Import Packages

In [3]:
```
import matplotlib.pyplot as plt
import numpy as np
import datetime
from datetime import timedelta
from datetime import datetime
import yfinance as yf
import statsmodels.api as sm
import seaborn as sns
import statsmodels.tsa.stattools as ts
import statsmodels.formula.api as smf
import pandas as pd
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
pd.reset_option('all')
import statsmodels as sm
```

Make functions

Import Data

In [87]:
```
def get_data(ticker1, ticker2):
    df = yf.download( tickers = ticker1 + " " + ticker2,
                      period = "5y",
                      interval = "1d")[["Adj Close"]]
    df.columns = [ticker1, ticker2]
    return(np.cumprod(df.dropna().pct_change().fillna(0.)+1.))
```

In [88]:
```
ticker1 = "^GSPC"
ticker2 = "^IXIC"
data = get_data(ticker1, ticker2)
```

```
[**********100%**********]  2 of 2 completed
```

In [ ]:
```

```

Calculate Cointegration

In [89]:
```
data.head()
```

Out[89]:

Date	^GSPC	^IXIC
2018-08-27	1.000000	1.000000
2018-08-28	1.000269	1.001514
2018-08-29	1.005972	1.011448
2018-08-30	1.001515	1.008788
2018-08-31	1.001650	1.011429

In [ ]:
```

```

Pair calculation

In [ ]:
```

```

In [90]:
```
#sm.regression.linear_model.OLS?
```

Build Indices

In [91]:
```
Z = 1.
```

In [92]:
```
data["Hedge"] = 0.
data["Spread"] = 0.
data["Upper"] = 0.
data["Lower"] = 0.
data["Mean"] = 0.
data["Signals"] = 0.

#calculate ourt hedge ratio
model=sm.regression.linear_model.OLS(data[ticker1], data[ticker2])
model = model.fit()
hedge = model.params[0]
data["Hedge"] = hedge

data["Spread"] = data[ticker1] - (data[ticker2] * hedge)

data["Mean"] = (data["Spread"]).mean()

data["SD"] = (data["Spread"]).std()

#data["Mean"] = (data["Spread"]).rolling(400).mean()
#data["SD"] = (data["Spread"]).rolling(400).std()


data["Upper"] = data["Mean"] + (Z * data["SD"])
data["Lower"] = data["Mean"] - (Z * data["SD"])
```

In [93]:
```
# Long if above bounds and short if below bounds
def strategy_signals_1(data):
    data["Signals"] = 0.
    data.loc[data["Upper"] < data["Spread"], "Signals"] = -1.
    data.loc[data["Lower"] > data["Spread"], "Signals"]= 1.
    return(data)
```

In [94]:
```
# ENTRY : Long if crossing above bounds and short if crossing below bounds
# EXIT : Crossing the mean line
def strategy_signals_2(data):
    data["Signals"] = 0.
    for i in range(1, len(data)):

        # Short Condition
        if data["Upper"][i] < data["Spread"][i]:
            data["Signals"][i] = -1.
        if (data["Signals"][i-1] == -1.) & (data["Spread"][i] > data["Mean"][i]):
            data["Signals"][i] = -1.

        # Long Condition

        if data["Lower"][i] > data["Spread"][i]:
            data["Signals"][i] = 1.
        if (data["Signals"][i-1] == 1.) & (data["Spread"][i] < data["Mean"][i]):
            data["Signals"][i] = 1.

    return(data)
```

In [95]:
```
data = strategy_signals_2(data)
```

In [96]:
```
# Get Strategy Profits
data["asset1"] = data[ticker1].pct_change()
data["asset2"] = data[ticker2].pct_change()

data["Return"] = (data["asset1"] + (data["Hedge"] * data["asset2"])) * data["Signals"]
data["cumulReturn"] = np.cumprod((data["Return"]).fillna(0)+1)
```

In [ ]:
```

```

Plot

In [97]:
```
longs = data["Spread"][data["Signals"] == 1.]
shorts = data["Spread"][data["Signals"] == -1.]
```

In [98]:
```
# Two Series
plt.figure(figsize = (10,6))
plt.plot(data[ticker1])
plt.plot(data[ticker2])
plt.legend([ticker1,ticker2])
plt.grid()
plt.show()

# Spread
plt.figure(figsize = (10,6))
plt.plot(data["Spread"], c = "purple")
plt.plot(data["Mean"], linestyle = "dashed", c = "black")
plt.plot(data["Upper"], linestyle = "dotted", c = "pink")
plt.plot(data["Lower"], linestyle = "dotted", c = "pink")

plt.scatter(longs.index, longs, c = "green" , s = 10 )
plt.scatter(shorts.index, shorts, c = "red" , s = 10 )

plt.legend(["Spread", "Mean", "Bounds"])
plt.grid()
plt.show()

# Plot the profits
plt.figure(figsize = (10,6))
plt.plot(data[ticker1])
plt.plot(data[ticker2])
plt.plot(data["cumulReturn"])
plt.legend([ticker1,ticker2, "Strategy Return"])
plt.grid()
plt.show()
```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```