Studies on The Impact of Economy Recession on stock prices

Elizabeth Peiwen Li

Project Goal:

Part 1 Extract Data from public sources: Yahoo Finance.

We chose 4 indexes, which are: Standard Poor 500, Dow Jones' Index, 10 Years Treasury Note Yield Index and Nasdaq Composite. And we chose 4 representative public companies in the U.S. , and they are: GOOG(Alphabet: Google);AIG (American International Group); XOM(Exxon Mobil Corporation); UAL(United Airline). Then we chose four counterpart Chinese companies which went public in the U.S., and they are: PTR(petrochina company limited); BIDU(baidu); CEA(china eastern airlines); LFC (China life insurance company limited)

Part 2: Study the impact of economy recession to the prices of the chosen indices and stock prices.

1. Study the impact of economy recession on the price of chosen stock market indices and chosen both U.S. public companies and Chinese companies listed in the U.S. for the past 40 years.
2. Study the impact of economy recession on the price change percentage of chosen stock market indices and chosen both U.S. public companies and Chinese companies listed in the U.S. for the past 40 years.

Note: The economy resession happened in the past 40 years are: Jan. 1980 - Jul. 1980; Jul. 1981 - Nov. 1982; Jul.1990 - Mar.1991; Mar. 2001 - Nov. 2001; Dec. 2007 - Jun. 2009.

Part 3: Choose one of the stock market index to conduct descriptive analysis and predictive analysis.

1. Decribe the dataset;
2. Study the distribution of the dataset;
3. Conduct the decriptive analysis.
4. Conduct the predictive analysis.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
# from yahoo_finance import Share
import yfinance as yf
# from yahoofinancials import YahooFinancials
from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()
from bokeh.io import output_notebook
from bokeh.plotting import figure, show
from bokeh.models import Title
from bokeh.models import BoxAnnotation
from bokeh.models import Legend

output_notebook()
%matplotlib notebook
```

(https://bokeh.pydata.org) BokehJS 1.0.4 successfully loaded.

Part 1: Data Extraction and Cleanning

In [2]:

```python
# download historical monthly stock price data of the 4 indices and 8 companies  from Yahoo Finance

index_data = ["^GSPC", "^DJI", "^TNX", "^IXIC"]
usa_inc_data = ["GOOG", "AIG", "XOM", "UAL"]
chn_inc_data = ["PTR", "BIDU", "CEA", "LFC"]


start = datetime(1979,1,1)
end = datetime(2019,1,1)

df_index_SPY = yf.download(index_data[0], start = start, end = end, interval = "1mo")
df_index_Dow = yf.download(index_data[1], start = start, end = end, interval = "1mo")
df_index_TNX = yf.download(index_data[2], start = start, end = end, interval = "1mo")
df_index_IXIC = yf.download(index_data[3], start = start, end = end, interval = "1mo")


df_usa_inc_GOOG = yf.download(usa_inc_data[0], start = start, end = end, interval = "1mo")
df_usa_inc_AIG = yf.download(usa_inc_data[1], start = start, end = end, interval = "1mo")
df_usa_inc_XOM = yf.download(usa_inc_data[2], start = start, end = end, interval = "1mo")
df_usa_inc_UAL = yf.download(usa_inc_data[3], start = start, end = end, interval = "1mo")


df_chn_inc_PTR = yf.download(chn_inc_data[0], start = start, end = end, interval = "1mo")
df_chn_inc_BIDU = yf.download(chn_inc_data[1], start = start, end = end, interval = "1mo")
df_chn_inc_CEA = yf.download(chn_inc_data[2], start = start, end = end, interval = "1mo")
df_chn_inc_LFC = yf.download(chn_inc_data[3], start = start, end = end, interval = "1mo")
```

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

In [3]:
```
# Let's see what the dataset downloaded might look like
df_index_SPY.head()
```

Out[3]:

Date	Open	High	Low	Close	Adj Close	Volume
1979-01-01	96.110001	102.589996	95.220001	99.930000	99.930000	615730000
1979-02-01	99.930000	100.519997	95.379997	96.279999	96.279999	475710000
1979-03-01	96.279999	103.309998	95.980003	101.589996	101.589996	649800000
1979-04-01	101.559998	103.949997	100.139999	101.760002	101.760002	620650000
1979-05-01	101.760002	102.570000	97.489998	99.080002	99.080002	623740000

In [4]:
```
df_index_SPY.tail()
```

Out[4]:

Date	Open	High	Low	Close	Adj Close	Volume
2018-08-01	2821.169922	2916.500000	2796.340088	2901.520020	2901.520020	69238220000
2018-09-01	2896.959961	2940.909912	2864.120117	2913.979980	2913.979980	62492080000
2018-10-01	2926.290039	2939.860107	2603.540039	2711.739990	2711.739990	91327930000
2018-11-01	2717.580078	2815.149902	2631.090088	2760.169922	2760.169922	80080110000
2018-12-01	2790.500000	2800.179932	2346.580078	2506.850098	2506.850098	83522570000

In [5]:
```
df_index_SPY.dtypes
```

Out[5]:
```
Open          float64
High          float64
Low           float64
Close         float64
Adj Close     float64
Volume          int64
dtype: object
```

In [6]:
```
df_index_SPY.shape
```

Out[6]:
```
(480, 6)
```

In [7]:
```
df_index_SPY.dtypes
```

Out[7]:
```
Open           float64
High           float64
Low            float64
Close          float64
Adj Close      float64
Volume           int64
dtype: object
```

In [8]:
```
df_index_Dow.head()
```

Out[8]:

Date	Open	High	Low	Close	Adj Close	Volume
1985-01-01	1277.719971	1305.099976	1266.890015	1286.770020	1286.770020	44450000
1985-02-01	1276.939941	1307.530029	1263.910034	1284.010010	1284.010010	207300000
1985-03-01	1285.339966	1309.959961	1242.819946	1266.780029	1266.780029	201050000
1985-04-01	1264.800049	1290.300049	1245.800049	1258.060059	1258.060059	187110000
1985-05-01	1257.180054	1320.790039	1235.530029	1315.410034	1315.410034	242250000

In [9]:
```
df_index_Dow.tail()
```

Out[9]:

Date	Open	High	Low	Close	Adj Close	Volume
2018-08-01	25461.630859	26167.939453	24965.769531	25964.820312	25964.820312	5635410000
2018-09-01	25916.070312	26769.160156	25754.320312	26458.310547	26458.310547	5262500000
2018-10-01	26598.359375	26951.810547	24122.230469	25115.759766	25115.759766	8373350000
2018-11-01	25142.080078	26277.820312	24268.740234	25538.460938	25538.460938	7226940000
2018-12-01	25779.570312	25980.210938	21712.529297	23327.460938	23327.460938	8101540000

In [10]:
```
df_index_Dow.shape
```

Out[10]:
```
(408, 6)
```

In [11]:
```
df_index_Dow.dtypes
```
Out[11]:
```
Open          float64
High          float64
Low           float64
Close         float64
Adj Close     float64
Volume          int64
dtype: object
```

In [12]:
```
df_usa_inc_GOOG.head()
```
Out[12]:

Date	Open	High	Low	Close	Adj Close	Volume
2004-08-01	49.813286	56.528118	47.800831	50.993862	50.993862	134241100.0
2004-09-01	51.158245	67.257904	49.285267	64.558022	64.558022	213503200.0
2004-10-01	65.155777	99.601669	64.209328	94.964050	94.964050	516060900.0
2004-11-01	96.413620	100.423584	80.353813	90.650223	90.650223	557267200.0
2004-12-01	90.635277	99.566803	83.920448	96.035034	96.035034	291772100.0

In [13]:
```
df_usa_inc_GOOG.tail()
```
Out[13]:

Date	Open	High	Low	Close	Adj Close	Volume
2018-08-01	1228.000000	1256.500000	1188.239990	1218.189941	1218.189941	28808400.0
2018-09-01	1204.270020	1212.989990	1146.910034	1193.469971	1193.469971	28862400.0
2018-10-01	1199.890015	1209.959961	995.830017	1076.770020	1076.770020	48494700.0
2018-11-01	1075.800049	1095.569946	996.020020	1094.430054	1094.430054	36735100.0
2018-12-01	1123.140015	1124.650024	970.109985	1035.609985	1035.609985	40257600.0

In [14]:
```
df_usa_inc_GOOG.shape
```
Out[14]:
```
(175, 6)
```

In [15]:
```
df_usa_inc_GOOG.dtypes
```

Out[15]:
```
Open          float64
High          float64
Low           float64
Close         float64
Adj Close     float64
Volume        float64
dtype: object
```

We chose to observe 3 representative datasets(2 index and 1 company), and learnt that, 1) all the dataset downloaded have the same columns: date; open price, highest price, lowest price, close price, adj close price and transaction volume. 2) Although we chose to download the monthly data from Jan. 1 1979 to Jan.1 2019, however, not all the indices or company have the data from Jan. 1 1979. For example, Google went public on August 19 2004, so the stock price data started from Aug. 2004. Therefore, for the analysis purposes, we might need clean or rearrange the dataset later.

In [16]:
```
# Clean the dataset, drop the rows with missing values first.
df_index_SPY = df_index_SPY.dropna()
df_index_Dow = df_index_Dow.dropna()
df_index_TNX = df_index_TNX.dropna()
df_index_IXIC = df_index_IXIC.dropna()


df_usa_inc_GOOG = df_usa_inc_GOOG.dropna()
df_usa_inc_AIG = df_usa_inc_AIG.dropna()
df_usa_inc_XOM = df_usa_inc_XOM.dropna()
df_usa_inc_UAL = df_usa_inc_UAL.dropna()


df_chn_inc_PTR = df_chn_inc_PTR.dropna()
df_chn_inc_BIDU = df_chn_inc_BIDU.dropna()
df_chn_inc_CEA = df_chn_inc_CEA.dropna()
df_chn_inc_LFC = df_chn_inc_LFC.dropna()
```

In [17]:

```
# Clean the dataset, drop all the rows with zero values.
df_index_SPY  = df_index_SPY[~(df_index_SPY == 0).any(axis=1)]
df_index_Dow = df_index_Dow[~(df_index_Dow == 0).any(axis=1)]
df_index_TNX = df_index_TNX[~(df_index_TNX == 0).any(axis=1)]
df_index_IXIC = df_index_IXIC[~(df_index_IXIC == 0).any(axis=1)]

df_usa_inc_GOOG = df_usa_inc_GOOG[~(df_usa_inc_GOOG == 0).any(axis=1)]
df_usa_inc_AIG = df_usa_inc_AIG[~(df_usa_inc_AIG == 0).any(axis=1)]
df_usa_inc_XOM = df_usa_inc_XOM[~(df_usa_inc_XOM == 0).any(axis=1)]
df_usa_inc_UAL = df_usa_inc_UAL[~(df_usa_inc_UAL == 0).any(axis=1)]

df_chn_inc_PTR = df_chn_inc_PTR[~(df_chn_inc_PTR == 0).any(axis=1)]
df_chn_inc_BIDU = df_chn_inc_BIDU[~(df_chn_inc_BIDU == 0).any(axis=1)]
df_chn_inc_CEA = df_chn_inc_CEA[~(df_chn_inc_CEA == 0).any(axis=1)]
df_chn_inc_LFC = df_chn_inc_LFC[~(df_chn_inc_LFC == 0).any(axis=1)]
```

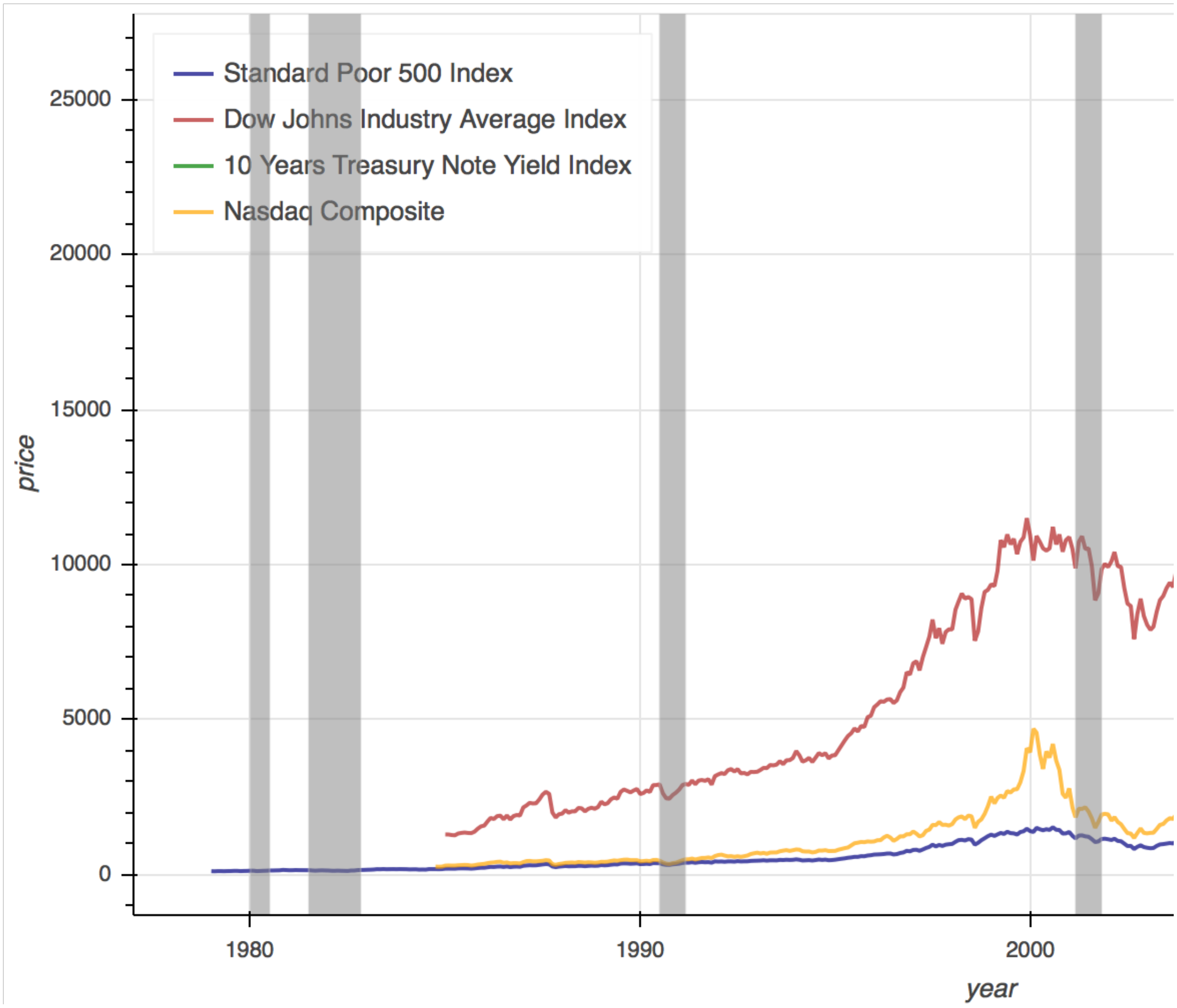
In [18]:

```
# From previous study, we know that the date in all the dataframes are
the index.
# Convert the index of the dataframe into a column
df_index_SPY = df_index_SPY.reset_index()
df_index_Dow = df_index_Dow.reset_index()
df_index_TNX = df_index_TNX.reset_index()
df_index_IXIC = df_index_IXIC.reset_index()


df_usa_inc_GOOG = df_usa_inc_GOOG.reset_index()
df_usa_inc_AIG = df_usa_inc_AIG.reset_index()
df_usa_inc_XOM = df_usa_inc_XOM.reset_index()
df_usa_inc_UAL = df_usa_inc_UAL.reset_index()


df_chn_inc_PTR = df_chn_inc_PTR.reset_index()
df_chn_inc_BIDU = df_chn_inc_BIDU.reset_index()
df_chn_inc_CEA = df_chn_inc_CEA.reset_index()
df_chn_inc_LFC = df_chn_inc_LFC.reset_index()
```

Part 2: Study the impact of economy recession to the prices of the chosen indices and stock prices.

In [19]:

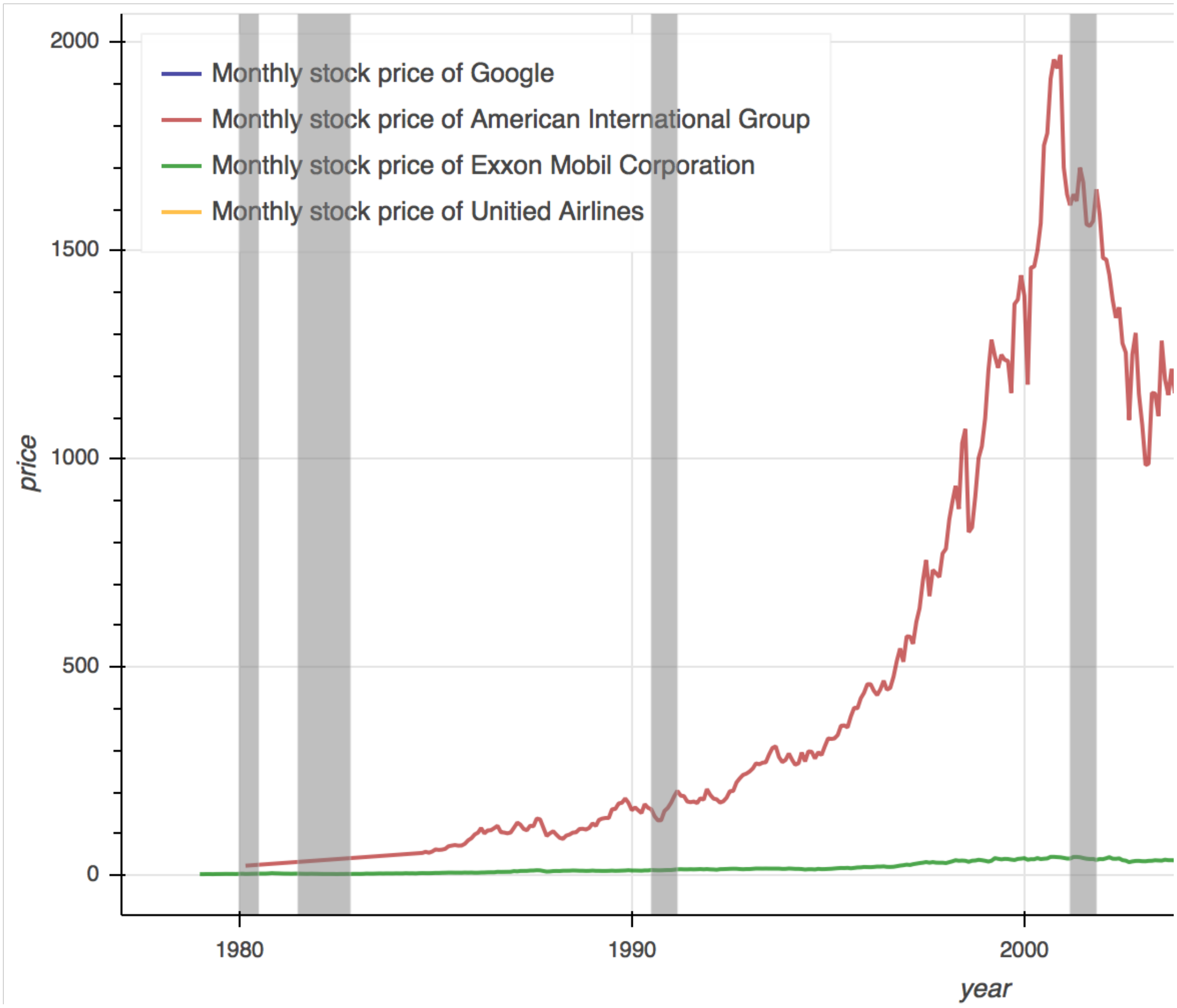
```
# Plot the widely used indexes with Bokeh;
p = figure(plot_width = 950, plot_height = 500, x_axis_type = "datetime")

p.line(df_index_SPY.Date, df_index_SPY['Close'], legend = 'Standard Poor 500 Index',
       alpha = 0.7,  muted_alpha=0.1, line_color = "navy", line_width = 2)
p.line(df_index_Dow.Date, df_index_Dow['Close'], legend = 'Dow Johns Industry Average Index',
       alpha = 0.7,  muted_alpha=0.1, line_color = "firebrick", line_width = 2)
p.line(df_index_TNX.Date, df_index_TNX['Close'], legend = '10 Years Treasury Note Yield Index',
       alpha = 0.7,  muted_alpha=0.1, line_color = "green", line_width = 2)
p.line(df_index_IXIC.Date, df_index_IXIC['Close'], legend = 'Nasdaq Composite',
       alpha = 0.7,  muted_alpha=0.1, line_color = "orange", line_width = 2)

#shade the area with recession during the past 40 years.
box1 = BoxAnnotation(left = datetime(1980,1,1), right = datetime(1980,7,1), fill_alpha = 0.5, fill_color = "grey")
box2 = BoxAnnotation(left = datetime(1981,7,1), right = datetime(1982,11,1), fill_alpha = 0.5, fill_color = "grey")
box3 = BoxAnnotation(left = datetime(1990,7,1), right = datetime(1991,3,1), fill_alpha = 0.5, fill_color = "grey")
box4 = BoxAnnotation(left = datetime(2001,3,1), right = datetime(2001,11,1), fill_alpha = 0.5, fill_color = "grey")
box5 = BoxAnnotation(left = datetime(2007,12,1), right = datetime(2009,6,1), fill_alpha = 0.5, fill_color = "grey")

# hide the line plot by clicking the legend
p.legend.location = "top_left"
p.legend.click_policy = "mute"
p.xaxis.axis_label = "year"
p.yaxis.axis_label = "price"

p.add_layout(box1)
p.add_layout(box2)
p.add_layout(box3)
p.add_layout(box4)
p.add_layout(box5)
show(p)
```



Standard Poor 500 Index
Dow Johns Industry Average Index
10 Years Treasury Note Yield Index
Nasdaq Composite
25000
20000
15000
10000
5000
0
price
1980
1990
2000
year

In [20]:

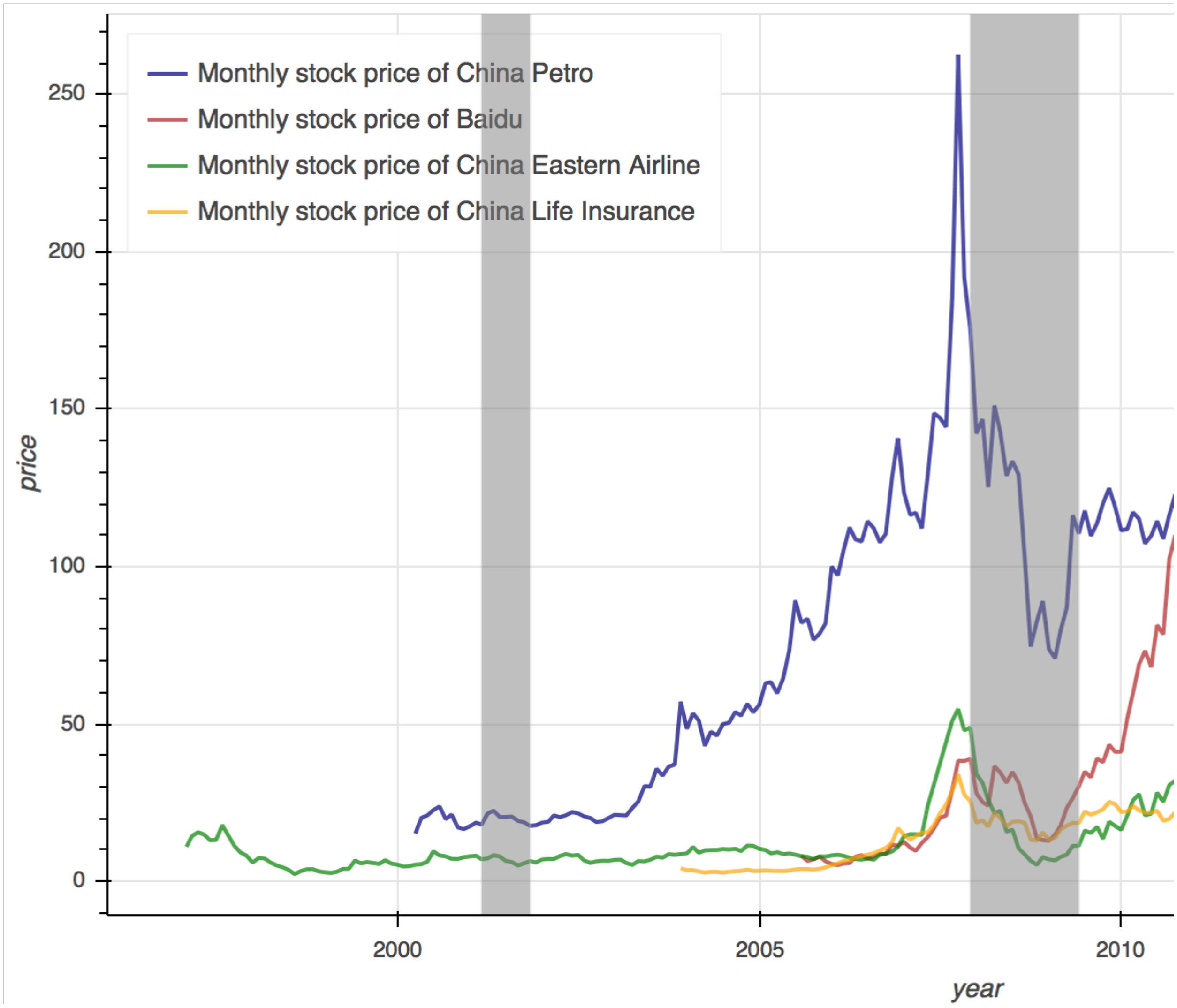
```
# Plot the close price of chosen American public companies with Bokeh;
p = figure(plot_width = 950, plot_height = 500, x_axis_type = "datetim
e")

p.line(df_usa_inc_GOOG.Date, df_usa_inc_GOOG['Close'], legend = 'Month
ly stock price of Google',
       alpha = 0.7,  muted_alpha=0.1, line_color = "navy", line_width
= 2)
p.line(df_usa_inc_AIG.Date, df_usa_inc_AIG['Close'], legend = 'Monthly
stock price of American International Group',
       alpha = 0.7,  muted_alpha=0.1, line_color = "firebrick", line_w
idth = 2)
p.line(df_usa_inc_XOM.Date, df_usa_inc_XOM['Close'], legend = 'Monthly
stock price of Exxon Mobil Corporation',
       alpha = 0.7,  muted_alpha=0.1, line_color = "green", line_width
= 2)
p.line(df_usa_inc_UAL.Date, df_usa_inc_UAL['Close'], legend = 'Monthly
stock price of Unitied Airlines',
       alpha = 0.7,  muted_alpha=0.1, line_color = "orange", line_widt
h = 2)

#shade the area with recession during the past 40 years.
box1 = BoxAnnotation(left = datetime(1980,1,1), right = datetime(1980,
7,1), fill_alpha = 0.5, fill_color = "grey")
box2 = BoxAnnotation(left = datetime(1981,7,1), right = datetime(1982,
11,1), fill_alpha = 0.5, fill_color = "grey")
box3 = BoxAnnotation(left = datetime(1990,7,1), right = datetime(1991,
3,1), fill_alpha = 0.5, fill_color = "grey")
box4 = BoxAnnotation(left = datetime(2001,3,1), right = datetime(2001,
11,1), fill_alpha = 0.5, fill_color = "grey")
box5 = BoxAnnotation(left = datetime(2007,12,1), right = datetime(2009
,6,1), fill_alpha = 0.5, fill_color = "grey")

# hide the line plot by clicking the legend
p.legend.location = "top_left"
p.legend.click_policy = "mute"
p.xaxis.axis_label = "year"
p.yaxis.axis_label = "price"

p.add_layout(box1)
p.add_layout(box2)
p.add_layout(box3)
p.add_layout(box4)
p.add_layout(box5)
show(p)
```



Monthly stock price of Google
Monthly stock price of American International Group
Monthly stock price of Exxon Mobil Corporation
Monthly stock price of Unitied Airlines
price
2000
1500
1000
500
0
1980
1990
2000
year

In [21]:

```
# Plot the close price of chosen Chinese public companies with Bokeh;
p = figure(plot_width = 950, plot_height = 500, x_axis_type = "datetim
e")

p.line(df_chn_inc_PTR.Date, df_chn_inc_PTR['Close'], legend = 'Monthly
stock price of China Petro',
       alpha = 0.7,  muted_alpha=0.1, line_color = "navy", line_width
= 2)
p.line(df_chn_inc_BIDU.Date, df_chn_inc_BIDU['Close'], legend = 'Month
ly stock price of Baidu',
       alpha = 0.7,  muted_alpha=0.1, line_color = "firebrick", line_w
idth = 2)
p.line(df_chn_inc_CEA.Date, df_chn_inc_CEA['Close'], legend = 'Monthly
stock price of China Eastern Airline',
       alpha = 0.7,  muted_alpha=0.1, line_color = "green", line_width
= 2)
p.line(df_chn_inc_LFC.Date, df_chn_inc_LFC['Close'], legend = 'Monthly
stock price of China Life Insurance',
       alpha = 0.7,  muted_alpha=0.1, line_color = "orange", line_widt
h = 2)

#shade the area with recession during the past 40 years.
box1 = BoxAnnotation(left = datetime(1980,1,1), right = datetime(1980,
7,1), fill_alpha = 0.5, fill_color = "grey")
box2 = BoxAnnotation(left = datetime(1981,7,1), right = datetime(1982,
11,1), fill_alpha = 0.5, fill_color = "grey")
box3 = BoxAnnotation(left = datetime(1990,7,1), right = datetime(1991,
3,1), fill_alpha = 0.5, fill_color = "grey")
box4 = BoxAnnotation(left = datetime(2001,3,1), right = datetime(2001,
11,1), fill_alpha = 0.5, fill_color = "grey")
box5 = BoxAnnotation(left = datetime(2007,12,1), right = datetime(2009
,6,1), fill_alpha = 0.5, fill_color = "grey")

# hide the line plot by clicking the legend
p.legend.location = "top_left"
p.legend.click_policy = "mute"
p.xaxis.axis_label = "year"
p.yaxis.axis_label = "price"

p.add_layout(box1)
p.add_layout(box2)
p.add_layout(box3)
p.add_layout(box4)
p.add_layout(box5)
show(p)
```



From the above data visualization of prices of chosen stock market index and stock prices, we can learn that:1) the economic ressession did affect the stock market inddex and each stock we chosen here. 2) Among the five economic ressession happened during the past 40 years, we can see that 2008 global finance crisis had the most negative influences in the stock market, especially for the American International Group. The price almost hit the bottom. It reminded me the U.S. government used 182 billion dollars to bail it out from bankruptcy. 3) Compared with the traditional giants like energy, airline or insurance finance companies, the high tech company has a tremendous growth rate. 4)For Chinese companies listed in the U.S. stock market, it shared the similar pattern with the U.S. counterpart, meaning the market is fair. If one industry is booming, both stock prices of U.S. companies and Chinese companies in that industry are likely to grow.

In [22]:

```
# Add a new column named Price Change in the dataset
df_index_cc_SPY = pd.DataFrame(df_index_SPY)
df_index_cc_SPY['Price Change Pct'] = df_index_cc_SPY.apply(lambda row
: (row.Close -
                                  row.Open)/row.Open, axis = 1)
df_index_cc_SPY.head()
```

Out[22]:

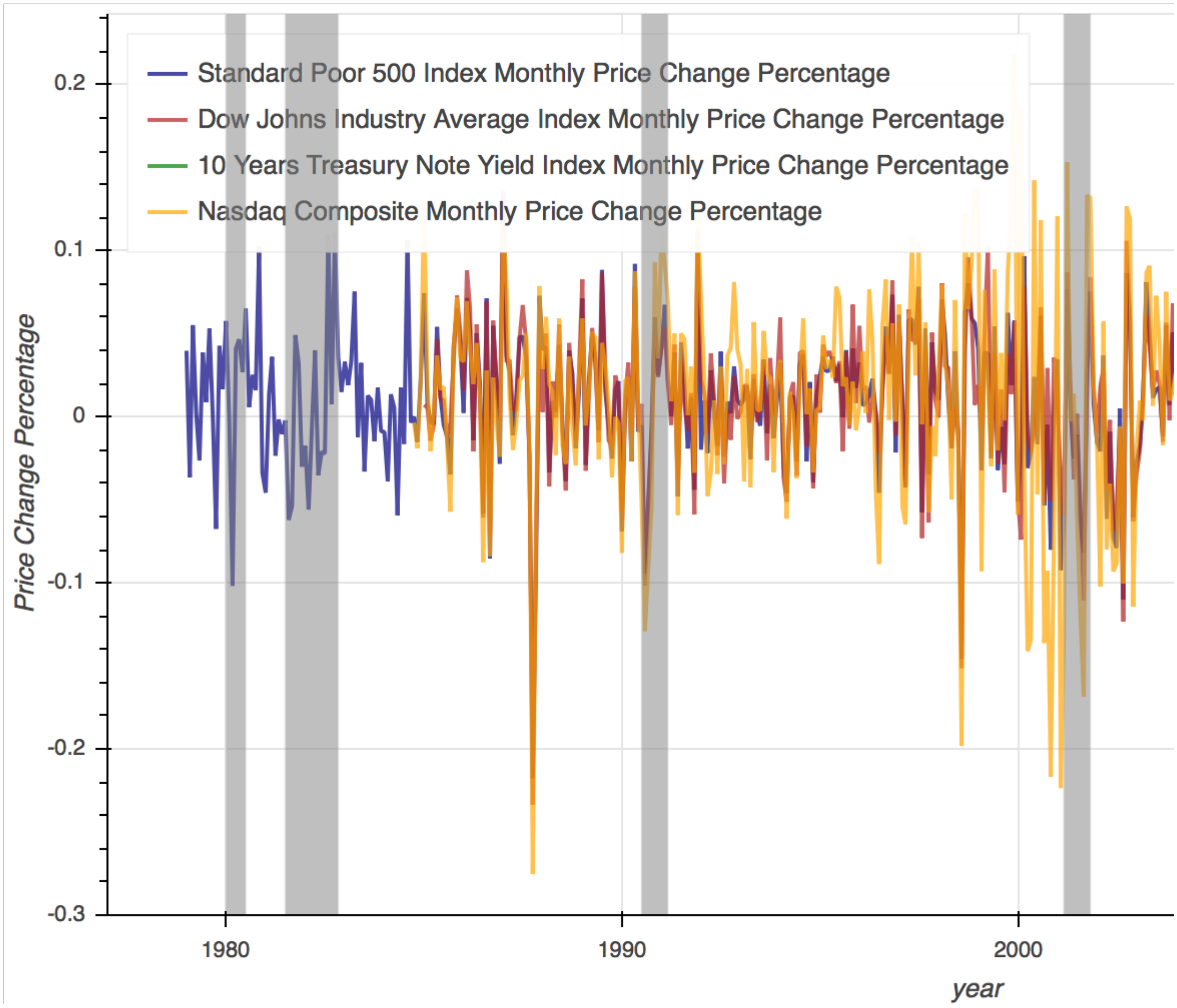
	Date	Open	High	Low	Close	Adj Close	Volume	Price Change Pct
0	1979-01-01	96.110001	102.589996	95.220001	99.930000	99.930000	615730000	0.039746
1	1979-02-01	99.930000	100.519997	95.379997	96.279999	96.279999	475710000	-0.036526
2	1979-03-01	96.279999	103.309998	95.980003	101.589996	101.589996	649800000	0.055152
3	1979-04-01	101.559998	103.949997	100.139999	101.760002	101.760002	620650000	0.001969
4	1979-05-01	101.760002	102.570000	97.489998	99.080002	99.080002	623740000	-0.026336

In [23]:

```
# Repeat the same process for all other dataset
df_index_cc_Dow = pd.DataFrame(df_index_Dow)
df_index_cc_Dow['Price Change Pct'] = df_index_cc_Dow.apply(lambda row
: (row.Close -
                                  row.Open)/row.Open, axis = 1)
df_index_cc_TNX = pd.DataFrame(df_index_TNX)
df_index_cc_TNX['Price Change Pct'] = df_index_cc_TNX.apply(lambda row
: (row.Close -
                                  row.Open)/row.Open, axis = 1)
df_index_cc_IXIC = pd.DataFrame(df_index_IXIC)
df_index_cc_IXIC['Price Change Pct'] = df_index_cc_IXIC.apply(lambda r
ow: (row.Close -
                                  row.Open)/row.Open, axis = 1)

df_index_cc_GOOG = pd.DataFrame(df_usa_inc_GOOG)
df_index_cc_GOOG['Price Change Pct'] = df_index_cc_GOOG.apply(lambda r
ow: (row.Close -
                                  row.Open)/row.Open, axis = 1)
df_index_cc_AIG = pd.DataFrame(df_usa_inc_AIG)
df_index_cc_AIG['Price Change Pct'] = df_index_cc_AIG.apply(lambda row
: (row.Close -
                                  row.Open)/row.Open, axis = 1)
df_index_cc_XOM = pd.DataFrame(df_usa_inc_XOM)
df_index_cc_XOM['Price Change Pct'] = df_index_cc_XOM.apply(lambda row
: (row.Close -
                                  row.Open)/row.Open, axis = 1)
df_index_cc_UAL = pd.DataFrame(df_usa_inc_UAL)
df_index_cc_UAL['Price Change Pct'] = df_index_cc_UAL.apply(lambda row
: (row.Close -
                                  row.Open)/row.Open, axis = 1)

df_index_cc_PTR = pd.DataFrame(df_chn_inc_PTR)
df_index_cc_PTR['Price Change Pct'] = df_index_cc_PTR.apply(lambda row
: (row.Close -
                                  row.Open)/row.Open, axis = 1)
df_index_cc_BIDU = pd.DataFrame(df_chn_inc_BIDU)
df_index_cc_BIDU['Price Change Pct'] = df_index_cc_BIDU.apply(lambda r
ow: (row.Close -
                                  row.Open)/row.Open, axis = 1)
df_index_cc_CEA = pd.DataFrame(df_chn_inc_CEA)
df_index_cc_CEA['Price Change Pct'] = df_index_cc_CEA.apply(lambda row
: (row.Close -
                                  row.Open)/row.Open, axis = 1)
df_index_cc_LFC = pd.DataFrame(df_chn_inc_LFC)
df_index_cc_LFC['Price Change Pct'] = df_index_cc_LFC.apply(lambda row
: (row.Close -
                                  row.Open)/row.Open, axis = 1)
```

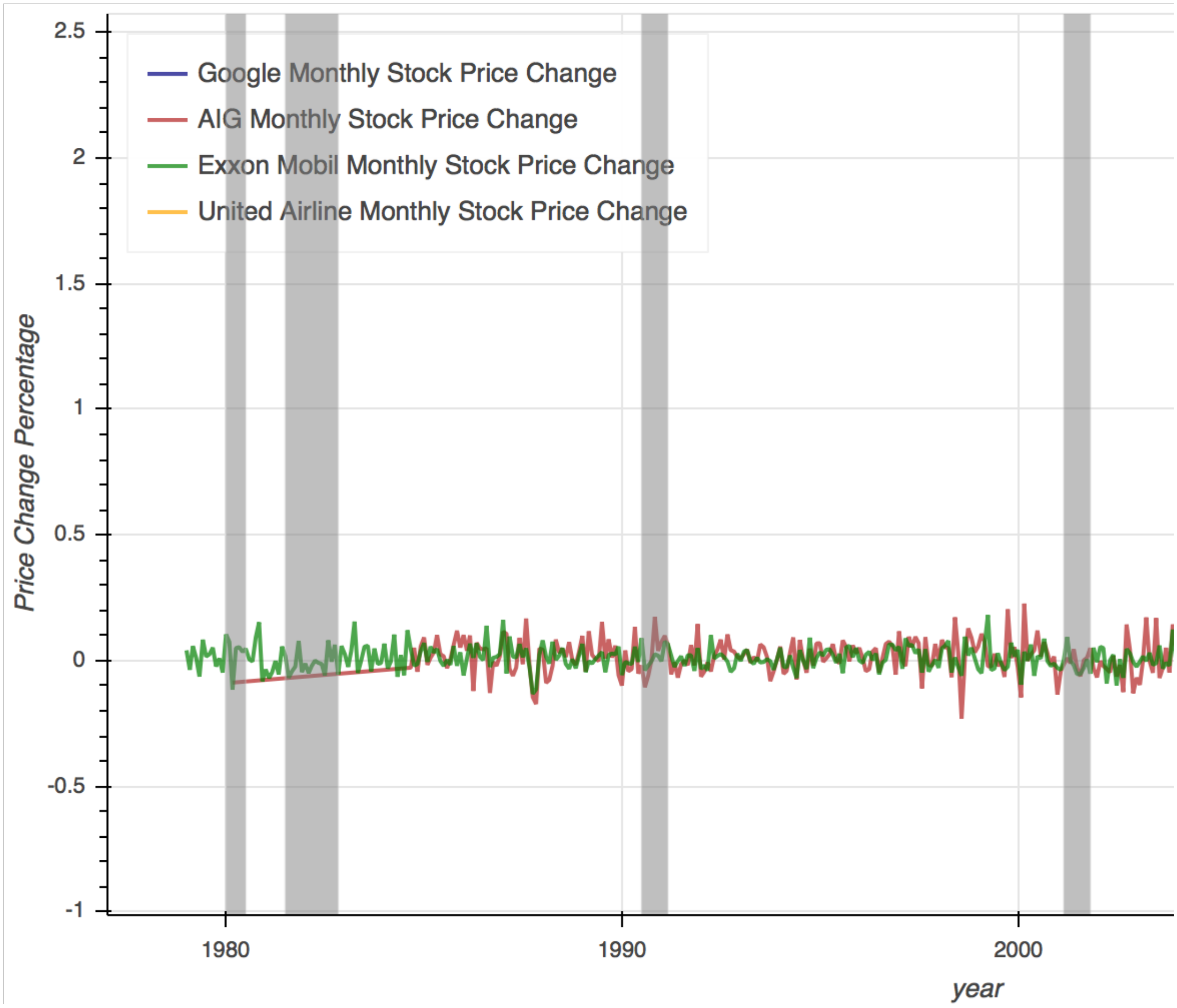
In [24]:

```
# Plot and compare the price change among the indexes with Bokeh:
p = figure(plot_width = 950, plot_height = 500, x_axis_type = "datetime")

p.line(df_index_cc_SPY.Date, df_index_cc_SPY['Price Change Pct'], legend = 'Standard Poor 500 Index Monthly Price Change Percentage',
       alpha = 0.7,  muted_alpha=0.1, line_color = "navy", line_width = 2)
p.line(df_index_cc_Dow.Date, df_index_cc_Dow['Price Change Pct'], legend = 'Dow Johns Industry Average Index Monthly Price Change Percentage',
       alpha = 0.7,  muted_alpha=0.1, line_color = "firebrick", line_width = 2)
p.line(df_index_cc_TNX.Date,df_index_cc_TNX['Price Change Pct'], legend = '10 Years Treasury Note Yield Index Monthly Price Change Percentage',
       alpha = 0.7,  muted_alpha=0.1, line_color = "green", line_width = 2)
p.line(df_index_cc_IXIC.Date,df_index_cc_IXIC['Price Change Pct'], legend= 'Nasdaq Composite Monthly Price Change Percentage',
       alpha = 0.7,  muted_alpha=0.1, line_color = "orange", line_width = 2)

#shade the area with recession during the past 40 years.
box1 = BoxAnnotation(left = datetime(1980,1,1), right = datetime(1980,7,1), fill_alpha = 0.5, fill_color = "grey")
box2 = BoxAnnotation(left = datetime(1981,7,1), right = datetime(1982,11,1), fill_alpha = 0.5, fill_color = "grey")
box3 = BoxAnnotation(left = datetime(1990,7,1), right = datetime(1991,3,1), fill_alpha = 0.5, fill_color = "grey")
box4 = BoxAnnotation(left = datetime(2001,3,1), right = datetime(2001,11,1), fill_alpha = 0.5, fill_color = "grey")
box5 = BoxAnnotation(left = datetime(2007,12,1), right = datetime(2009,6,1), fill_alpha = 0.5, fill_color = "grey")

# hide the line plot by clicking the legend
p.legend.location = "top_left"
p.legend.click_policy = "mute"
p.xaxis.axis_label = "year"
p.yaxis.axis_label = "Price Change Percentage"

p.add_layout(box1)
p.add_layout(box2)
p.add_layout(box3)
p.add_layout(box4)
p.add_layout(box5)
show(p)
```


Standard Poor 500 Index Monthly Price Change Percentage
Dow Johns Industry Average Index Monthly Price Change Percentage
10 Years Treasury Note Yield Index Monthly Price Change Percentage
Nasdaq Composite Monthly Price Change Percentage
Price Change Percentage
0.2
0.1
0
-0.1
-0.2
-0.3
1980
1990
2000
year

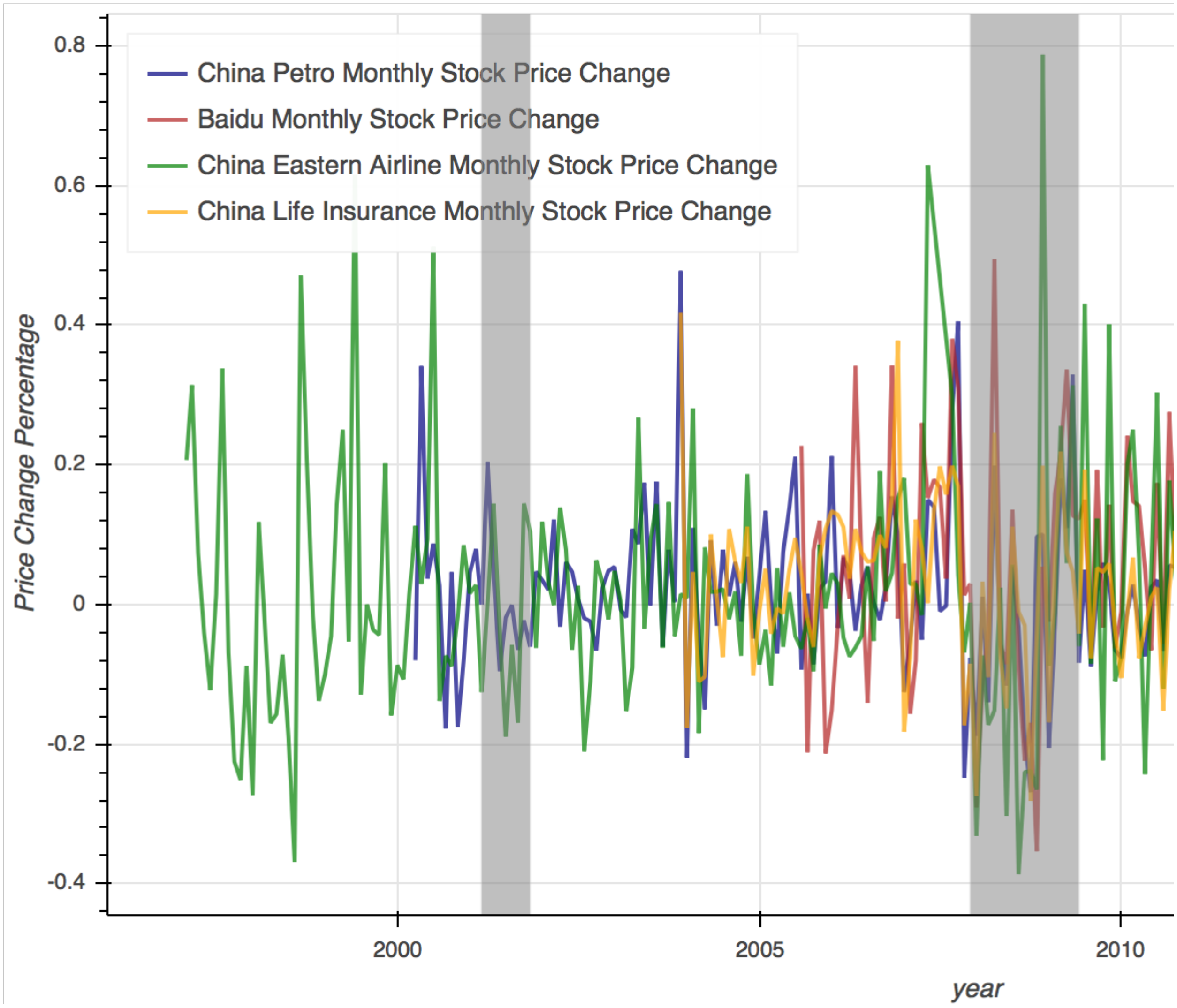
In [25]:

```
# Plot and compare the price change among the chose American stocks with Bokeh:
p = figure(plot_width = 950, plot_height = 500, x_axis_type = "datetime")

p.line(df_index_cc_GOOG.Date, df_index_cc_GOOG['Price Change Pct'], legend = 'Google Monthly Stock Price Change',
       alpha = 0.7,  muted_alpha=0.1, line_color = "navy", line_width = 2)
p.line(df_index_cc_AIG.Date, df_index_cc_AIG['Price Change Pct'], legend = 'AIG Monthly Stock Price Change',
       alpha = 0.7,  muted_alpha=0.1, line_color = "firebrick", line_width = 2)
p.line(df_index_cc_XOM.Date,df_index_cc_XOM['Price Change Pct'], legend = 'Exxon Mobil Monthly Stock Price Change',
       alpha = 0.7,  muted_alpha=0.1, line_color = "green", line_width = 2)
p.line(df_index_cc_UAL.Date,df_index_cc_UAL['Price Change Pct'], legend = 'United Airline Monthly Stock Price Change',
       alpha = 0.7,  muted_alpha=0.1, line_color = "orange", line_width = 2)

#shade the area with recession during the past 40 years.
box1 = BoxAnnotation(left = datetime(1980,1,1), right = datetime(1980,7,1), fill_alpha = 0.5, fill_color = "grey")
box2 = BoxAnnotation(left = datetime(1981,7,1), right = datetime(1982,11,1), fill_alpha = 0.5, fill_color = "grey")
box3 = BoxAnnotation(left = datetime(1990,7,1), right = datetime(1991,3,1), fill_alpha = 0.5, fill_color = "grey")
box4 = BoxAnnotation(left = datetime(2001,3,1), right = datetime(2001,11,1), fill_alpha = 0.5, fill_color = "grey")
box5 = BoxAnnotation(left = datetime(2007,12,1), right = datetime(2009,6,1), fill_alpha = 0.5, fill_color = "grey")

# hide the line plot by clicking the legend
p.legend.location = "top_left"
p.legend.click_policy = "mute"
p.xaxis.axis_label = "year"
p.yaxis.axis_label = "Price Change Percentage"

p.add_layout(box1)
p.add_layout(box2)
p.add_layout(box3)
p.add_layout(box4)
p.add_layout(box5)
show(p)
```



Google Monthly Stock Price Change
AIG Monthly Stock Price Change
Exxon Mobil Monthly Stock Price Change
United Airline Monthly Stock Price Change
Price Change Percentage
2.5
2
1.5
1
0.5
0
-0.5
-1
1980
1990
2000
year

In [26]:
```
# Plot and compare the price change among the chose Chinese stocks wit
h Bokeh:
p = figure(plot_width = 950, plot_height = 500, x_axis_type = "datetim
e")

p.line(df_index_cc_PTR.Date, df_index_cc_PTR['Price Change Pct'], lege
nd = 'China Petro Monthly Stock Price Change',
       alpha = 0.7,  muted_alpha=0.1, line_color = "navy", line_width
= 2)
p.line(df_index_cc_BIDU.Date, df_index_cc_BIDU['Price Change Pct'], le
gend = 'Baidu Monthly Stock Price Change',
       alpha = 0.7,  muted_alpha=0.1, line_color = "firebrick", line_w
idth = 2)
p.line(df_index_cc_CEA.Date,df_index_cc_CEA['Price Change Pct'], legen
d = 'China Eastern Airline Monthly Stock Price Change',
       alpha = 0.7,  muted_alpha=0.1, line_color = "green", line_width
= 2)
p.line(df_index_cc_LFC.Date,df_index_cc_LFC['Price Change Pct'], legen
d = 'China Life Insurance Monthly Stock Price Change',
       alpha = 0.7,  muted_alpha=0.1, line_color = "orange", line_widt
h = 2)

#shade the area with recession during the past 40 years.
box1 = BoxAnnotation(left = datetime(1980,1,1), right = datetime(1980,
7,1), fill_alpha = 0.5, fill_color = "grey")
box2 = BoxAnnotation(left = datetime(1981,7,1), right = datetime(1982,
11,1), fill_alpha = 0.5, fill_color = "grey")
box3 = BoxAnnotation(left = datetime(1990,7,1), right = datetime(1991,
3,1), fill_alpha = 0.5, fill_color = "grey")
box4 = BoxAnnotation(left = datetime(2001,3,1), right = datetime(2001,
11,1), fill_alpha = 0.5, fill_color = "grey")
box5 = BoxAnnotation(left = datetime(2007,12,1), right = datetime(2009
,6,1), fill_alpha = 0.5, fill_color = "grey")

# hide the line plot by clicking the legend
p.legend.location = "top_left"
p.legend.click_policy = "mute"
p.xaxis.axis_label = "year"
p.yaxis.axis_label = "Price Change Percentage"

p.add_layout(box1)
p.add_layout(box2)
p.add_layout(box3)
p.add_layout(box4)
p.add_layout(box5)
show(p)
```



From above data visualization of the price change percentage for the past 40 years, we find that:1) Compare with the major three stock market index (S&P 500, Dow and Nasdaq Composite), S&P 500 has been relatively stable than the other two. 2)From the 4 chose U.S. stocks, the energy company Exxon Mobil has been relatively stable. Over the past 40 years the insurance tycoon AIG has also been relatively stable, except during the time in 2008 finance crisis. Similarly, for Chinese companies, both the energy and insurance company China Petro and China Life Insurance have relatively stable stock prices compared with the other 2. Therefore, to choose a stock, if considering a long term investment, may consider the traditional industry such as energy and life insurance.

Part 3: Choose one of the stock market index to conduct descriptive analysis and predictive analysis on stock prices.

First of all, as we can see from above analysis, the fluctuation of 10 years Treasury Note Yield Index has been in a relatively very small range. Therefore, we decide to not analyze the TNX dataset first.

Among all other three indices, Dow is a stock market index that measures the stock performance of 30 large companies listed on stock exchanges in the United States. Although it is one of the most commonly followed equity indices, since it only includes 30 companies and is not weighted by market capitalization and is not a weighted arithmetic mean.many investors consider the S&P 500 Index, which also includes the 30 components of the Dow, to be a better representation of the U.S. stock market.

In addition, The Nasdaq-100 is heavily allocated towards top performing industries such as Technology, Consumer Services, and Health Care. The traditional industry like utility only weighted 1%, and industrials only weighted 5%. However, the varieties and weighted propotion of industries in S&P 500 are more balanced.And here, 6 out 8 chosen stocks in this study are from traditional industries.

Therefore, among all four chosen stock market indices, we'll choose S&P 500 for further descriptive analysis and predictive analysis.

Descriptive Analysis

In [27]:
```
df_index_SPY.head()
```

Out[27]:

	Date	Open	High	Low	Close	Adj Close	Volume	Price Change Pct
0	1979-01-01	96.110001	102.589996	95.220001	99.930000	99.930000	615730000	0.039746
1	1979-02-01	99.930000	100.519997	95.379997	96.279999	96.279999	475710000	-0.036526
2	1979-03-01	96.279999	103.309998	95.980003	101.589996	101.589996	649800000	0.055152
3	1979-04-01	101.559998	103.949997	100.139999	101.760002	101.760002	620650000	0.001969
4	1979-05-01	101.760002	102.570000	97.489998	99.080002	99.080002	623740000	-0.026336

In [28]:
```
# Reset date into index
df_index_SPY.set_index('Date', inplace=True)
```

In [29]:
```
df_index_SPY.head()
```

Out[29]:

Date	Open	High	Low	Close	Adj Close	Volume	Price Change Pct
1979-01-01	96.110001	102.589996	95.220001	99.930000	99.930000	615730000	0.039746
1979-02-01	99.930000	100.519997	95.379997	96.279999	96.279999	475710000	-0.036526
1979-03-01	96.279999	103.309998	95.980003	101.589996	101.589996	649800000	0.055152
1979-04-01	101.559998	103.949997	100.139999	101.760002	101.760002	620650000	0.001969
1979-05-01	101.760002	102.570000	97.489998	99.080002	99.080002	623740000	-0.026336

In [30]:
```
# Describe the dataset
# Measure the central tendency
# Measure the variablity

df_index_SPY.describe()
```

Out[30]:

	Open	High	Low	Close	Adj Close	Volume	Ch
count	480.000000	480.000000	480.000000	480.000000	480.000000	4.800000e+02	480.00
mean	927.330268	956.528356	896.323250	931.910812	931.910812	3.321781e+10	0.00
std	690.643531	707.897561	669.697820	692.413605	692.413605	3.615204e+10	0.04
min	96.110001	100.519997	94.230003	96.279999	96.279999	4.757100e+08	-0.21
25%	302.359993	316.750008	292.495003	304.000000	304.000000	3.417065e+09	-0.01
50%	918.869995	964.359985	869.385010	919.230011	919.230011	1.557574e+10	0.01
75%	1328.909973	1376.079987	1281.195007	1329.197479	1329.197479	6.642231e+10	0.03
max	2926.290039	2940.909912	2864.120117	2913.979980	2913.979980	1.618436e+11	0.13

1. Measures of central tendency By using the describe function, we can learn that: 1) there are 480 data entries for the monthly price of SPY 500 for the past 40 years; 2) the mean values of SPY500 open price is around $927.33; themeanvalueofSPY500closepriceisaround$931.91; the mean value of transaction volume is around 33.22billion; the mean value of price change is around 0.757%. 3) the median values of SPY500 open price is around $918.87; themedianvalueofSPY500closepriceisaround$919.23; the median value of transaction volume is around 15.76billion; the median value of price change is around 1.04%.
2. Measures of variability we can learn that: 1)the range of SPY500 open price is (96.11, 2926.29); the range of SPY500 close price is around (96.30, 2913.98); the range of transaction volume is (4.757100e+08, 1.618436e+11); the range of price change is around (-21.76%, 13.17%).2) the standard deviation of SPY500 open price is around $690.64; thestandarddeviationofSPY500closepriceisaround$692.41; the standard deviation of transaction volume is around 36.15 billion ; the standard deviation of price change is around 4.26%. Combine with the range and the standard deviation, we can learn that the dataset is pretty spread out from the mean.

In [31]:
```
# Find IQR for outliers of the dataset next step.
Q1 = df_index_SPY.quantile(0.25)
Q3 = df_index_SPY.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

```
Open                 1.026550e+03
High                 1.059330e+03
Low                  9.887000e+02
Close                1.025197e+03
Adj Close            1.025197e+03
Volume               6.300524e+10
Price Change Pct     5.170080e-02
dtype: float64
```

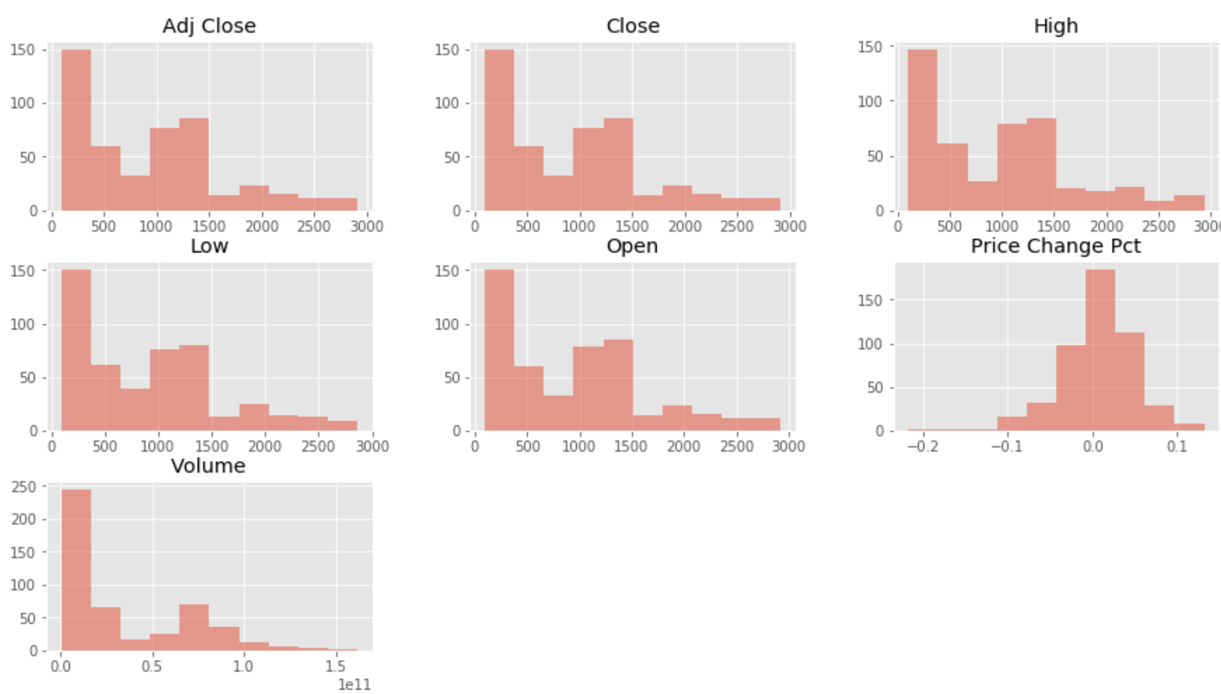
In [32]:
```
# Drop the outliers if there's any.
df_new_index_SPY = df_index_SPY[~((df_index_SPY < (Q1 - 1.5 * IQR)) |(
df_index_SPY > (Q3 + 1.5 * IQR))).any(axis=1)]
df_new_index_SPY.shape
```

Out[32]:
```
(468, 7)
```

Compared with the total count of 480, we now know, we droped 12 rows of outliers.

In [33]:
```
# Measure Kurtosis and Skew.
# Plot histogram for each column.

matplotlib.style.use('ggplot')

df_index_SPY.hist(alpha=0.5, figsize=(15, 8))
```

Out[33]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x1a1ce0dba
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a1d1e35f
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a1d20b86
0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x1a1d230ac
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a1d259f2
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a1cf2f55
0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x1a1d2af9e
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a1d2d5f9
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1a1d2d5fd
0>]],
      dtype=object)
```



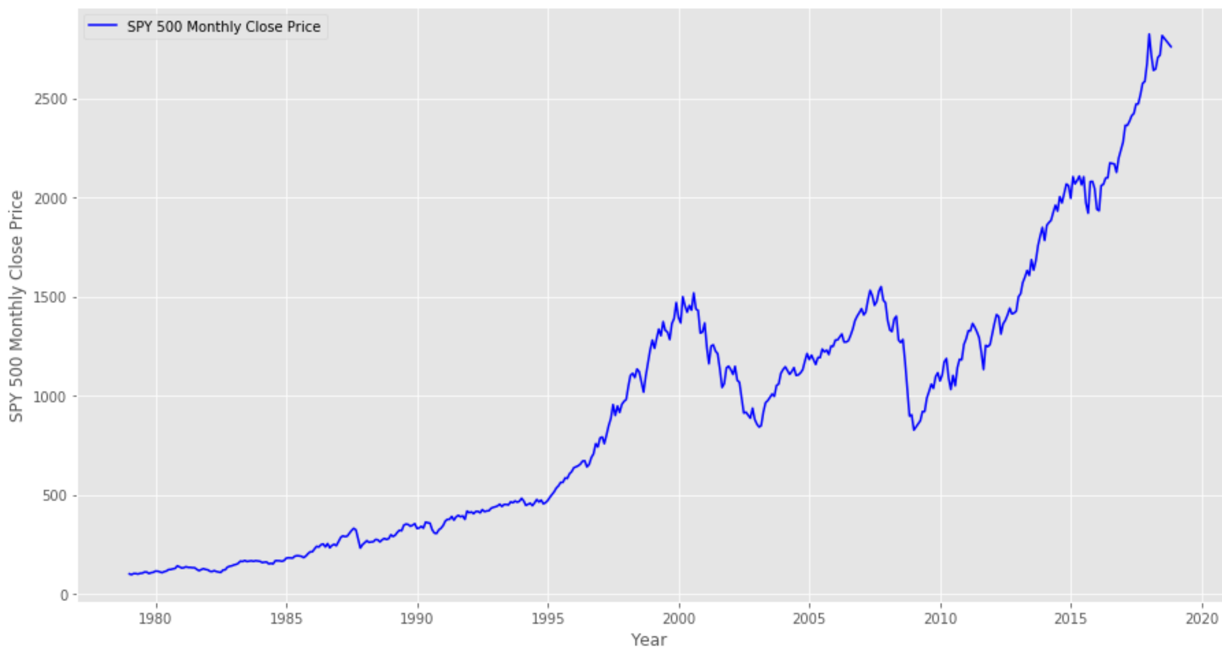
From above histograms, we can see that, except price change percentage, all other 6 columns are obviously right skewed. Price change Percentage looks like Gaussian distribution, although not exactly centered around zero.

Predictive Analysis

1. Time series studies in the close price of SPY500
2. Compare regression models for the transaction volume and the close price of SPY500.

In [34]:

```
#1. Time series studies in the close price of SPY500
# Have a close look of fluctation and irregularity SPY 500 monthly clo
se price.
plt.figure(figsize=(15,8))
plt.plot(df_new_index_SPY.index, df_new_index_SPY.Close,'b',label='SPY
500 Monthly Close Price')

plt.xlabel(r"Year")
plt.ylabel(r"SPY 500 Monthly Close Price")
plt.legend(loc='upper left')
plt.show()
```



From above line chart, we can learn that, it looks like the components on the graph jumps in magnitude overtime, and the pattern of seasonal viation is not roughly stable over the years. Therefore, it might be better to choose multiplicative model instead of addictive model. And compared with ARMA, for this case, it looks like it might be better to choose ARIMA model(Autoregressive Interated Moving Average Model) , let's test the stationarity of the dataset first.

In [35]:
```
# Test stationarity of the dataset
# Creat a new sub-set for time and close price of SPY 500 only
df_cp_SPY = pd.DataFrame(df_new_index_SPY['Close'])
df_cp_SPY.head()
```

Out[35]:

Date	Close
1979-01-01	99.930000
1979-02-01	96.279999
1979-03-01	101.589996
1979-04-01	101.760002
1979-05-01	99.080002

In [36]:
```
#we can split the time series into two contiguous sequences.
#We can then calculate the mean and variance of each group of numbers
and compare the values.

X = df_cp_SPY['Close'].values
split = round(len(X) / 2)
X1, X2 = X[0:split], X[split:]
mean1, mean2 = X1.mean(), X2.mean()
var1, var2 = X1.var(), X2.var()
print('mean1=%f, mean2=%f' % (mean1, mean2))
print('variance1=%f, variance2=%f' % (var1, var2))
```

```
mean1=358.958462, mean2=1484.549528
variance1=61049.909513, variance2=222094.050799
```

We can clearly see that, the mean and variance values are very different. Now we know, the time series sub dataset of SPY 500 Close price is not stationary. To further test it, we can also try a hypothesis testing to verify it.

In [37]:
```
#Augmented Dickey-Fuller test
#Null Hypothesis (H0): If failed to be rejected, it suggests the time
series has a unit root, meaning it is non-stationary. It has some time
dependent structure.
#Alternate Hypothesis (H1): The null hypothesis is rejected; it sugges
ts the time series does not have a unit root, meaning it is stationary
. It does not have time-dependent structure.
```

In [38]:
```
from statsmodels.tsa.stattools import adfuller
result = adfuller(X)
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
        print('\t%s: %.3f' % (key, value))
```

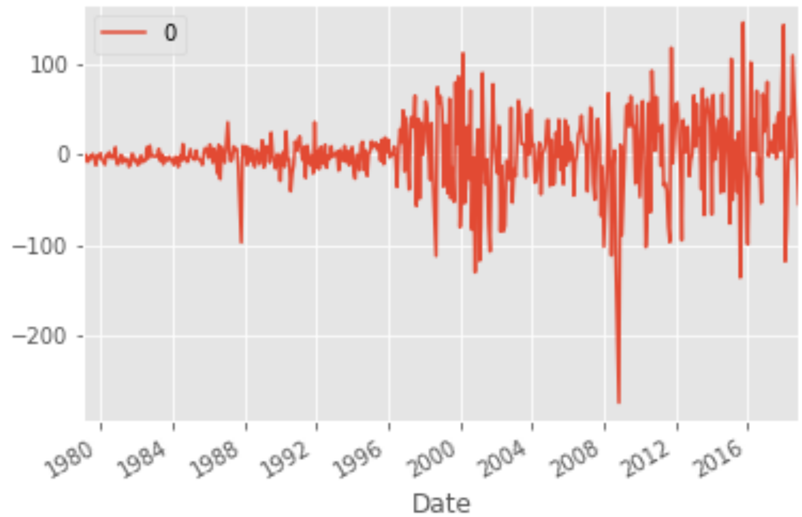
```
ADF Statistic: 1.600418
p-value: 0.997859
Critical Values:
        1%: -3.444
        5%: -2.868
        10%: -2.570
```

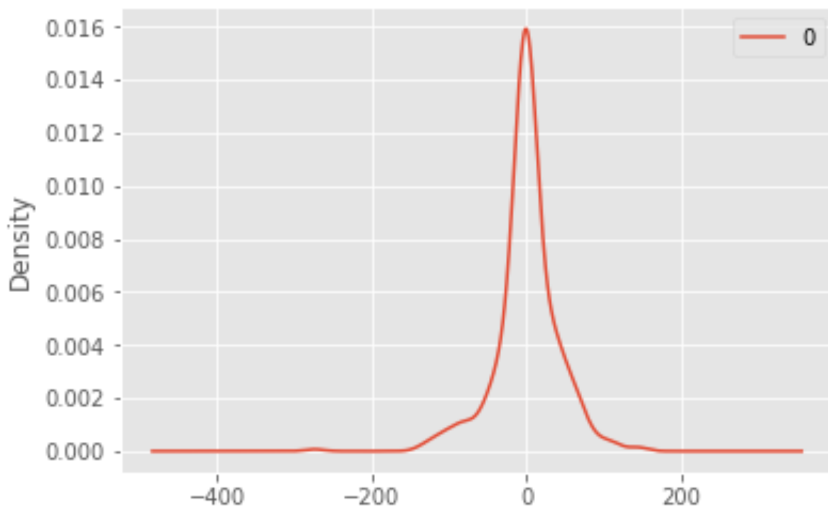
The test statistic value is around 1.6, which is sitting inside the range is less than the critical value 2.57 (10%), and larger than the p-value 0.997859. This suggests that we can not reject the null hypothesis with a significance level of less than 10%,which verified our previous guessing. Therefore, we'll use ARIMA model.

In [47]:
```
from statsmodels.tsa.arima_model import ARIMA

def parser(x):
        return datetime.strptime('190'+x, '%Y-%m')

series1 = df_cp_SPY.Close
X2 = series1.values
# Print the summary of fit model
model = ARIMA(series1, order=(5,1,0))
model_fit = model.fit(disp=0)
print(model_fit.summary())
# plot residual errors

residuals = pd.DataFrame(model_fit.resid)
residuals.plot()
plt.show()
residuals.plot(kind='kde')
plt.show()
print(residuals.describe())
```

```
/anaconda3/lib/python3.6/site-packages/statsmodels/tsa/base/tsa_mode
l.py:225: ValueWarning: A date index has been provided, but it has n
o associated frequency information and so will be ignored when e.g.
forecasting.
  ' ignored when e.g. forecasting.', ValueWarning)
/anaconda3/lib/python3.6/site-packages/statsmodels/tsa/base/tsa_mode
l.py:225: ValueWarning: A date index has been provided, but it has n
o associated frequency information and so will be ignored when e.g.
forecasting.
  ' ignored when e.g. forecasting.', ValueWarning)
```

```
                             ARIMA Model Results
====================================================================
==========
Dep. Variable:                D.Close   No. Observations:
467
Model:                 ARIMA(5, 1, 0)   Log Likelihood
-2395.826
Method:                       css-mle   S.D. of innovations
40.904
Date:                Fri, 08 Jan 2021   AIC
4805.652
Time:                        09:46:47   BIC
4834.676
Sample:                             1   HQIC
4817.074

====================================================================
=============
                    coef    std err          z      P>|z|      [0.02
5      0.975]
--------------------------------------------------------------------
-------------
const             5.7164      2.155      2.653      0.008       1.49
3       9.939
ar.L1.D.Close     0.0019      0.046      0.042      0.967      -0.08
8       0.092
ar.L2.D.Close    -0.0457      0.046     -0.989      0.323      -0.13
6       0.045
ar.L3.D.Close     0.0387      0.046      0.837      0.403      -0.05
2       0.129
ar.L4.D.Close     0.0157      0.046      0.339      0.735      -0.07
5       0.106
ar.L5.D.Close     0.1123      0.046      2.432      0.015       0.02
2       0.203
                                    Roots
====================================================================
=========
                  Real          Imaginary           Modulus
Frequency
--------------------------------------------------------------------
---------
AR.1            1.5115           -0.0000j            1.5115
-0.0000
AR.2           -1.2337           -0.9707j            1.5698
-0.3939
AR.3           -1.2337           +0.9707j            1.5698
0.3939
AR.4            0.4082           -1.4911j            1.5460
-0.2075
AR.5            0.4082           +1.4911j            1.5460
0.2075
--------------------------------------------------------------------
---------
```





	0
count	467.000000
mean	0.007299
std	40.948152
min	-273.488723
25%	-11.845397
50%	-0.475628
75%	14.122300
max	145.055342

From above model analysis, we can learn that: 1) From the residual erros line chart, there's no very obvious trend, hence we can say this model did capture the trend of the dataset. 2) From the density plot of the residual error values, we can learn that the errors are Gaussian and they are centered on zero.And from the residuals'descriptive stastics, we can also learn that the mean value is 0.007299, which is very close to 0, hence we can say that there's merely no bias in the prediction.

In [40]:

```
# Now let's use the fit model to calculate the predictive values and p
lot the prediction using ARIMA model.

from pandas import read_csv
from pandas import datetime
from statsmodels.tsa.arima_model import ARIMA
from sklearn.metrics import mean_squared_error

def parser(x):
        return datetime.strptime('190'+x, '%Y-%m')

series = df_new_index_SPY.Close
X = series.values
size = int(len(X) * 0.66)
train, test = X[0:size], X[size:len(X)]
history = [x for x in train]
predictions = list()
for t in range(len(test)):
        model = ARIMA(history, order=(5,1,0))
        model_fit = model.fit(disp=0)
        output = model_fit.forecast()
        yhat = output[0]
        predictions.append(yhat)
        obs = test[t]
        history.append(obs)
        print('predicted=%f, expected=%f' % (yhat, obs))
error = mean_squared_error(test, predictions)
print('Test MSE: %.3f' % error)
```

```
predicted=1188.006461, expected=1203.599976
predicted=1210.899104, expected=1180.589966
predicted=1189.830754, expected=1156.849976
predicted=1168.998963, expected=1191.500000
predicted=1187.607023, expected=1191.329956
predicted=1195.834057, expected=1234.180054
predicted=1233.187550, expected=1220.329956
predicted=1218.698114, expected=1228.810059
predicted=1239.168170, expected=1207.010010
predicted=1210.230126, expected=1249.479980
predicted=1257.779636, expected=1248.290039
predicted=1247.532314, expected=1280.079956
predicted=1284.123477, expected=1280.660034
predicted=1279.692824, expected=1294.869995
predicted=1304.626270, expected=1310.609985
predicted=1312.460178, expected=1270.089966
predicted=1280.852015, expected=1270.199951
predicted=1275.006310, expected=1276.660034
predicted=1280.379278, expected=1303.819946
predicted=1307.899624, expected=1335.849976
predicted=1330.682847, expected=1377.939941
predicted=1379.101377, expected=1400.630005
predicted=1403.747673, expected=1418.300049
```

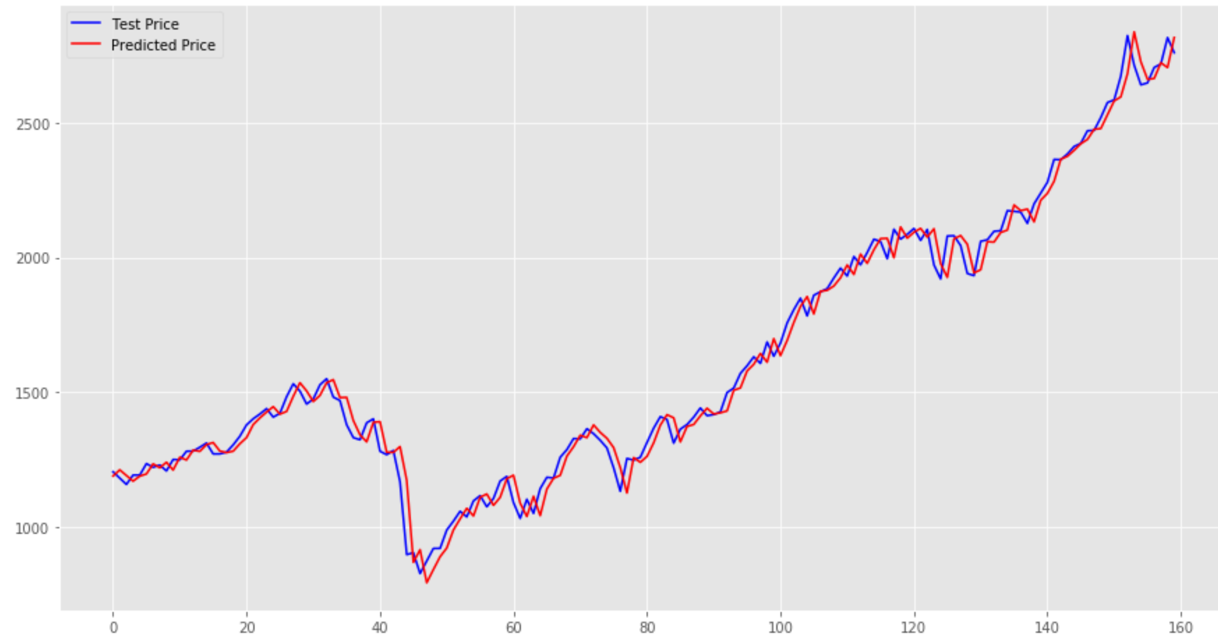
```
predicted=1425.477524, expected=1438.239990
predicted=1445.601288, expected=1406.819946
predicted=1417.908821, expected=1420.859985
predicted=1427.875743, expected=1482.369995
predicted=1483.916672, expected=1530.619995
predicted=1534.497823, expected=1503.349976
predicted=1503.967992, expected=1455.270020
predicted=1464.596472, expected=1473.989990
predicted=1487.348911, expected=1526.750000
predicted=1534.472181, expected=1549.380005
predicted=1545.826886, expected=1481.140015
predicted=1479.284315, expected=1468.359985
predicted=1480.117242, expected=1378.550049
predicted=1394.265570, expected=1330.630005
predicted=1341.090314, expected=1322.699951
predicted=1314.856070, expected=1385.589966
predicted=1387.383438, expected=1400.380005
predicted=1389.813836, expected=1280.000000
predicted=1276.521886, expected=1267.380005
predicted=1275.793845, expected=1282.829956
predicted=1297.418851, expected=1166.359985
predicted=1172.023412, expected=896.239990
predicted=867.544699, expected=903.250000
predicted=914.450057, expected=825.880005
predicted=791.736624, expected=872.809998
predicted=840.813027, expected=919.140015
predicted=888.434612, expected=919.320007
predicted=920.689761, expected=987.479980
predicted=987.729106, expected=1020.619995
predicted=1028.895659, expected=1057.079956
predicted=1068.159521, expected=1036.189941
predicted=1039.718339, expected=1095.630005
predicted=1109.146330, expected=1115.099976
predicted=1121.310020, expected=1073.869995
predicted=1079.209141, expected=1104.489990
predicted=1108.846424, expected=1169.430054
predicted=1178.127653, expected=1186.689941
predicted=1190.940618, expected=1089.410034
predicted=1087.325096, expected=1030.709961
predicted=1037.824600, expected=1101.599976
predicted=1112.835275, expected=1049.329956
predicted=1041.068113, expected=1141.199951
predicted=1139.800986, expected=1183.260010
predicted=1179.587852, expected=1180.550049
predicted=1190.558742, expected=1257.640015
predicted=1261.917443, expected=1286.119995
predicted=1296.865239, expected=1327.219971
predicted=1339.405146, expected=1325.829956
predicted=1330.512657, expected=1363.609985
predicted=1377.769882, expected=1345.199951
predicted=1349.940054, expected=1320.640015
predicted=1328.010264, expected=1292.280029
predicted=1293.616136, expected=1218.890015
```

```
predicted=1219.495893, expected=1131.420044
predicted=1125.051408, expected=1253.300049
predicted=1256.407598, expected=1246.959961
predicted=1239.104658, expected=1257.599976
predicted=1260.747019, expected=1312.410034
predicted=1311.218903, expected=1365.680054
predicted=1376.959502, expected=1408.469971
predicted=1415.678933, expected=1397.910034
predicted=1404.178544, expected=1310.329956
predicted=1314.805139, expected=1362.160034
predicted=1372.054828, expected=1379.319946
predicted=1379.416379, expected=1406.579956
predicted=1411.351120, expected=1440.670044
predicted=1439.708096, expected=1412.160034
predicted=1418.250815, expected=1416.180054
predicted=1422.839053, expected=1426.189941
predicted=1429.798972, expected=1498.109985
predicted=1506.407741, expected=1514.680054
predicted=1515.213973, expected=1569.189941
predicted=1578.288792, expected=1597.569946
predicted=1603.060774, expected=1630.739990
predicted=1643.164463, expected=1606.280029
predicted=1610.795499, expected=1685.729980
predicted=1698.208823, expected=1632.969971
predicted=1635.070930, expected=1681.550049
predicted=1691.449062, expected=1756.540039
predicted=1758.341668, expected=1805.810059
predicted=1817.567679, expected=1848.359985
predicted=1854.538399, expected=1782.589966
predicted=1790.011315, expected=1859.449951
predicted=1873.962674, expected=1872.339966
predicted=1877.732186, expected=1883.949951
predicted=1893.799924, expected=1923.569946
predicted=1924.469204, expected=1960.229980
predicted=1971.889068, expected=1930.670044
predicted=1936.699664, expected=2003.369995
predicted=2012.278896, expected=1972.290039
predicted=1977.818917, expected=2018.050049
predicted=2028.145572, expected=2067.560059
predicted=2069.774728, expected=2058.899902
predicted=2070.763460, expected=1994.989990
predicted=1998.692730, expected=2104.500000
predicted=2113.073742, expected=2067.889893
predicted=2071.418774, expected=2085.510010
predicted=2093.420343, expected=2107.389893
predicted=2107.717598, expected=2063.110107
predicted=2075.176860, expected=2103.840088
predicted=2106.342249, expected=1972.180054
predicted=1977.016403, expected=1920.030029
predicted=1925.823675, expected=2079.360107
predicted=2068.927357, expected=2080.409912
predicted=2081.050835, expected=2043.939941
predicted=2048.586357, expected=1940.239990
```

```
predicted=1942.450382, expected=1932.229980
predicted=1954.089298, expected=2059.739990
predicted=2058.408209, expected=2065.300049
predicted=2056.793603, expected=2096.949951
predicted=2092.514215, expected=2098.860107
predicted=2101.442860, expected=2173.600098
predicted=2194.748450, expected=2170.949951
predicted=2173.585988, expected=2168.270020
predicted=2179.438062, expected=2126.149902
predicted=2131.962990, expected=2198.810059
predicted=2212.426367, expected=2238.830078
predicted=2237.890798, expected=2278.870117
predicted=2282.462460, expected=2363.639893
predicted=2364.427176, expected=2362.719971
predicted=2375.531280, expected=2384.199951
predicted=2397.620516, expected=2411.800049
predicted=2421.965370, expected=2423.409912
predicted=2438.170083, expected=2470.300049
predicted=2475.774530, expected=2471.649902
predicted=2478.402784, expected=2519.360107
predicted=2529.454586, expected=2575.260010
predicted=2581.011367, expected=2584.840088
predicted=2596.089400, expected=2673.610107
predicted=2682.116478, expected=2823.810059
predicted=2837.550461, expected=2713.830078
predicted=2726.582683, expected=2640.870117
predicted=2661.176473, expected=2648.050049
predicted=2664.406544, expected=2705.270020
predicted=2722.834526, expected=2718.370117
predicted=2705.019987, expected=2816.290039
predicted=2816.403284, expected=2760.169922
Test MSE: 3314.363
```

In [41]:

```
# Plot the prediction, and see whether ARIMA model works well.
plt.figure(figsize=(15,8))
plt.plot(test,'b',label='Test Price')
plt.plot(predictions, color='red',label='Predicted Price')

plt.legend(loc='upper left')
plt.show()
```



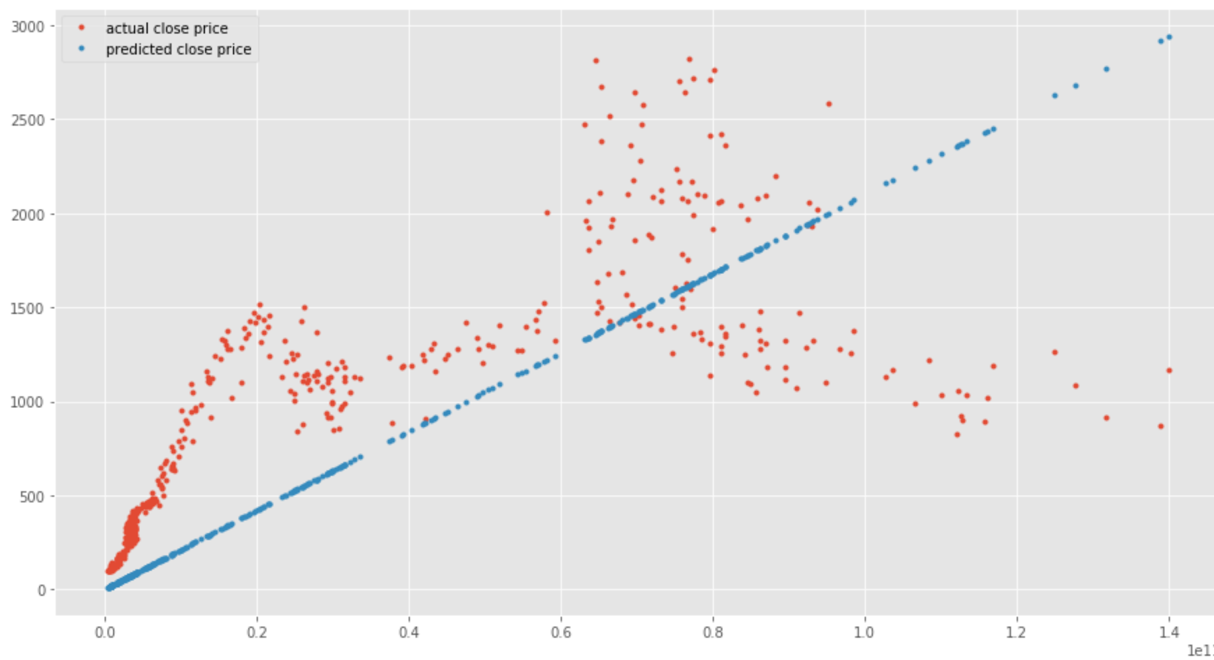
From above chart, we can learn that ARIMA model works well for the time analysis and forecast S&P 500 close price.

Let's also make prediction on the close by the transaction value, and see which model works well.

In [42]:

```
# Use the linear regression model for predictive analyis on SPY 500 Cl
ose price.
from sklearn import linear_model
plt.figure(figsize=(15,8))
X = df_new_index_SPY[['Volume']]
y = df_new_index_SPY['Close'].values

model = linear_model.LinearRegression(fit_intercept=False)
res = model.fit(X, y)
plt.plot(X['Volume'], y, '.')
plt.plot(X['Volume'], model.predict(X), '.')
plt.legend(['actual close price', 'predicted close price'])
```

Out[42]: <matplotlib.legend.Legend at 0x1c22b45f98>



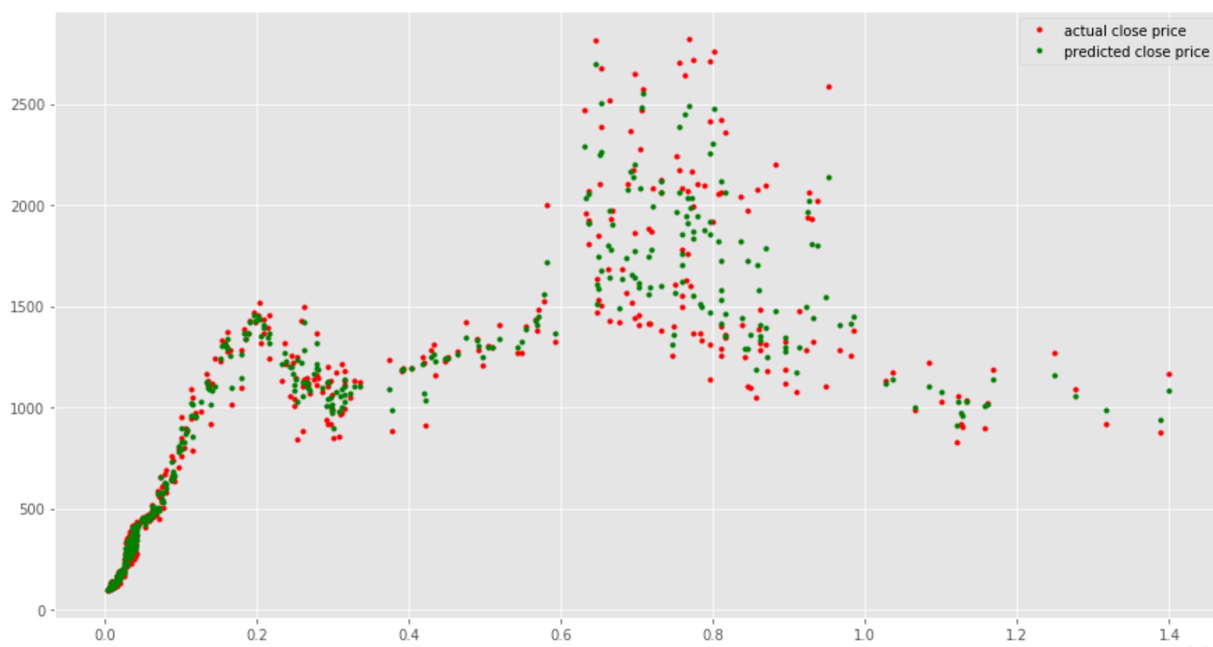
In [43]:
```
# Using the random forest regression model for predictive analyis on SPY 500 Close price.
from sklearn import ensemble
ensemble_model = ensemble.RandomForestRegressor()
ensemble_model.fit(X,y)
```

```
/anaconda3/lib/python3.6/site-packages/sklearn/ensemble/forest.py:246: FutureWarning: The default value of n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

Out[43]:
```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
           oob_score=False, random_state=None, verbose=0, warm_start=False)
```

In [44]:
```
plt.figure(figsize=(15,8))
plt.plot(X[['Volume']], y, 'r.')
plt.plot(X[['Volume']], ensemble_model.predict(X), 'g.')
plt.legend(['actual close price', 'predicted close price'])
```

Out[44]:
```
<matplotlib.legend.Legend at 0x1c22e799e8>
```



Compared with the linear regression model and the random forest model, we can obviously see that, the random forest model works better for the S&P 500 dataset for the predictive analysis. So to predict the close price by the transaction volume, we'll use the random forest regression model.