Mojo Partners LLC

An FI351 Project Creation

Jared Reimer

Professor Cicchetti

FI351

Our Mission Statement:

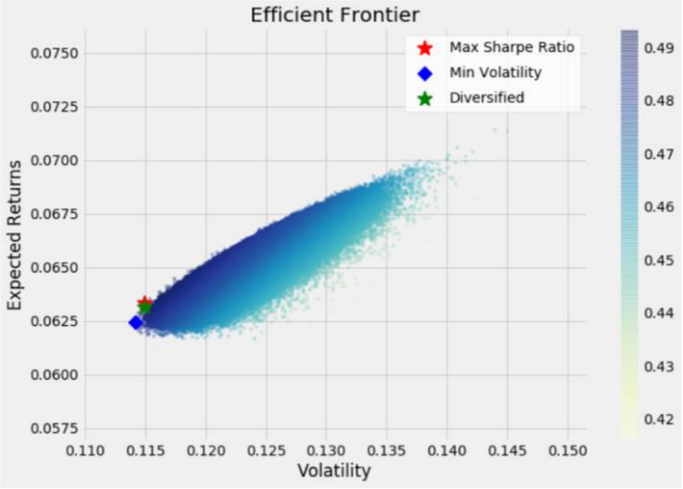
Here are Mojo Partners LLC, our funds mission is to help the investor first. It is our jobs to be up to speed and ahead of the curve with any and all financial news around the clock for the sake of your investment. We accomplish this by investing in a diverse pool of foreign assets, aiming to capitalize on shifts in currency valuations and global monetary trends. This means picking out the best stocks in the best markets throughout the world, as only the best will do for our investors. To pick these stocks we stay up to date with all international news that may affects foreign markets and combine this with different logic-based algorithms to most optimally diversify our holdings. We hope that the following report showcases the dedication we have towards preserving and bettering your financial future

*Values shown at end are not exact to FTS positions as they are from yahoo finance's closing prices, and FTS purchases were made midday Friday. As expected returns were retrieved from FTS on Thursday night, the numbers expressed are still relevant for the analysis of portfolio performance.

Jared Reimer- Fund Manager

Currently a student studying Data Analytics and Finance at Bentley University, nobody knows how he does it. Referred to as "The Kid" in various articles by Wall Street Journal, nobody understands how he is able to balance school, athletics (being a member of the Bentley Track and Field team) as well as his premier cutting-edge hedge fund. Many wonder if there is anything he can't do. And if as you read this report, you find yourself asking this question yourself, allow me to give you the answer: the ability to either stay humble or take himself seriously are far beyond his personal capabilities, as showcased in this immaculate bio.

Efficient Frontier of Stock Combinations:



Diversified Portfolio Balance:

Returns	Volatility	PGJ Weight	EZA Weight	GXC Weight	FXI Weight	EIDO Weight	IDX Weight	EWY Weight	VNM Weight	EWT Weight	TUR Weight	EIS Weight	EWD Weight	EWG Weight	ILF Weight	EWZ Weight	Sharpe
6.36%	11.60%	3.00%	4.00%	7.50%	7.00%	4.50%	4.50%	12.00%	13.50%	3.00%	0.00%	17.00%	7.50%	9.00%	3.50%	4.00%	0.492

Minimum Volatility Portfolio Balance:

Returns	Volatility	PGJ Weight	EZA Weight	GXC Weight	FXI Weight	EIDO Weight	IDX Weight	EWY Weight	VNM Weight	EWT Weight	TUR Weight	EIS Weight	EWD Weight	EWG Weight	ILF Weight	EWZ Weight	Sharpe
6.24%	11.42%	0.13%	0.76%	3.61%	10.08%	1.33%	5.72%	7.92%	8.56%	12.49%	3.82%	12.84%	9.98%	10.31%	9.40%	3.03%	0.489

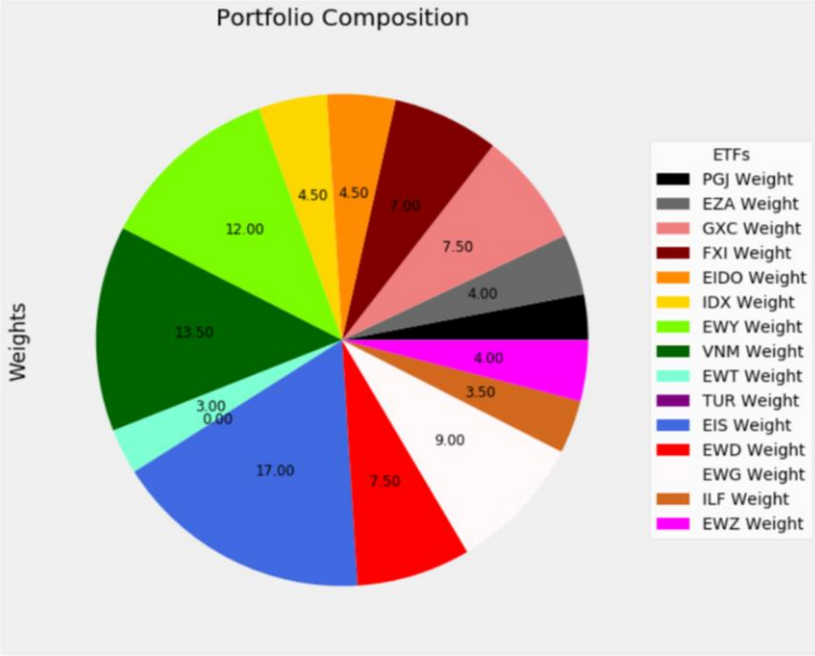
Maximum Sharpe Ratio Portfolio Balance:

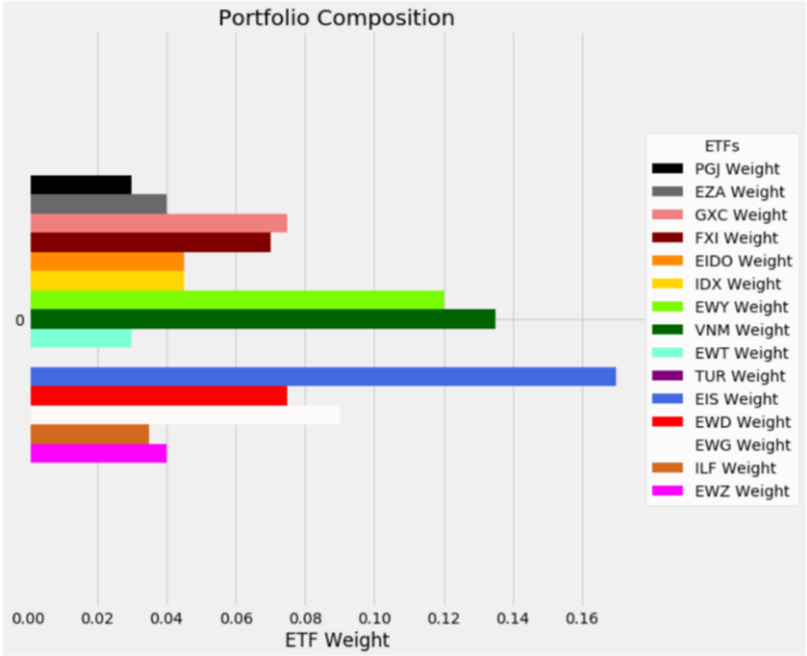
Returns	Volatility	PGJ Weight	EZA Weight	GXC Weight	FXI Weight	EIDO Weight	IDX Weight	EWY Weight	VNM Weight	EWT Weight	TUR Weight	EIS Weight	EWD Weight	EWG Weight	ILF Weight	EWZ Weight	Sharpe
6.33%	11.50%	2.87%	1.28%	0.81%	12.17%	3.60%	3.62%	11.04%	13.85%	18.43%	2.45%	17.90%	1.42%	4.98%	2.82%	2.77%	0.494

The Strategy:

In order to find the optimal portfolio, we first narrowed down the usable foreign ETFs to the best third by return, as there is a large portion of them have an extremely underperforming expected return. From here, we randomly generated 50,000 portfolios to create an efficient frontier of portfolio compositions. Points highlighted on this frontier are the portfolio with the minimum variance, the maximum Sharpe ratio, and a "diversified" portfolio. This diversified portfolio was created by looking amongst the top 10 portfolio options by Sharpe ratio, and calculating which portfolio had the most even distribution of weighting among stocks (least variance in weighting among all chosen ETFs). Overall, we decided that the diversified portfolio best fit our interests, as in times as volatile as now with the ongoing pandemic, it is wise to distribute risk as evenly as possible in case disaster hits the market of one of our ETFs. The balance of this chosen portfolio is shown on the next page, expressed as both a pie chart and bar chart to clearly show the composition.

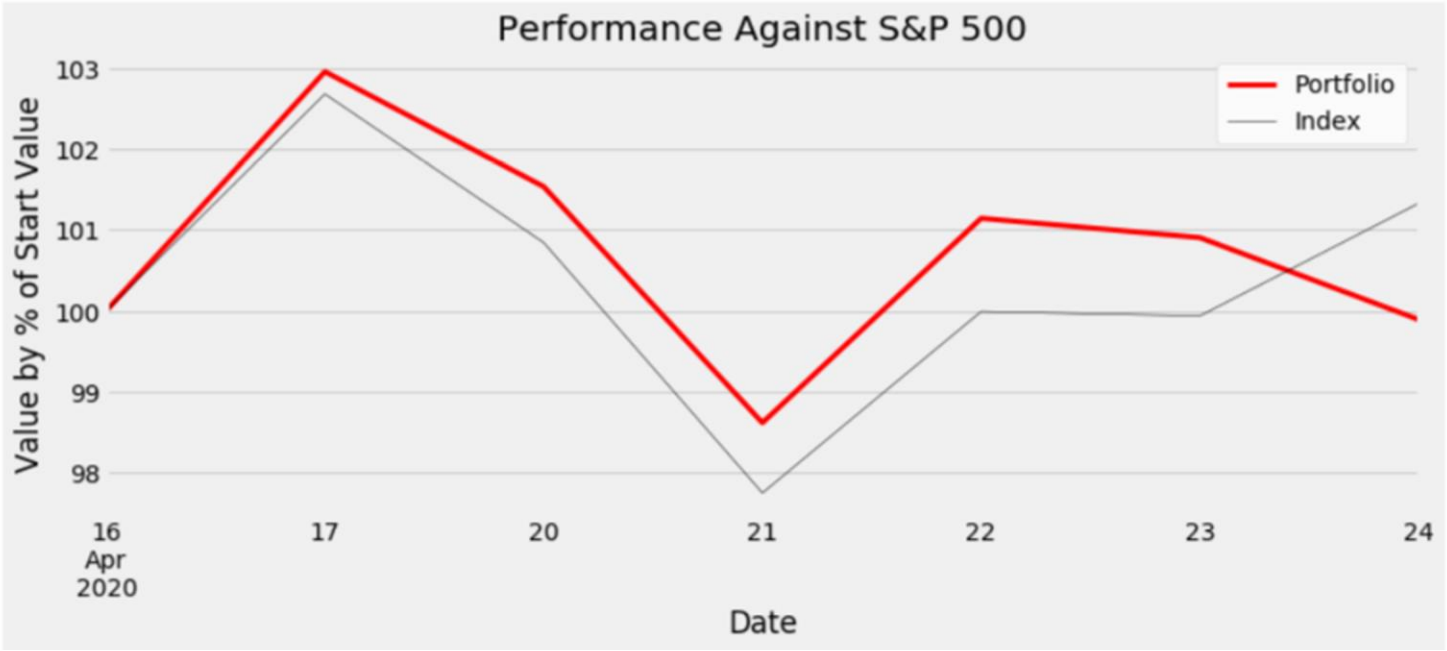
SELECTED COMPOSITION





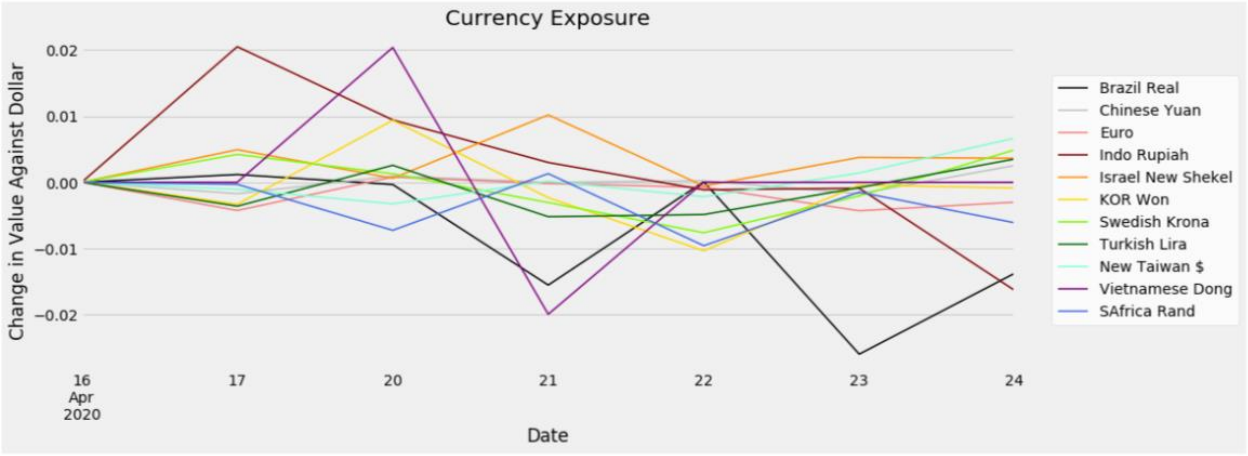
```
Starting portfolio value is: $1,003,000.00
Current portfolio value is: $1,001,916.69

Portfolio return is: -0.11%
Index return is: 1.33%
```

Portfolio Performance



As you can see in the graphic above, our portfolio's performance was extremely volatile and ended up slightly underperforming the S&P 500. This can be attributed to two things. First, this is only a week of performance, and is not a great representation of how the portfolio will perform in the long run, only very short-term economic conditions. Second, are the previously mentioned short-term economic conditions. Our portfolio is made up of various foreign ETFs, all in markets affected to varying degrees by the coronavirus pandemic. This condition is greatly emphasized by the effect that the US economy's 2 trillion dollar stimulus has had on foreign exchange rates, with many of the currencies involved in our stocks devaluing against the dollar.

As previously mentioned, our portfolio's performance was greatly hindered by the appreciation of the dollar against many foreign currencies. Seen below, we have a graph depicting the valuation of the currencies involved in the domestic markets of our ETFs against the dollar. This graphic clearly shows that the Brazilian Real has declined over 5% against the dollar, which is greatly reflective of our portfolio's performance, with the Brazil Cap ETF being the biggest loser for our portfolio. The rest of the stocks chosen and their respective currency's valuation against the dollar showed similar correlations. Overall, this shows that the current shortcomings of our portfolio are not directly to be attributed to the underperformance of its chosen ETFs, but the current state of the market and its relation to the massive US federal stimulus package. As time passes, we at Mojo Partners LLC are sure that the markets will balance out and this portfolio will not only meet, but exceed expectations.



Cumulative Growth/Decline Against Dollar

Date	BRLUSD=X	CNYUSD=X	EURUSD=X	IDRUSD=X	ILSUSD=X	KRWUSD=X	SEKUSD=X	TRYUSD=X	TWDUSD=X	VNDUSD=X	ZARUSD=X
2020-04-24 00:00:00	-5.35%	0.02%	-1.17%	1.42%	2.27%	-0.80%	-0.25%	-0.85%	0.17%	0.00%	-2.33%

Code Used:

```
import yfinance as yf
import datetime
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
import random
import warnings
warnings.filterwarnings('ignore')

# ## Get returns and covariance matrix from excel file
# ### Clean data to include relevant ETFs only

data = pd.read_excel(r'~Jared Reimer\Documents\FTS_ETF_Cut.xlsx')
clean_data = data.nlargest(int(len(data['Expected Return'])/3), 'Expected Return')
select_etfs = ['Ticker', 'Name', 'Expected Return']
selected = clean_data['Name'].to_list()
select_etfs.extend(selected)
clean_data = clean_data[select_etfs]

ticker_list = clean_data['Ticker'].to_list()
returns = clean_data['Expected Return']

clean_data

# ### Separate matrix and set index to properly use
cov_matrix = clean_data.drop(columns=['Ticker', 'Expected Return'])
cov_matrix= cov_matrix.set_index('Name')
cov_matrix


# ## Make random portfolio creation process
p_ret=[]
p_vol=[]
p_weights = []

num_assets = len(ticker_list)
num_portfolios = 500000

for single_portfolio in range(num_portfolios):
    weights = np.random.random(num_assets)
    weights /= np.sum(weights)
    rets = np.dot(weights, returns)
    volatility = np.sqrt(np.dot(weights.T, np.dot(cov_matrix, weights)))
    p_ret.append(rets)
    p_vol.append(volatility)
    p_weights.append(weights)
```

```
# ### Put resulting portfolios and their performance in dataframe

data1 = {'Returns': p_ret, 'Volatility': p_vol}
for counter, symbol in enumerate(ticker_list):
   data1[symbol+ ' Weight'] = [w[counter] for w in p_weights]
portfolios = pd.DataFrame(data1)


# ### Get t-bill data to calculate risk free rate for Sharpe Ratio calculations
rf_data = yf.download('^TNX', '2020-04-01', datetime.datetime.now())
#print(rf_data)
rf_returns = rf_data['Adj Close']
rfr = (rf_returns.mean())/100 # divide by 100 to convert to x percent
portfolios['Sharpe']=(portfolios['Returns']-rfr)/portfolios['Volatility']
portfolios.head(3)


# ### Find portfolio within top 10 Sharpe ratios with most diversification
best_weights = portfolios.nlargest(10, 'Sharpe')
best_weights['P Weight Var'] = best_weights.iloc[:,2:-1].var(axis=1)
diversified = (best_weights[best_weights['P Weight Var']==best_weights['P Weight Var'].min()])
diversified = diversified.iloc[:,:-1]


# ### Also find portfolios with optimal Sharpe ratio and minimum variance for plotting purposes
sharpe = (portfolios[portfolios['Sharpe']==portfolios['Sharpe'].max()])
min_vol = (portfolios[portfolios['Volatility']==portfolios['Volatility'].min()])
sharpe


# ## Plot Efficient Frontier
plt.style.use('fivethirtyeight')
plt.figure(figsize=(10,7))
plt.scatter(x=portfolios['Volatility'], y=portfolios['Returns'], c= portfolios['Sharpe'], cmap='YlGnBu', marker='o', s=10, alpha=.3)
plt.colorbar()
plt.scatter(x=sharpe['Volatility'], y=sharpe['Returns'], color= 'r', marker='*', s=200, lw=2., label='Max Sharpe Ratio')
plt.scatter(x=min_vol['Volatility'], y=min_vol['Returns'], color= 'b', marker='D', s=100, lw=1., label='Min Volatility')
plt.scatter(x=diversified['Volatility'], y=diversified['Returns'], color= 'g', marker='*', s=200, lw=2., label='Diversified')
plt.xlabel('Volatility')
plt.ylabel('Expected Returns')
plt.title('Efficient Frontier')
plt.legend(labelspacing=.8, facecolor='white')

plt.show()

# ### Create Style dictionary to more clearly show portfolio weightings and performance
div_cols = diversified.columns.to_list()
values = []
for x in range(0, (len(div_cols)-1)):
```

```
    values.append('{:.2%}')
values.append('{:.3}')

port_format = dict(zip(div_cols, values))

print('Diversified Portfolio Balance:')
# ADD ROW OF ACTUAL BC FORGOT TO SET SEED ON INITIAL RUN
actual_row = [[.0636, .116, .03, .04, .075, .07, .045, .045, .12, .135, .03, 0, .17, .075, .09, .035, .04, .492]]
temp = pd.DataFrame(actual_row, columns=diversified.columns.to_list())
diversified = pd.concat([temp, diversified], ignore_index=True)
diversified = diversified.iloc[0:1,:]
(diversified.style.format(port_format))

transposed = diversified.iloc[:,2:-1].T
transposed['Weights'] = transposed

# Plot Weights
ax1 = transposed.plot.pie(y='Weights', autopct='%.2f', fontsize=12, labels=None, figsize=(9, 9), colors=['k', 'dimgrey', 'lightcoral', 'maroon',
'darkorange',
                          'gold', 'lawngreen', 'darkgreen', 'aquamarine', 'purple', 'royalblue',
                          'red', 'snow', 'chocolate', 'magenta' ])
plt.title('Portfolio Composition')
plt.legend(title='ETFs', bbox_to_anchor=(1.0,0.5), loc="center left", borderaxespad=0, facecolor='white', labels=diversified.iloc[:,2:-
1].columns.to_list())
ax2 = diversified.iloc[:,2:-1].plot.barh(figsize=(9,9), color=['k', 'dimgrey', 'lightcoral', 'maroon', 'darkorange',
                          'gold', 'lawngreen', 'darkgreen', 'aquamarine', 'purple', 'royalblue',
                          'red', 'snow', 'chocolate', 'magenta' ])
plt.legend(title='ETFs', bbox_to_anchor=(1.0,0.5), loc="center left", borderaxespad=0, facecolor='white', labels=diversified.iloc[:,2:-
1].columns.to_list())
plt.title('Portfolio Composition')
plt.gca().invert_yaxis()
plt.ylabel('')
plt.xlabel('ETF Weight')
plt.show()

print('\nMinimum Volatility Portfolio Balance:')
(min_vol.style.format(port_format))

print('\nMaximum Sharpe Ratio Portfolio Balance:')
sharpe.style.format(port_format)


# # Track Performance against the S&P 500

# ### Get data from week of holding
start_date = '2020-04-16'
end_date = '2020-04-25'
perf_data = yf.download(ticker_list, start_date, end_date)
perf_prices = perf_data['Adj Close']
perf_prices
```

```
weighting = diversified.iloc[:,2:-1]
opt_weights = weighting.values
opt_weights = opt_weights[0]
opt_weights

np.sum(opt_weights)

# ### Establish dataframe to initialize positions according to decided weighting

initial_capital = 1003000.0

positions = pd.DataFrame(index=perf_prices.index).fillna(0.0)
for counter, symbol in enumerate(perf_prices.columns.tolist()):
   positions[symbol + ' Shares'] = ((opt_weights[counter]*initial_capital)/perf_prices.iloc[0,counter])

# ### Create portfolio dataframe to update value of positions according to changes in value
# Create list of columns for portfolio
col_list= []
for counter, symbol in enumerate(perf_prices.columns.tolist()):
   col_list.append(symbol + ' Value')
# Initialize portfolio
portfolio = pd.DataFrame(perf_prices.values*positions.values, columns=col_list, index=perf_prices.index)
portfolio['total'] = portfolio.sum(axis=1)
portfolio['returns'] = portfolio['total'].pct_change()

# ### Get index data for comparison
index_data = yf.download('^GSPC', start_date, end_date)
index_data['returns'] = index_data['Adj Close'].pct_change()

# ### Standardize values of index and portfolio
portfolio['Std Value'] = portfolio['total'] / portfolio['total'].iloc[0] * 100
index_data['Std Value'] = index_data['Adj Close'] / index_data['Adj Close'].iloc[0] * 100

# ## Plot Performance
fig = plt.figure(figsize=(12,10))
ax1 = fig.add_subplot(211, ylabel='Value by % of Start Value')
portfolio['Std Value'].plot(ax=ax1, color='red', lw=2.)
index_data['Std Value'].plot(ax=ax1, color='black', lw=1.)
plt.legend(labels=['Portfolio', 'Index'], facecolor='white')
plt.title('Performance Against S&P 500')
# Get final value of portfolio
portfolio_value = (portfolio.iloc[-1,-3])
# Calc cumulative returns of port and index
cum_port_change = (1+portfolio['returns']).cumprod()
cum_index_change = (1+index_data['returns']).cumprod()
# Annualize
portfolio_cum_ret = (cum_port_change)-1 #**(252/len(portfolio.index)))-1
index_cum_ret = (cum_index_change)-1 #**(252/len(index_data.index)))-1
#print(portfolio_cum_ret)
```

```
# Slice most recent date
portfolio_ret= portfolio_cum_ret[-1]
index_ret = index_cum_ret[-1]

print('\nStarting portfolio value is: $'+ str('{:,.2f}'.format(portfolio.iloc[0,-3])))
print('Current portfolio value is: $'+ str('{:,.2f}'.format(portfolio_value)))
print('\nPortfolio return is: '+ str("{:.2%}".format(portfolio_ret)))
print('Index return is: '+ str("{:.2%}".format(index_ret)))
plt.show()

# # Check movement of currencies involved in ETFs against USD
currency_list = ['CNYUSD=X', 'ZARUSD=X', 'IDRUSD=X', 'KRWUSD=X', 'VNDUSD=X', 'TWDUSD=X', 'TRYUSD=X', 'ILSUSD=X', 'SEKUSD=X',
'EURUSD=X', 'BRLUSD=X']
currency_data =  yf.download(currency_list, start_date, end_date)
currency_prices = currency_data['Adj Close']
change_currency_prices = currency_prices.pct_change()
change_currency_prices = change_currency_prices.fillna(0)
change_currency_prices

cum_changes = ((1+change_currency_prices).cumprod())-1
#print(cum_changes)
col_names = ['Brazil Real', 'Chinese Yuan', 'Euro', 'Indo Rupiah', 'Israel New Shekel', 'KOR Won', 'Swedish Krona', 'Turkish Lira', 'New Taiwan $',
'Vietnamese Dong', 'SAfrica Rand']
#cum_changes.columns = col_names
#print(cum_changes)

# Plot
fig2 = plt.figure(figsize=(14,12))
ax2 = fig2.add_subplot(211, ylabel='Change in Value Against Dollar')
change_currency_prices.plot(ax=ax2, lw=3/2., color=['k', 'silver', 'lightcoral', 'maroon', 'darkorange',
                                'gold', 'lawngreen', 'darkgreen', 'aquamarine', 'purple', 'royalblue'])
plt.legend(bbox_to_anchor=(1.04,0.5), loc="center left", borderaxespad=0, facecolor='white', labels=col_names)
plt.xlabel('Date')
plt.ylabel('Change in Value Against Dollar')
plt.title('Currency Exposure')
plt.show()
print('\nCumulative Growth/Decline Against Dollar')
#print(cum_changes.iloc[-1:,:])
currency_format = dict(zip(cum_changes.columns, values))
(cum_changes.iloc[-1:,:].style.format(currency_format))
```