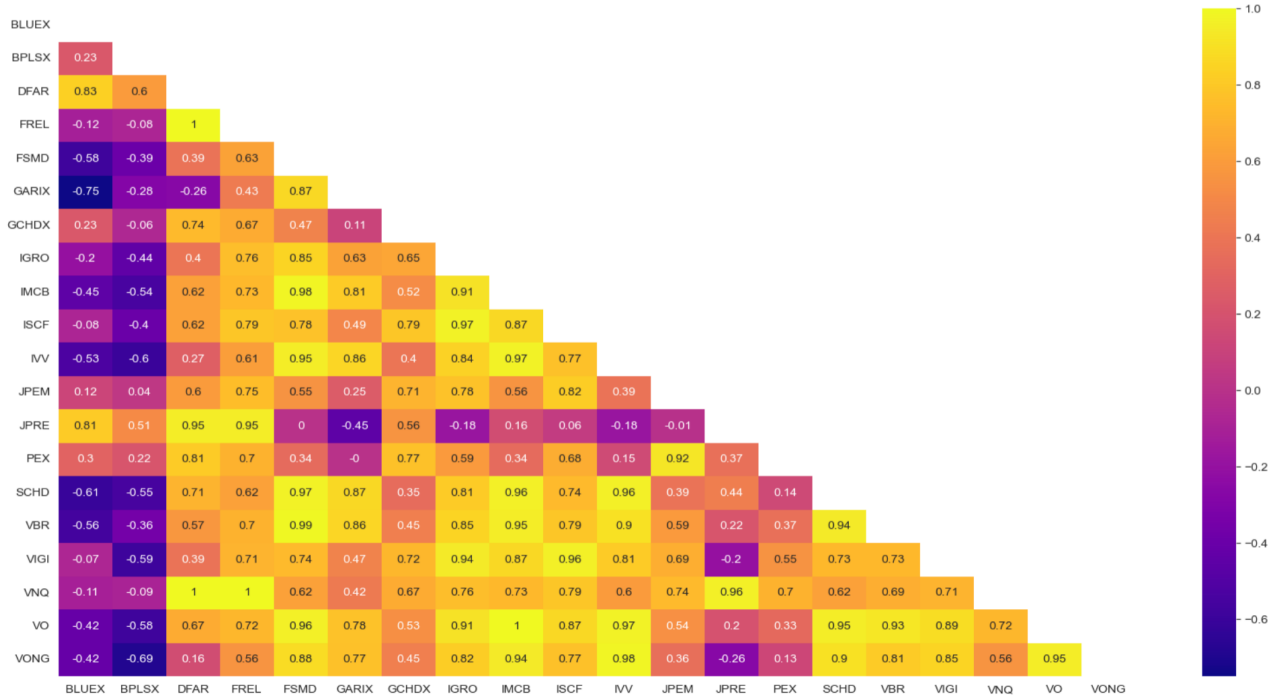
Economically Sensitive

Load Pricing Time Series

[68]:
```
# ticker symbols
ticker_lst = list(folio_econ_df['Ticker'])
print('ticker #:', len(ticker_lst))


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

#Download closing prices
#price_df = pdr.get_data_yahoo(ticker_lst, start=start_date, end=end_date)['Close']
price_econ_df = pdr.get_data_yahoo(ticker_lst, period=period)['Close']

#From the closing prices, calculate periodic returns
return_econ_df = price_econ_df.pct_change()
return_econ_df = return_econ_df[1:]
print('return_df #:', return_econ_df.shape)
```

```
ticker #: 20
price #: 20
[**********************100%%**********************]  20 of 20 completed
return_df #: (1257, 20)
```

[69]:
```
plt.figure(figsize=(20,10))
mask = np.zeros_like(price_econ_df.corr())
mask[np.triu_indices_from(mask)] = True
sns.set_style("white")
_p = sns.heatmap(price_econ_df.corr().round(2),
                 annot=True, mask=mask,
                 cmap="plasma", annot_kws={"size": 10})
```



Portfolio Returns

[70]:
```
port_ret_weighted= return_econ_df.mul(weight_lst, axis=1)
print('port_ret_weighted #:', port_ret_weighted.shape)

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

```
port_ret_weighted #: (1257, 20)
return_econ_df #: (1257, 21)
```

[71]:
```
tickers='^DJA'
#tickers = 'FHNFX'
DJA = pdr.get_data_yahoo(tickers=tickers, period=period)
return_econ_df['Benchmark']= DJA['Close']
return_econ_df['Benchmark']= return_econ_df['Benchmark'].pct_change(1) #.dropna()
#return_df= return_df.dropna()
print('return_econ_df #:', return_econ_df.shape)
```

```
[**********************100%%**********************]  1 of 1 completed
return_econ_df #: (1257, 22)
```

[72]:
```
cum_ret_port = ((1 + return_econ_df[['Portfolio','Benchmark']]).cumprod()-1)
cum_ret_port.plot(label='Cumulative Returns of the Portfolio' ,figsize=(19,8),title='Cumulative Returns - Economically Sensitive (vs ^DJA)')
_=plt.xlabel('Date' )
_=plt.ylabel('Portfolio Cumulative Return' )
```

[73]:
```
port_bench = pd.concat([return_econ_df['Portfolio' ],return_econ_df['Benchmark' ]],axis=1 ).dropna()
port_bench.columns=['Portfolio',' Benchmark' ]
correlation=port_bench.corr()
correlation
```

[73]:

	Portfolio	Benchmark
Portfolio	1.000000	0.947144
Benchmark	0.947144	1.000000

Sharpe Ratio

[74]:
```
return_econ_df['RF Rate']= rf_rate

sharpe_ratio=((return_econ_df['Portfolio'].mean() - return_econ_df['RF Rate'].mean()))/return_econ_df['Portfolio' ].std()
sh = round(sharpe_ratio,4)
print(f"Sharpe Ratio: {sh:.7f} (50% {sh*0.5:.7f})")

avg = return_econ_df['Portfolio'].mean() * 100
print(f"Portfolio Return Mean: {avg:.7f}")

std = return_econ_df['Portfolio' ].std()
print(f"Portfolio Return Standard Deviation: {std:.7f}")
```

```
Sharpe Ratio: 2.3514000 (50% 1.1757000)
Portfolio Return Mean: 0.0090987
Portfolio Return Standard Deviation: 0.0070701
```

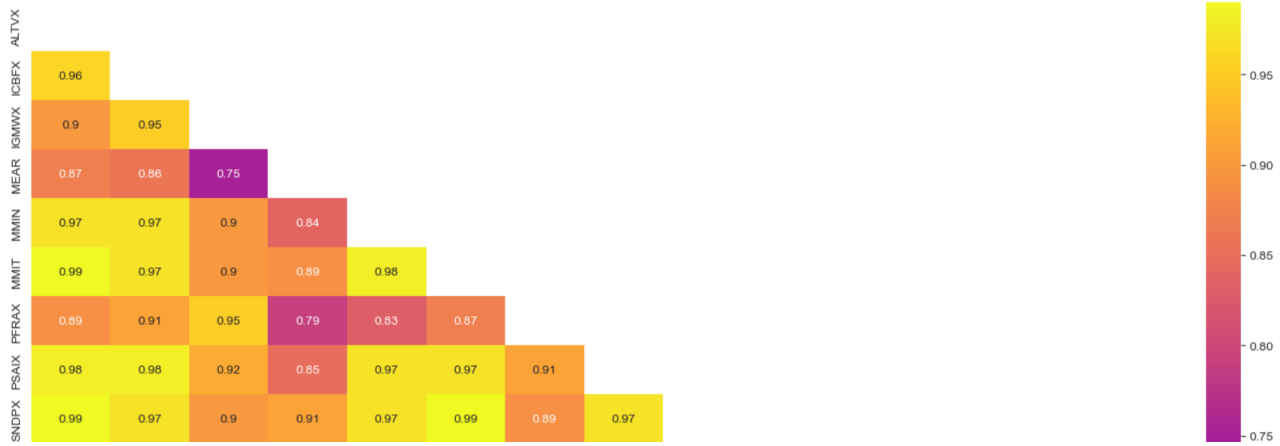
Interest Rate Sensitive

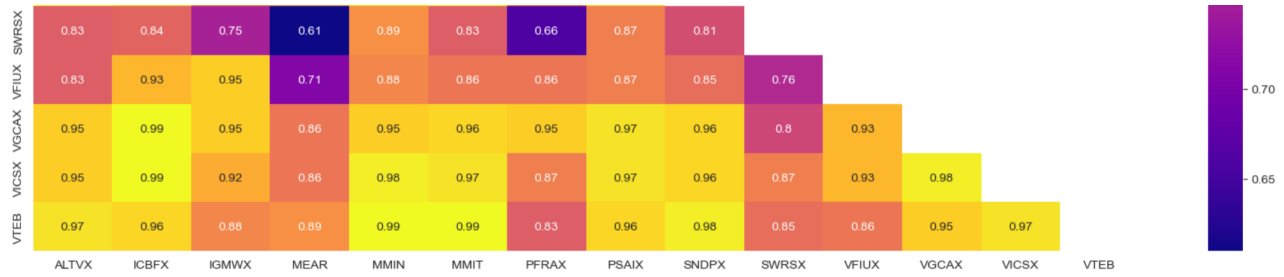
Load Pricing Time Series

[75]:
```
# ticker symbols
ticker_lst = list(folio_int_df['Ticker'])
print('ticker #:', len(ticker_lst))


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


#Download closing prices
#price_df = pdr.get_data_yahoo(ticker_lst, start=start_date, end=end_date)['Close']
price_int_df = pdr.get_data_yahoo(ticker_lst, period=period)['Close']

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

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

```
ticker #: 14
price #: 14
[**********100%**********]  14 of 14 completed
return_df #: 14
```

[76]:
```
plt.figure(figsize=(20,10))
mask = np.zeros_like(price_int_df.corr())
mask[np.triu_indices_from(mask)] = True
sns.set_style("white")
_p = sns.heatmap(price_int_df.corr().round(2),
                 annot=True, mask=mask,
                 cmap="plasma", annot_kws={"size": 10})
```





Portfolio Returns

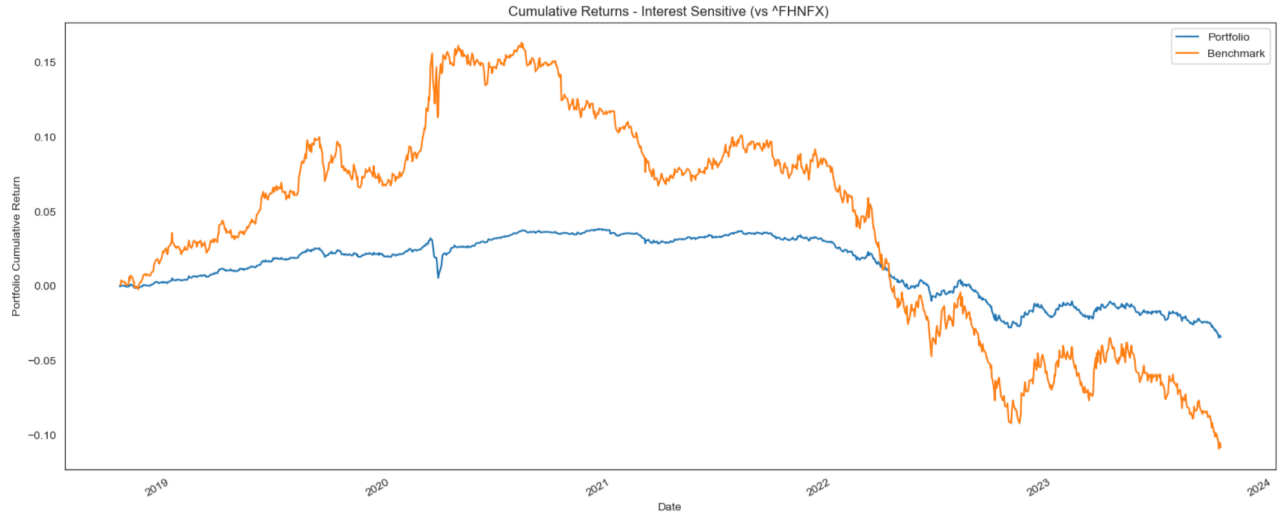
[77]:
```
port_ret_weighted= return_int_df.mul(weight_lst, axis=1)
print('port_ret_weighted #:', len(port_ret_weighted))

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

```
port_ret_weighted #: 1258
return_int_df #: (15,)
```

[78]:
```
#tickers='^SP500BDT'
tickers = 'FHNFX'
BOND = pdr.get_data_yahoo(tickers=tickers, period=period)
return_int_df['Benchmark']= BOND['Close']
return_int_df['Benchmark']= return_int_df['Benchmark'].pct_change(1)

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

```
[**********100%**********]  1 of 1 completed
return_int_df #: (1258, 16)
```

[79]:
```
cum_ret_port = ((1 + return_int_df[['Portfolio','Benchmark']]).cumprod()-1)
cum_ret_port.plot(label='Cumulative Returns of the Portfolio' ,figsize=(19,8),title='Cumulative Returns - Interest Sensitive (vs ^FHNFX) ')
_=plt.xlabel('Date' )
_=plt.ylabel('Portfolio Cumulative Return' )
```



[80]:
```
port_bench = pd.concat([return_int_df['Portfolio' ],return_int_df['Benchmark' ]],axis=1 ).dropna()
port_bench.columns=['Portfolio',' Benchmark' ]
correlation=port_bench.corr()
correlation
```

[80]:

	Portfolio	Benchmark
Portfolio	1.000000	0.840301
Benchmark	0.840301	1.000000

Sharpe Ratio

[81]:
```
return_int_df['RF Rate']= rf_rate

sharpe_ratio=((return_int_df['Portfolio'].mean() - return_int_df['RF Rate'].mean()))/return_int_df['Portfolio' ].std()
sh = round(sharpe_ratio,4)
print(f"Sharpe Ratio: {sh:.7f} (50% {sh*0.5:.7f})")

avg = return_int_df['Portfolio'].mean() * 100
print(f"Portfolio Return Mean: {avg:.7f}")

std = return_int_df['Portfolio' ].std()
print(f"Portfolio Return Standard Deviation: {std:.7f}")
```

```
Sharpe Ratio: 18.2277000 (50% 9.1138500)
Portfolio Return Mean: -0.0027182
Portfolio Return Standard Deviation: 0.0009056
```

[ ]:


Simple
0
13
Python 3 (ipykernel) | Idle
Mode: Edit
Ln 1, Col 1
DATA618_W6_Sharpe.ipynb
1