# 19CSE304 Data Science Final Assignment


## Dheepthi Priyangha S J

## CB.EN.U4CSE20217


## B - Q2

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
%matplotlib inline
import seaborn; seaborn.set()
```

index by time

```
data=pd.read_csv('/content/fremont-bridge.csv',index_col='Date', parse_dates=True)
```

```
data.head()
```

| Date | Fremont Bridge Total | Fremont Bridge East Sidewalk | Fremont Bridge West Sidewalk |
|---|---|---|---|
| 2012-10-03 00:00:00 | 13.0 | 4.0 | 9.0 |
| 2012-10-03 01:00:00 | 10.0 | 4.0 | 6.0 |
| 2012-10-03 02:00:00 | 2.0 | 1.0 | 1.0 |
| 2012-10-03 03:00:00 | 5.0 | 2.0 | 3.0 |
| 2012-10-03 04:00:00 | 7.0 | 6.0 | 1.0 |

```
data.columns = ["Total","West","East"]
data["Total"] = data["West"] + data["East"]
data.head()
```

| Date | Total | West | East |
|---|---|---|---|
| 2012-10-03 00:00:00 | 13.0 | 4.0 | 9.0 |
| 2012-10-03 01:00:00 | 10.0 | 4.0 | 6.0 |
| 2012-10-03 02:00:00 | 2.0 | 1.0 | 1.0 |
| 2012-10-03 03:00:00 | 5.0 | 2.0 | 3.0 |
| 2012-10-03 04:00:00 | 7.0 | 6.0 | 1.0 |

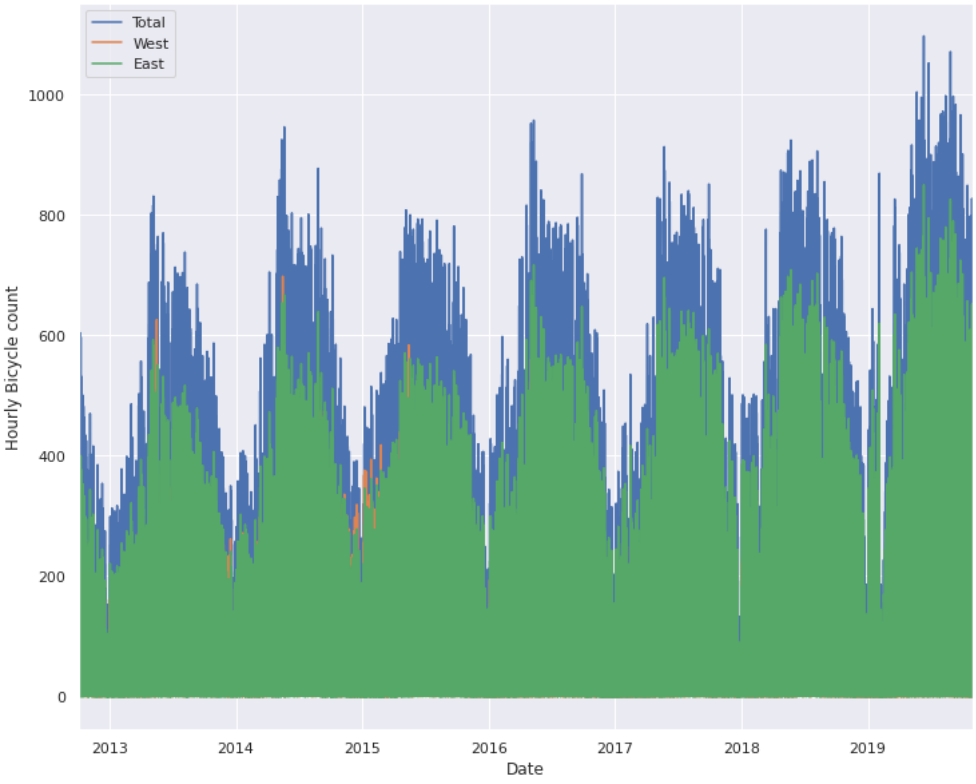
```
data
```

Total West East

```
data.dropna().describe()
```

| | Total | West | East |
|---|---|---|---|
| **count** | 62030.000000 | 62030.000000 | 62030.000000 |
| **mean** | 114.654732 | 52.859455 | 61.795276 |
| **std** | 145.686289 | 67.739036 | 90.417584 |
| **min** | 0.000000 | 0.000000 | 0.000000 |
| **25%** | 15.000000 | 7.000000 | 7.000000 |
| **50%** | 62.000000 | 29.000000 | 30.000000 |
| **75%** | 150.000000 | 71.000000 | 74.000000 |
| **max** | 1097.000000 | 698.000000 | 850.000000 |
| **2019-10-31 23:00:00** | 18.0 | 6.0 | 12.0 |

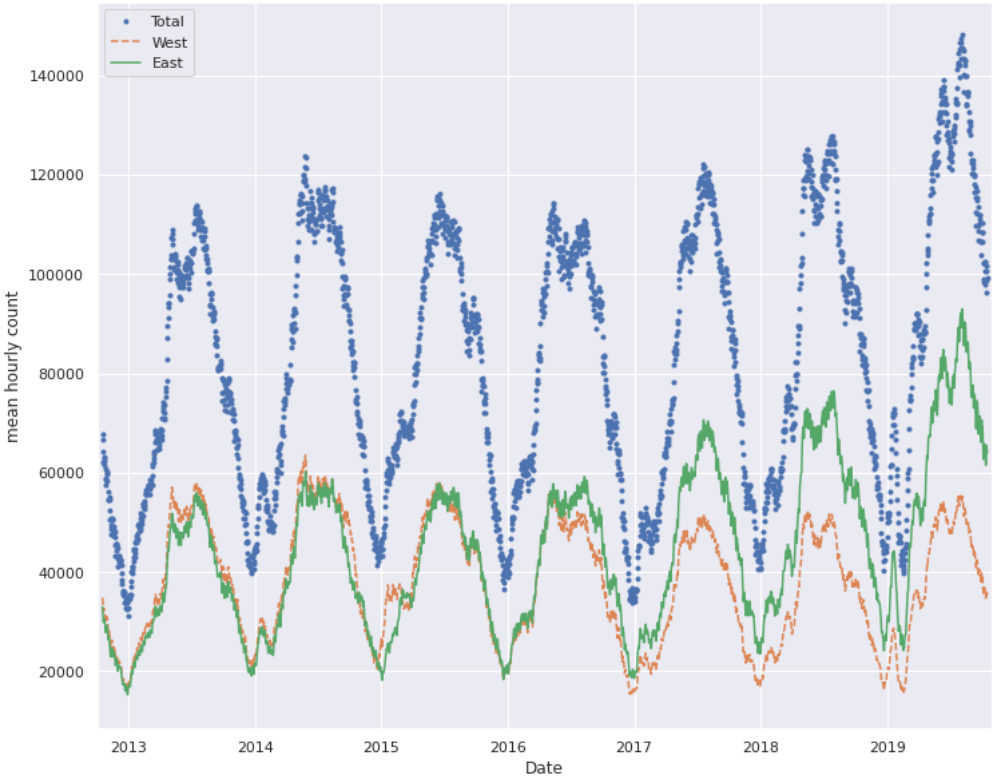
```
data.plot(figsize=(12,10))
plt.ylabel("Hourly Bicycle count")
plt.show()
```




## Resampling

```
weekly = data.resample("W").sum()
weekly.plot(style=['-', '-', '-'],figsize=(12,10))
plt.ylabel('Weekly bicycle count')
plt.show()
```


```
daily = data.resample('D').sum()
daily.rolling(30, center=True).sum().plot(style=['.', '--', '-'],figsize=(12,10))
plt.ylabel('mean hourly count')
plt.show()
```




```
daily = data.resample('D').sum()
daily.rolling(30, center=True).sum().plot(style=[':', '--', '-'],figsize=(12,10))
plt.ylabel('mean hourly count')
plt.show()
```

Total

```
daily = data.resample('M').sum()
daily.rolling(30, center=True).sum().plot(style=['.', '--', '-'],figsize=(12,10))
plt.ylabel('mean hourly count')
plt.show()
```


## Shifting

```
shift1=data.shift(2, axis = 0)
shift1.head()
```

| Date | Total | West | East |
|---|---|---|---|
| 2012-10-03 00:00:00 | NaN | NaN | NaN |
| 2012-10-03 01:00:00 | NaN | NaN | NaN |
| 2012-10-03 02:00:00 | 13.0 | 4.0 | 9.0 |
| 2012-10-03 03:00:00 | 10.0 | 4.0 | 6.0 |
| 2012-10-03 04:00:00 | 2.0 | 1.0 | 1.0 |

```
shift1=data.shift(-2, axis = 0)
shift1.tail()
```

| Date | Total | West | East |
|---|---|---|---|
| 2019-10-31 19:00:00 | 41.0 | 16.0 | 25.0 |
| 2019-10-31 20:00:00 | 32.0 | 14.0 | 18.0 |
| 2019-10-31 21:00:00 | 18.0 | 6.0 | 12.0 |
| 2019-10-31 22:00:00 | NaN | NaN | NaN |
| 2019-10-31 23:00:00 | NaN | NaN | NaN |

```
shift1=data.shift(1, axis = 1)
shift1.head()
```

| Date | Total | West | East |
|---|---|---|---|
| 2012-10-03 00:00:00 | NaN | 13.0 | 4.0 |
| 2012-10-03 01:00:00 | NaN | 10.0 | 4.0 |

```
shift1=data.shift(-1, axis = 1)
shift1.head()
```

| Date | Total | West | East |
|---|---|---|---|
| 2012-10-03 00:00:00 | 4.0 | 9.0 | NaN |
| 2012-10-03 01:00:00 | 4.0 | 6.0 | NaN |
| 2012-10-03 02:00:00 | 1.0 | 1.0 | NaN |
| 2012-10-03 03:00:00 | 2.0 | 3.0 | NaN |
| 2012-10-03 04:00:00 | 6.0 | 1.0 | NaN |

## Time shifting

```
data.isna().any()
```

```
Total    True
West     True
East     True
dtype: bool
```

```
data=data.dropna()
```

```
data['Total'].max()
```

```
1097.0
```

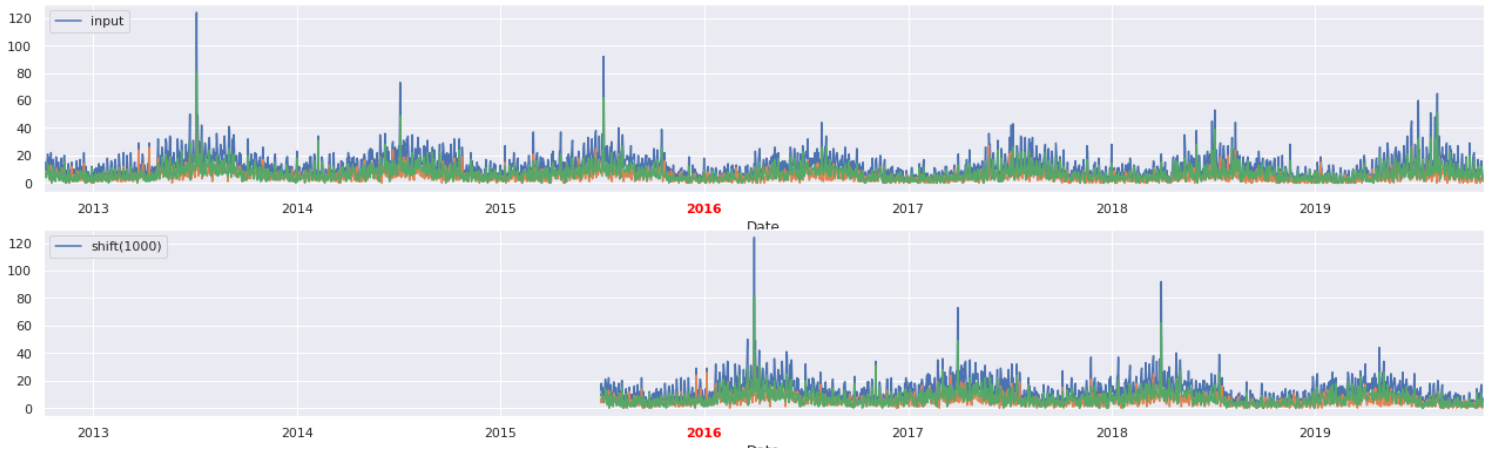
```
data[data['Total']==1097].index
```

```
DatetimeIndex(['2019-06-11 17:00:00'], dtype='datetime64[ns]', name='Date', freq=None)
```

shifting by 1000 days

```
fig, ax = plt.subplots(3, sharey=True)

data = data.dropna().asfreq('D', method='pad')
data.plot(ax=ax[0],figsize=(22,10))
data.shift(1000).plot(ax=ax[1])
data.tshift(1000).plot(ax=ax[2])

local_max = pd.to_datetime('2022-08-01')
offset = pd.Timedelta(1000, 'D')
ax[0].legend(['input'], loc=2)
ax[0].get_xticklabels()[4].set(weight='heavy', color='red')
ax[0].axvline(local_max, alpha=0.3, color='red')


ax[1].legend(['shift(1000)'], loc=2)
ax[1].get_xticklabels()[4].set(weight='heavy', color='red')
ax[1].axvline(local_max + offset, alpha=0.3, color='red')


ax[2].legend(['tshift(1000)'], loc=2)
ax[2].get_xticklabels()[1].set(weight='heavy', color='red')
ax[2].axvline(local_max + offset, alpha=0.3, color='red');
```

```
<ipython-input-53-0541e07a253f>:6: FutureWarning: tshift is deprecated and will be removed in a future version. Please use shift instead.
  data.tshift(1000).plot(ax=ax[2])
```




## Windowing and rolling window

```
data1=data.copy()
data1['Rolling sum of total'] = data1['Total'].rolling(3).sum()
data1
```

| Date | Total | West | East | Rolling sum of total |
|---|---|---|---|---|
| **2012-10-03** | 13.0 | 4.0 | 9.0 | NaN |
| **2012-10-04** | 18.0 | 7.0 | 11.0 | NaN |
| **2012-10-05** | 11.0 | 4.0 | 7.0 | 42.0 |
| **2012-10-06** | 15.0 | 8.0 | 7.0 | 44.0 |
| **2012-10-07** | 11.0 | 6.0 | 5.0 | 37.0 |
| **...** | ... | ... | ... | ... |
| **2019-10-27** | 11.0 | 2.0 | 9.0 | 41.0 |
| **2019-10-28** | 5.0 | 2.0 | 3.0 | 32.0 |
| **2019-10-29** | 3.0 | 1.0 | 2.0 | 19.0 |
| **2019-10-30** | 5.0 | 2.0 | 3.0 | 13.0 |
| **2019-10-31** | 5.0 | 2.0 | 3.0 | 13.0 |

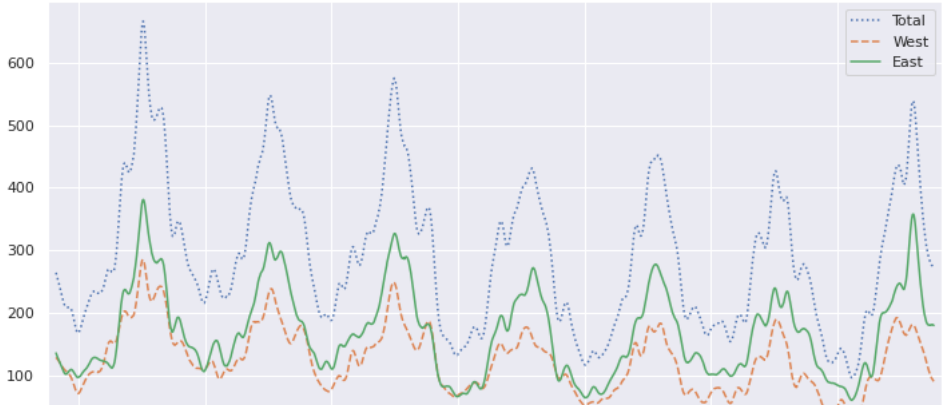
2585 rows × 4 columns

```
data1.drop(['Rolling sum of total'],axis=1)
```

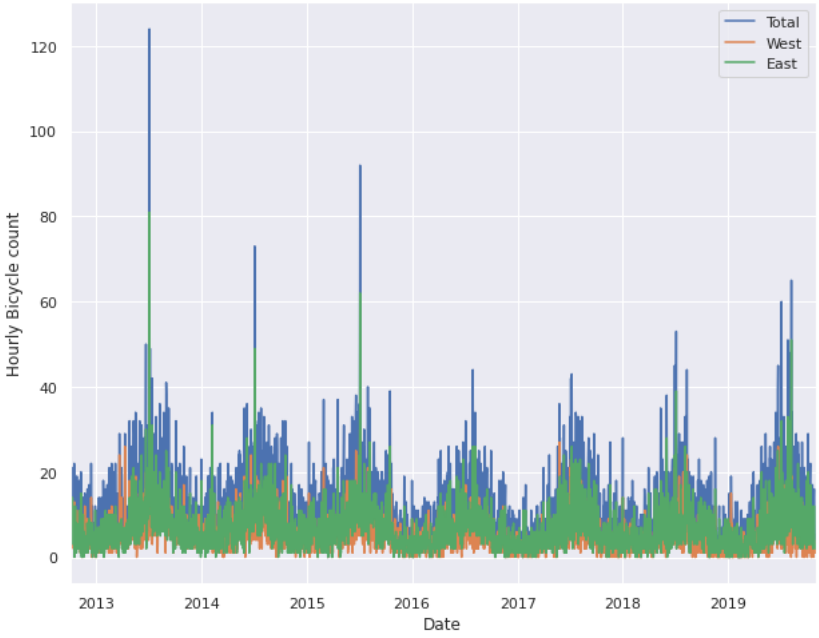
| Date | Total | West | East |
|---|---|---|---|
| **2012-10-03** | 13.0 | 4.0 | 9.0 |
| **2012-10-04** | 18.0 | 7.0 | 11.0 |
| **2012-10-05** | 11.0 | 4.0 | 7.0 |
| **2012-10-06** | 15.0 | 8.0 | 7.0 |
| **2012-10-07** | 11.0 | 6.0 | 5.0 |
| **...** | ... | ... | ... |
| **2019-10-27** | 11.0 | 2.0 | 9.0 |
| **2019-10-28** | 5.0 | 2.0 | 3.0 |
| **2019-10-29** | 3.0 | 1.0 | 2.0 |
| **2019-10-30** | 5.0 | 2.0 | 3.0 |
| **2019-10-31** | 5.0 | 2.0 | 3.0 |

2585 rows × 3 columns

```
data.rolling(50, center=True,win_type='gaussian').sum(std=10).plot(style=[':', '--', '-'],figsize=(12,6));
```




```
data.plot(figsize=(10,8))
plt.ylabel("Hourly Bicycle count")
plt.show()
```




## B - Q3

indexed by time and parsed the values as datetime

```
stock=pd.read_csv('https://raw.githubusercontent.com/wangruinju/python-for-data-analysis/master/pydata-book-2nd-editio
stock.head()
```

| | AAPL | MSFT | XOM | SPX |
|---|---|---|---|---|
| **2003-01-02** | 7.40 | 21.11 | 29.22 | 909.03 |
| **2003-01-03** | 7.45 | 21.14 | 29.24 | 908.59 |
| **2003-01-06** | 7.45 | 21.52 | 29.96 | 929.01 |
| **2003-01-07** | 7.43 | 21.93 | 28.95 | 922.93 |
| **2003-01-08** | 7.28 | 21.31 | 28.83 | 909.93 |

## Indexing,Selection,Subsetting

indexed based on time

```
stock['AAPL'].loc['2003-01-06']
```

```
7.45
```

```
stock.index.is_unique
```

```
True
```

So no need to drop duplicates

```
#Apple stock
stock.loc[:,['AAPL']]
```

| | AAPL |
|---|---|
| **2003-01-02** | 7.40 |
| **2003-01-03** | 7.45 |
| **2003-01-06** | 7.45 |
| **2003-01-07** | 7.43 |
| **2003-01-08** | 7.28 |
| **...** | ... |
| **2011-10-10** | 388.81 |
| **2011-10-11** | 400.29 |
| **2011-10-12** | 402.19 |
| **2011-10-13** | 408.43 |
| **2011-10-14** | 422.00 |

2214 rows × 1 columns

```
#microsoft stock
stock.loc[:,['MSFT']]
```

| | MSFT |
|---|---|
| **2003-01-02** | 21.11 |
| **2003-01-03** | 21.14 |
| **2003-01-06** | 21.52 |
| **2003-01-07** | 21.93 |
| **2003-01-08** | 21.31 |
| **...** | ... |
| **2011-10-10** | 26.94 |
| **2011-10-11** | 27.00 |
| **2011-10-12** | 26.96 |
| **2011-10-13** | 27.18 |
| **2011-10-14** | 27.27 |

2214 rows × 1 columns

```
#Exxon Corp stock
stock.loc[:,['XOM']]
```

| | XOM |
|---|---|
| **2003-01-02** | 29.22 |
| **2003-01-03** | 29.24 |
| **2003-01-06** | 29.96 |
| **2003-01-07** | 28.95 |
| **2003-01-08** | 28.83 |
| **...** | ... |
| **2011-10-10** | 76.28 |
| **2011-10-11** | 76.27 |
| **2011-10-12** | 77.16 |
| **2011-10-13** | 76.37 |
| **2011-10-14** | 78.11 |

2214 rows × 1 columns

```
#S&P 500 stock
stock.loc[:,['SPX']]
```

| | SPX |
|---|---|
| **2003-01-02** | 909.03 |
| **2003-01-03** | 908.59 |
| **2003-01-06** | 929.01 |
| **2003-01-07** | 922.93 |
| **2003-01-08** | 909.93 |
| **...** | ... |
| **2011-10-10** | 1194.89 |
| **2011-10-11** | 1195.54 |
| **2011-10-12** | 1207.25 |
| **2011-10-13** | 1203.66 |
| **2011-10-14** | 1224.58 |

```
stock.loc['2011'].count()
```

```
AAPL    199
MSFT    199
XOM     199
SPX     199
dtype: int64
```

```
stock.loc['2011-10'].count()
```

```
AAPL    10
MSFT    10
XOM     10
SPX     10
dtype: int64
```

```
stock['SPX'][stock['SPX']>1000]
```

```
2003-06-16    1010.74
2003-06-17    1011.66
2003-06-18    1010.09
2003-07-07    1004.42
2003-07-08    1007.84
               ...
2011-10-10    1194.89
2011-10-11    1195.54
2011-10-12    1207.25
2011-10-13    1203.66
2011-10-14    1224.58
Name: SPX, Length: 1847, dtype: float64
```

```
stock['AAPL'][stock['AAPL']>10]
```

```
2003-07-08     10.20
2003-07-17     10.45
2003-07-18     10.43
2003-07-21     10.31
2003-07-22     10.40
               ...
2011-10-10    388.81
2011-10-11    400.29
2011-10-12    402.19
2011-10-13    408.43
2011-10-14    422.00
Name: AAPL, Length: 2069, dtype: float64
```

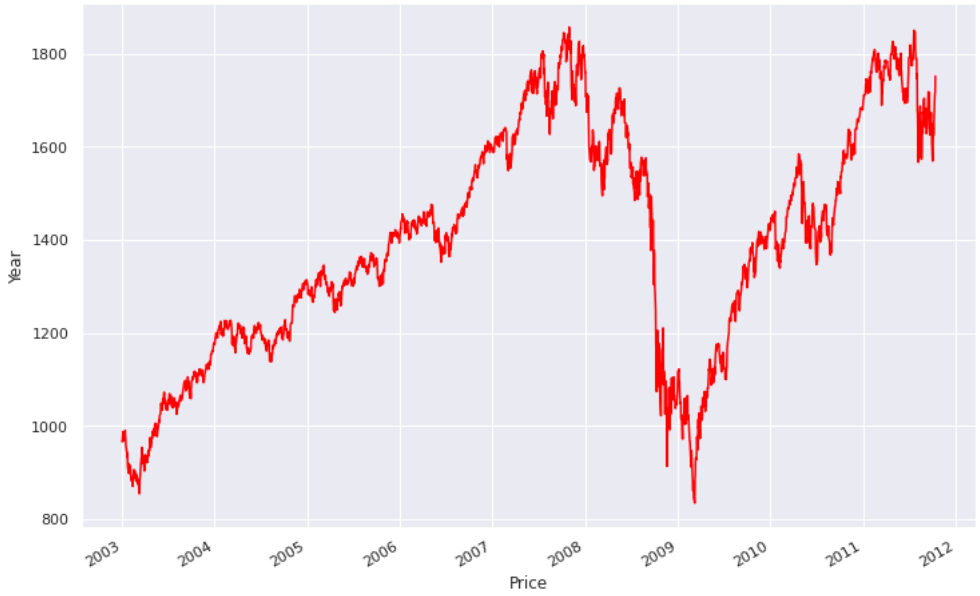
```
stock1=stock.copy()
```

```
stock1['Sum']=stock1['AAPL']+stock1['SPX']+stock1['MSFT']+stock1['XOM']
stock1
```

| | AAPL | MSFT | XOM | SPX | Sum |
|---|---|---|---|---|---|
| **2003-01-02** | 7.40 | 21.11 | 29.22 | 909.03 | 966.76 |
| **2003-01-03** | 7.45 | 21.14 | 29.24 | 908.59 | 966.42 |
| **2003-01-06** | 7.45 | 21.52 | 29.96 | 929.01 | 987.94 |
| **2003-01-07** | 7.43 | 21.93 | 28.95 | 922.93 | 981.24 |

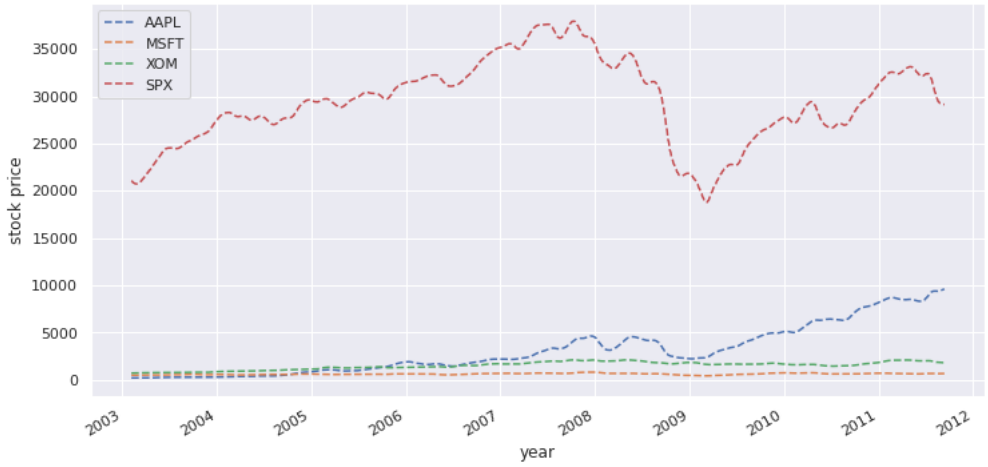
```
stock1.iloc[:,4].plot(figsize=(12,8),color='red')
plt.xlabel('Price')
plt.ylabel('Year')
```

```
Text(0, 0.5, 'Year')
```




```
stock.rolling(50, center=True,win_type='gaussian').sum(std=10).plot(style= '--',figsize=(12,6));
plt.xlabel("year")
plt.ylabel("stock price")
```

```
Text(0, 0.5, 'stock price')
```




## Date Ranges, Frequencies, and Shifting (Leading and Lagging) Data

```
from datetime import datetime

now = datetime.now().time()

print("now =", now)
```

```
now = 17:42:29.567426
```

```
stock2=pd.read_csv('https://raw.githubusercontent.com/wangruinju/python-for-data-analysis/master/pydata-book-2nd-edit
stock2['Unnamed: 0']
```

```
0       2003-01-02 00:00:00
1       2003-01-03 00:00:00
2       2003-01-06 00:00:00
3       2003-01-07 00:00:00
```

```
4       2003-01-08 00:00:00
              ...
2209    2011-10-10 00:00:00
2210    2011-10-11 00:00:00
2211    2011-10-12 00:00:00
2212    2011-10-13 00:00:00
2213    2011-10-14 00:00:00
Name: Unnamed: 0, Length: 2214, dtype: object
```

```
import datetime
datetime.datetime.combine(datetime.date(2011, 1, 1), datetime.time(10, 23))
```

```
datetime.datetime(2011, 1, 1, 10, 23)
```

```
pd.date_range(start='2012-04-01', periods=20)
```

```
DatetimeIndex(['2012-04-01', '2012-04-02', '2012-04-03', '2012-04-04',
               '2012-04-05', '2012-04-06', '2012-04-07', '2012-04-08',
               '2012-04-09', '2012-04-10', '2012-04-11', '2012-04-12',
               '2012-04-13', '2012-04-14', '2012-04-15', '2012-04-16',
               '2012-04-17', '2012-04-18', '2012-04-19', '2012-04-20'],
              dtype='datetime64[ns]', freq='D')
```

```
pd.date_range(end='2012-04-01', periods=20)
```

```
DatetimeIndex(['2012-03-13', '2012-03-14', '2012-03-15', '2012-03-16',
               '2012-03-17', '2012-03-18', '2012-03-19', '2012-03-20',
               '2012-03-21', '2012-03-22', '2012-03-23', '2012-03-24',
               '2012-03-25', '2012-03-26', '2012-03-27', '2012-03-28',
               '2012-03-29', '2012-03-30', '2012-03-31', '2012-04-01'],
              dtype='datetime64[ns]', freq='D')
```

```
pd.date_range('2000-01-01', '2000-01-03 23:59', freq='4h')
```

```
DatetimeIndex(['2000-01-01 00:00:00', '2000-01-01 04:00:00',
               '2000-01-01 08:00:00', '2000-01-01 12:00:00',
               '2000-01-01 16:00:00', '2000-01-01 20:00:00',
               '2000-01-02 00:00:00', '2000-01-02 04:00:00',
               '2000-01-02 08:00:00', '2000-01-02 12:00:00',
               '2000-01-02 16:00:00', '2000-01-02 20:00:00',
               '2000-01-03 00:00:00', '2000-01-03 04:00:00',
               '2000-01-03 08:00:00', '2000-01-03 12:00:00',
               '2000-01-03 16:00:00', '2000-01-03 20:00:00'],
              dtype='datetime64[ns]', freq='4H')
```

```
pd.date_range('2000-01-01', periods=10, freq='1h30min')
```

```
DatetimeIndex(['2000-01-01 00:00:00', '2000-01-01 01:30:00',
               '2000-01-01 03:00:00', '2000-01-01 04:30:00',
               '2000-01-01 06:00:00', '2000-01-01 07:30:00',
               '2000-01-01 09:00:00', '2000-01-01 10:30:00',
               '2000-01-01 12:00:00', '2000-01-01 13:30:00'],
              dtype='datetime64[ns]', freq='90T')
```

```
rng = pd.date_range('2012-01-01', '2012-09-01', freq='WOM-3FRI')
rng
```

```
DatetimeIndex(['2012-01-20', '2012-02-17', '2012-03-16', '2012-04-20',
               '2012-05-18', '2012-06-15', '2012-07-20', '2012-08-17'],
              dtype='datetime64[ns]', freq='WOM-3FRI')
```

```
shift1=stock.shift(1, axis = 1)
shift1.head()
```

| | AAPL | MSFT | XOM | SPX |
|---|---|---|---|---|
| **2003-01-02** | NaN | 7.40 | 21.11 | 29.22 |
| **2003-01-03** | NaN | 7.45 | 21.14 | 29.24 |
| **2003-01-06** | NaN | 7.45 | 21.52 | 29.96 |
| **2003-01-07** | NaN | 7.43 | 21.93 | 28.95 |
| **2003-01-08** | NaN | 7.28 | 21.31 | 28.83 |

```
shift1=stock.shift(1, axis = 0)
shift1.head()
```

| | AAPL | MSFT | XOM | SPX |
|---|---|---|---|---|
| **2003-01-02** | NaN | NaN | NaN | NaN |
| **2003-01-03** | 7.40 | 21.11 | 29.22 | 909.03 |

```
shift1=stock.shift(-2, axis = 0)
shift1.tail()
```

| | AAPL | MSFT | XOM | SPX |
|---|---|---|---|---|
| **2011-10-10** | 402.19 | 26.96 | 77.16 | 1207.25 |
| **2011-10-11** | 408.43 | 27.18 | 76.37 | 1203.66 |
| **2011-10-12** | 422.00 | 27.27 | 78.11 | 1224.58 |
| **2011-10-13** | NaN | NaN | NaN | NaN |
| **2011-10-14** | NaN | NaN | NaN | NaN |

```
shift1=stock.shift(-2, axis = 1)
shift1.tail()
```

| | AAPL | MSFT | XOM | SPX |
|---|---|---|---|---|
| **2011-10-10** | 76.28 | 1194.89 | NaN | NaN |
| **2011-10-11** | 76.27 | 1195.54 | NaN | NaN |
| **2011-10-12** | 77.16 | 1207.25 | NaN | NaN |
| **2011-10-13** | 76.37 | 1203.66 | NaN | NaN |
| **2011-10-14** | 78.11 | 1224.58 | NaN | NaN |

## Time Localization and conversion

```
import pytz
pytz.common_timezones[-5:]
```

```
['US/Eastern', 'US/Hawaii', 'US/Mountain', 'US/Pacific', 'UTC']
```

```
tz = pytz.timezone('America/New_York')
tz
```

```
<DstTzInfo 'America/New_York' LMT-1 day, 19:04:00 STD>
```

```
pd.date_range('3/9/2012 9:30', periods=10, freq='D', tz='UTC')
```

```
DatetimeIndex(['2012-03-09 09:30:00+00:00', '2012-03-10 09:30:00+00:00',
               '2012-03-11 09:30:00+00:00', '2012-03-12 09:30:00+00:00',
               '2012-03-13 09:30:00+00:00', '2012-03-14 09:30:00+00:00',
               '2012-03-15 09:30:00+00:00', '2012-03-16 09:30:00+00:00',
               '2012-03-17 09:30:00+00:00', '2012-03-18 09:30:00+00:00'],
              dtype='datetime64[ns, UTC]', freq='D')
```

```
rng = pd.date_range('3/9/2012 9:30', periods=6, freq='D')
ts = pd.Series(np.random.randn(len(rng)), index=rng)
ts
```

```
2012-03-09 09:30:00   -0.043411
2012-03-10 09:30:00    0.900567
2012-03-11 09:30:00   -0.405461
2012-03-12 09:30:00    1.594490
2012-03-13 09:30:00   -1.491439
2012-03-14 09:30:00    0.382421
Freq: D, dtype: float64
```

```
ts_utc = ts.tz_localize('UTC')
ts_utc
```

```
2012-03-09 09:30:00+00:00   -0.043411
2012-03-10 09:30:00+00:00    0.900567
2012-03-11 09:30:00+00:00   -0.405461
2012-03-12 09:30:00+00:00    1.594490
2012-03-13 09:30:00+00:00   -1.491439
2012-03-14 09:30:00+00:00    0.382421
Freq: D, dtype: float64
```

```
ts_utc.index
```

```
DatetimeIndex(['2012-03-09 09:30:00+00:00', '2012-03-10 09:30:00+00:00',
               '2012-03-11 09:30:00+00:00', '2012-03-12 09:30:00+00:00',
               '2012-03-13 09:30:00+00:00', '2012-03-14 09:30:00+00:00'],
              dtype='datetime64[ns, UTC]', freq='D')
```

```
ts_eastern = ts.tz_localize('America/New_York')
ts_eastern
```

```
2012-03-09 09:30:00-05:00   -0.043411
2012-03-10 09:30:00-05:00    0.900567
2012-03-11 09:30:00-04:00   -0.405461
2012-03-12 09:30:00-04:00    1.594490
2012-03-13 09:30:00-04:00   -1.491439
2012-03-14 09:30:00-04:00    0.382421
dtype: float64
```

```
ts_eastern.tz_convert('UTC')
```

```
2012-03-09 14:30:00+00:00   -0.043411
2012-03-10 14:30:00+00:00    0.900567
2012-03-11 13:30:00+00:00   -0.405461
2012-03-12 13:30:00+00:00    1.594490
2012-03-13 13:30:00+00:00   -1.491439
2012-03-14 13:30:00+00:00    0.382421
dtype: float64
```

## Periods and Period Arithmetic and Period Frequency Conversion

```
p = pd.Period(2010, freq='A-DEC')
p
```

```
Period('2010', 'A-DEC')
```

```
p+5
```

```
Period('2015', 'A-DEC')
```

```
rng = pd.period_range('2000-01-01', '2000-06-30', freq='M')
rng
```

```
PeriodIndex(['2000-01', '2000-02', '2000-03', '2000-04', '2000-05', '2000-06'], dtype='period[M]')
```

```
p.asfreq('M',how='start')
```

```
Period('2010-01', 'M')
```

```
p = pd.Period('2007', freq='A-JUN')
p
```

```
Period('2007', 'A-JUN')
```

```
p.asfreq('M', 'start')
```

```
Period('2006-07', 'M')
```

```
p = pd.Period('2012Q4', freq='Q-JAN')
p
```

```
Period('2012Q4', 'Q-JAN')
```

```
p.asfreq('D', 'end')
```

```
Period('2012-01-31', 'D')
```

```
 rng = pd.date_range('2000-01-01', periods=3, freq='M')
```

```
 ts = pd.Series(np.random.randn(3), index=rng)
```

```
ts
```

```
2000-01-31    0.788718
2000-02-29   -0.538415
```

```
2000-03-31   -0.930824
Freq: M, dtype: float64
```

```
pts=ts.to_period()
pts
```

```
2000-01    0.788718
2000-02   -0.538415
2000-03   -0.930824
Freq: M, dtype: float64
```

## Resampling and frequency conversion

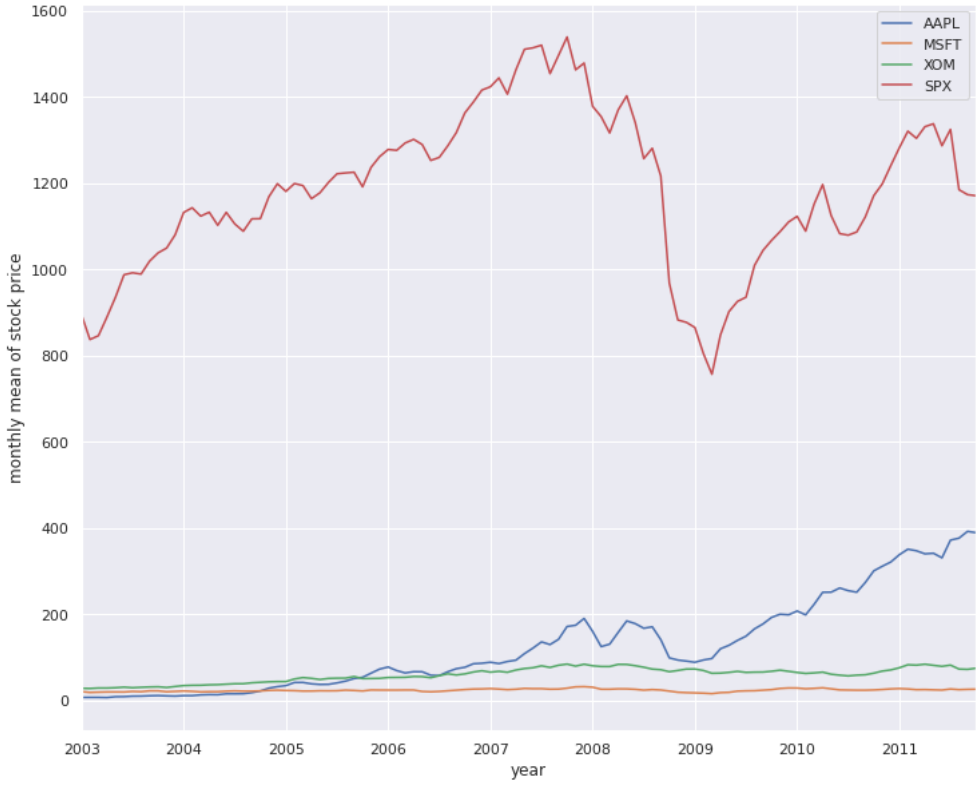
```
mean_month=stock.resample('M').mean()
mean_month
```

| | AAPL | MSFT | XOM | SPX |
|---|---|---|---|---|
| **2003-01-31** | 7.239048 | 20.742381 | 28.356190 | 895.836190 |
| **2003-02-28** | 7.333684 | 18.786316 | 27.985263 | 837.618947 |
| **2003-03-31** | 7.299048 | 19.395238 | 29.065238 | 846.621429 |
| **2003-04-30** | 6.923810 | 19.871429 | 29.050476 | 890.025714 |
| **2003-05-31** | 8.885238 | 19.946667 | 29.728571 | 935.962857 |
| **...** | ... | ... | ... | ... |
| **2011-06-30** | 331.081364 | 24.299091 | 79.365909 | 1287.288636 |
| **2011-07-31** | 372.238000 | 26.799000 | 82.196500 | 1325.184500 |
| **2011-08-31** | 376.762609 | 25.317826 | 73.058696 | 1185.305652 |
| **2011-09-30** | 392.493333 | 25.969048 | 72.399524 | 1173.879048 |
| **2011-10-31** | 389.424000 | 26.370000 | 74.957000 | 1171.356000 |

106 rows × 4 columns

```
mean_month.plot(style='-',figsize=(12,10))
plt.xlabel("year")
plt.ylabel("monthly mean of stock price")
```

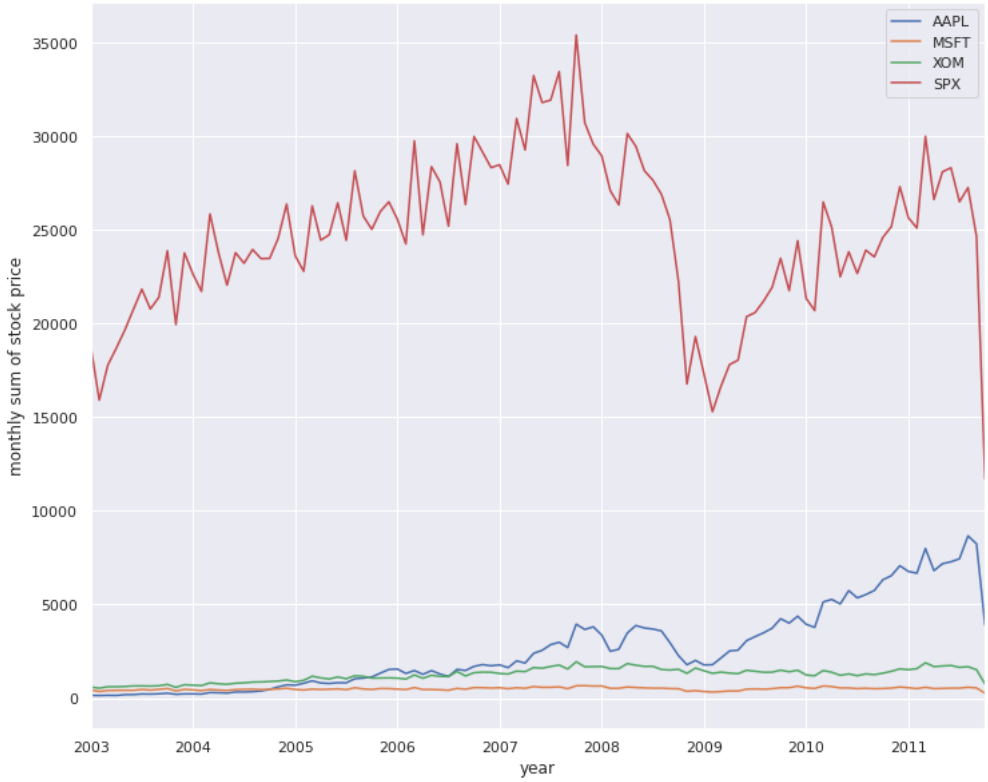
```
Text(0, 0.5, 'monthly mean of stock price')
```




```
sum_month=stock.resample('M').sum()
sum_month
```

| | AAPL | MSFT | XOM | SPX |
|---|---|---|---|---|
| **2003-01-31** | 152.02 | 435.59 | 595.48 | 18812.56 |
| **2003-02-28** | 139.34 | 356.94 | 531.72 | 15914.76 |
| **2003-03-31** | 153.28 | 407.30 | 610.37 | 17779.05 |
| **2003-04-30** | 145.40 | 417.30 | 610.06 | 18690.54 |
| **2003-05-31** | 186.59 | 418.88 | 624.30 | 19655.22 |
| **...** | ... | ... | ... | ... |
| **2011-06-30** | 7283.79 | 534.58 | 1746.05 | 28320.35 |
| **2011-07-31** | 7444.76 | 535.98 | 1643.93 | 26503.69 |
| **2011-08-31** | 8665.54 | 582.31 | 1680.35 | 27262.03 |
| **2011-09-30** | 8242.36 | 545.35 | 1520.39 | 24651.46 |

```
sum_month.plot(style='-',figsize=(12,10))
plt.xlabel("year")
plt.ylabel("monthly sum of stock price")
```



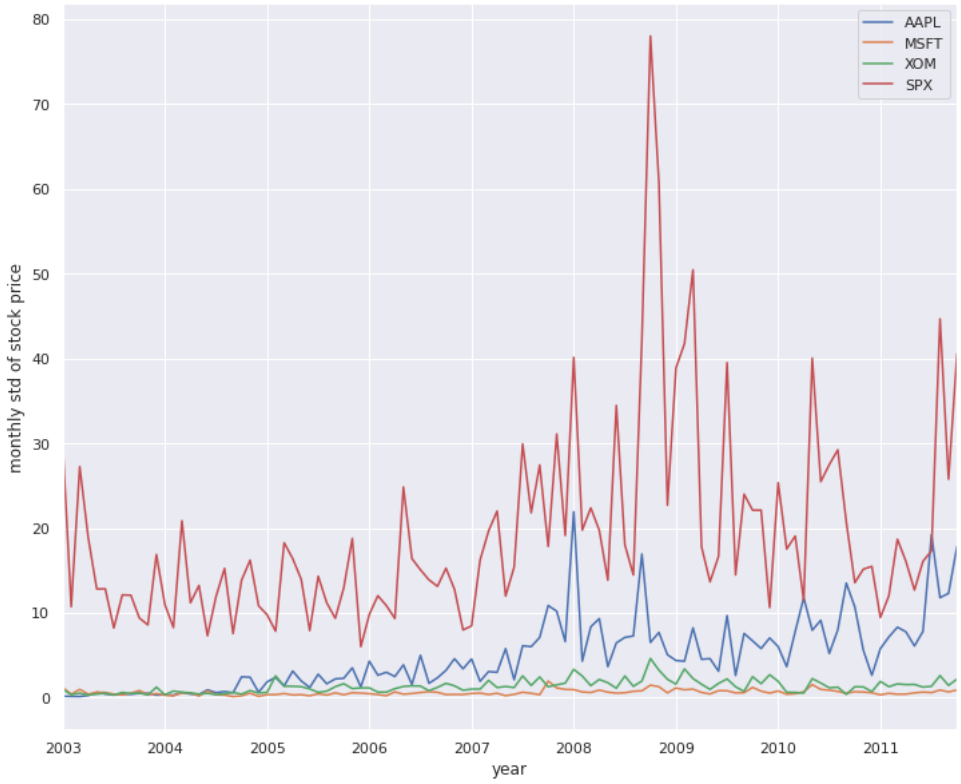

```
std_month=stock.resample('M').std()
std_month
```

| | AAPL | MSFT | XOM | SPX |
|---|---|---|---|---|
| **2003-01-31** | 0.173577 | 1.203287 | 0.982005 | 30.056188 |
| **2003-02-28** | 0.145190 | 0.400738 | 0.355549 | 10.717981 |
| **2003-03-31** | 0.133862 | 0.974862 | 0.529524 | 27.253559 |
| **2003-04-30** | 0.258001 | 0.409100 | 0.314269 | 18.846719 |
| **2003-05-31** | 0.634591 | 0.634896 | 0.414551 | 12.824100 |
| **...** | ... | ... | ... | ... |
| **2011-06-30** | 7.810975 | 0.652752 | 1.243006 | 16.078314 |
| **2011-07-31** | 19.197723 | 0.592167 | 1.336800 | 17.310910 |
| **2011-08-31** | 11.792712 | 0.896731 | 2.595504 | 44.669536 |
| **2011-09-30** | 12.294938 | 0.679838 | 1.444176 | 25.762277 |
| **2011-10-31** | 17.842748 | 0.896177 | 2.195627 | 40.542296 |

106 rows × 4 columns

```
std_month.plot(style='-',figsize=(12,10))

plt.xlabel("year")
plt.ylabel("monthly std of stock price")
```

```
Text(0, 0.5, 'monthly std of stock price')
```




## Downsampling

Aggregating 1 week full of days into a single data

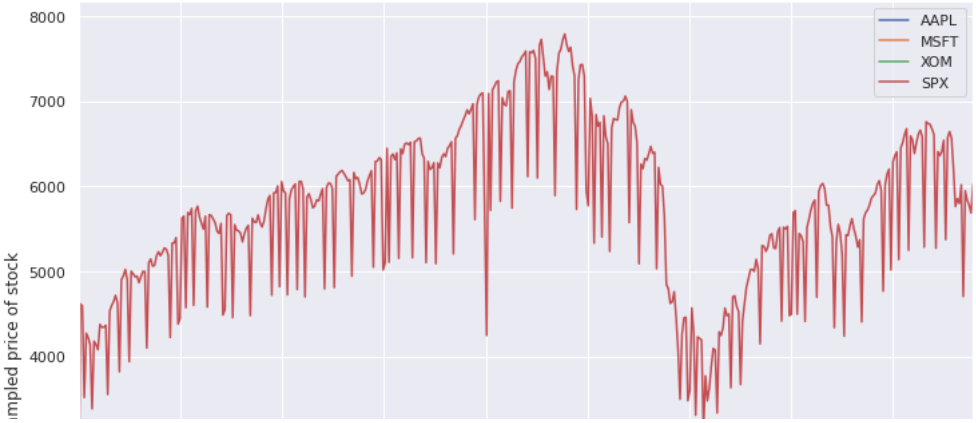
```
downsample=stock.resample('W',closed='right').sum()
downsample
```

| | AAPL | MSFT | XOM | SPX |
|---|---|---|---|---|
| **2003-01-05** | 14.85 | 42.25 | 58.46 | 1817.62 |
| **2003-01-12** | 36.86 | 108.66 | 146.21 | 4617.01 |
| **2003-01-19** | 36.20 | 108.63 | 144.35 | 4592.52 |
| **2003-01-26** | 27.94 | 80.34 | 109.97 | 3514.72 |
| **2003-02-02** | 36.17 | 95.71 | 136.49 | 4270.69 |
| **...** | ... | ... | ... | ... |
| **2011-09-18** | 1947.32 | 132.54 | 364.69 | 5948.94 |
| **2011-09-25** | 2043.34 | 130.30 | 358.23 | 5838.93 |
| **2011-10-02** | 1971.33 | 127.03 | 363.21 | 5781.21 |
| **2011-10-09** | 1872.52 | 128.35 | 365.38 | 5687.64 |
| **2011-10-16** | 2021.72 | 135.35 | 384.19 | 6025.92 |

459 rows × 4 columns

```
downsample.plot(style='-',figsize=(12,10))

plt.xlabel("year")
plt.ylabel("downsampled price of stock")
```

```
Text(0, 0.5, 'downsampled price of stock')
```




## Upsampling

increasing the number of data

```
upsample=stock.resample('H')
upsample
```

```
<pandas.core.resample.DatetimeIndexResampler object at 0x7f8866b951c0>
```

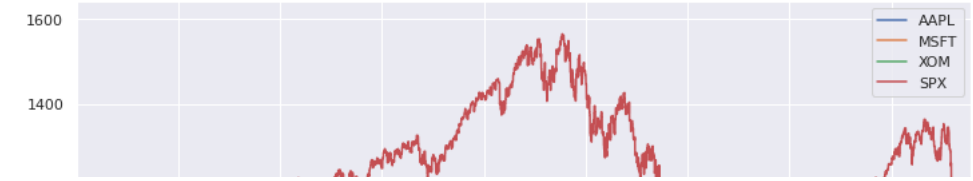
```
stock.resample('H').ffill()
```

| | AAPL | MSFT | XOM | SPX |
|---|---|---|---|---|
| **2003-01-02 00:00:00** | 7.40 | 21.11 | 29.22 | 909.03 |
| **2003-01-02 01:00:00** | 7.40 | 21.11 | 29.22 | 909.03 |
| **2003-01-02 02:00:00** | 7.40 | 21.11 | 29.22 | 909.03 |
| **2003-01-02 03:00:00** | 7.40 | 21.11 | 29.22 | 909.03 |
| **2003-01-02 04:00:00** | 7.40 | 21.11 | 29.22 | 909.03 |
| **...** | ... | ... | ... | ... |
| **2011-10-13 20:00:00** | 408.43 | 27.18 | 76.37 | 1203.66 |
| **2011-10-13 21:00:00** | 408.43 | 27.18 | 76.37 | 1203.66 |
| **2011-10-13 22:00:00** | 408.43 | 27.18 | 76.37 | 1203.66 |
| **2011-10-13 23:00:00** | 408.43 | 27.18 | 76.37 | 1203.66 |
| **2011-10-14 00:00:00** | 422.00 | 27.27 | 78.11 | 1224.58 |

76969 rows × 4 columns

```
a=stock.resample('H').ffill()
a.plot(style='-',figsize=(12,10))

plt.xlabel("year")
plt.ylabel("Stock Price")
```

```
Text(0, 0.5, 'Stock Price')
```




## Moving Window Function


```
stock1=stock[['AAPL','MSFT','XOM']]
stock1=stock1.resample('B').ffill()
```

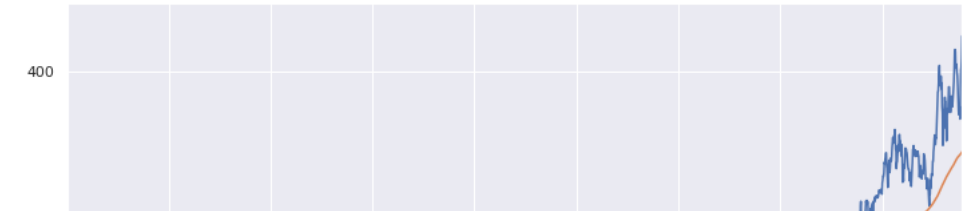
ck

```
stock1.AAPL.plot(figsize=(12,10))
plt.xlabel("year")
plt.ylabel("Apple stock price")
```

```
Text(0, 0.5, 'Apple stock price')
```


```
stock1.AAPL.plot(figsize=(12,10))
stock1.AAPL.rolling(250).mean().plot()

plt.xlabel("year")
plt.ylabel("Apple mean")
```

```
Text(0, 0.5, 'Apple mean')
```




```
stock1.AAPL.plot(figsize=(12,10))
one=stock1.AAPL.rolling(250,min_periods=10).std()
one.plot(figsize=(12,10))

plt.xlabel("year")
plt.ylabel("Apple std")
```

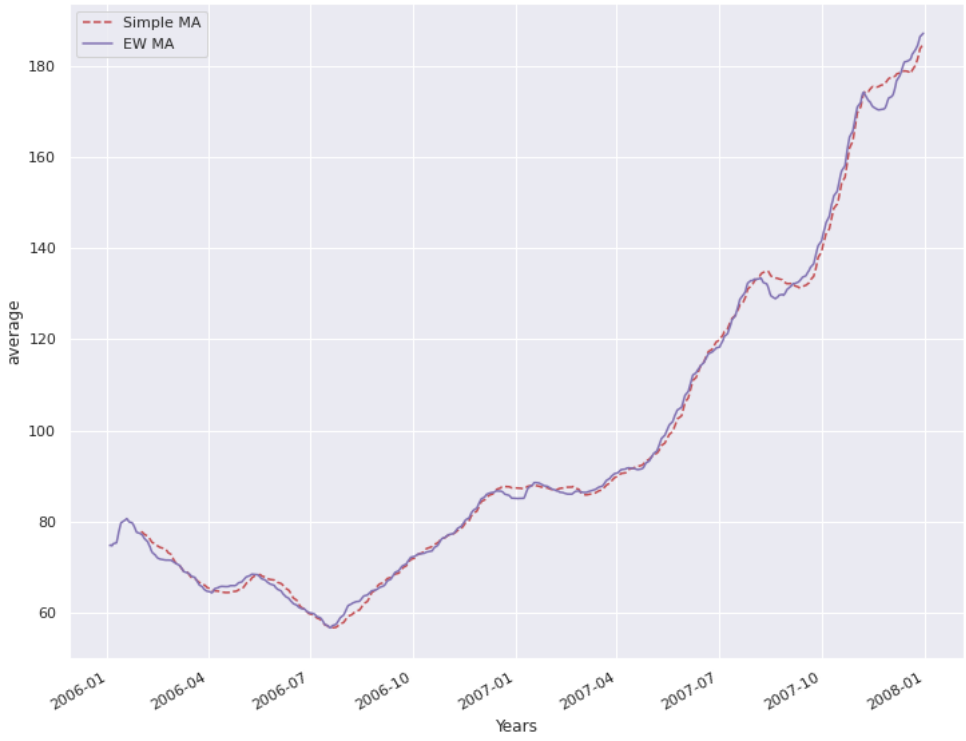
```
Text(0, 0.5, 'Apple std')
```


```
stock.rolling(60).mean().plot(logy=True,figsize=(12,10))

plt.xlabel("average")
plt.ylabel("year")
```

## Exponential Weighted Functions

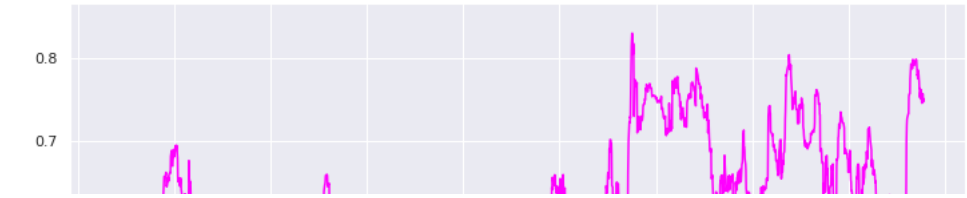
```
Apple=stock.AAPL['2006':'2007']
Moving_average60 = Apple.rolling(30, min_periods=20).mean()
Exponentiallyweighted_moving_average = Apple.ewm(span=30).mean()

Moving_average60.plot(style='r--', label='Simple MA',figsize=(12,10))
Exponentiallyweighted_moving_average.plot(style='m-', label='EW MA')

plt.xlabel("Years")
plt.ylabel("average")

plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f8861e1fa90>
```



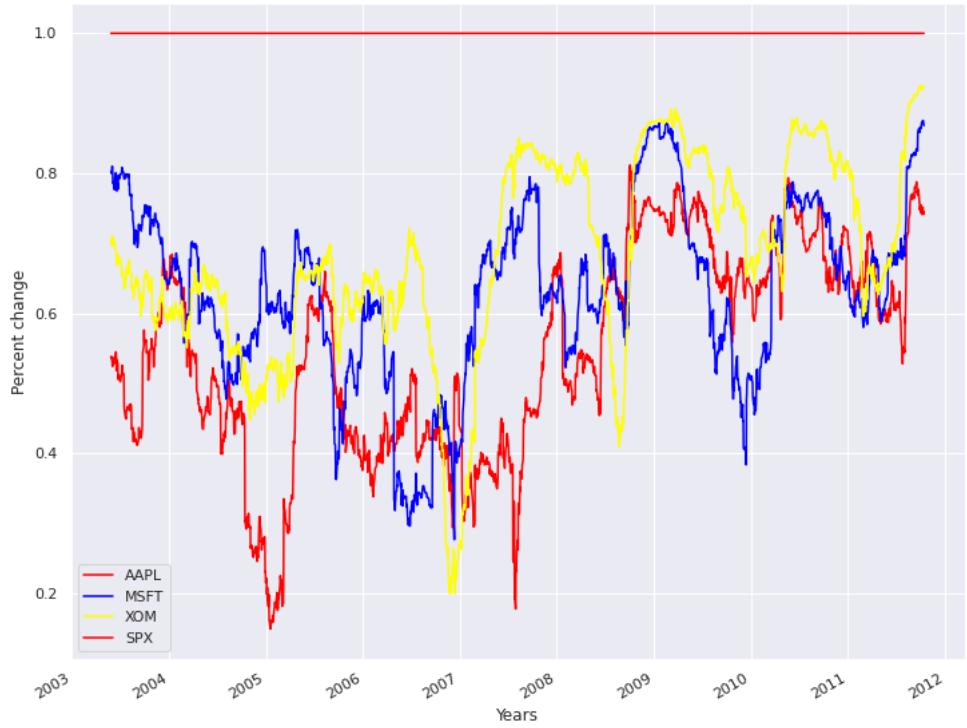

```
spx=stock['SPX']
spx_ret=spx.pct_change()
returns=stock.pct_change()

corr = returns.AAPL.rolling(90, min_periods=85).corr(spx_ret)
corr.plot(figsize=(12,10),color='magenta')


plt.xlabel("Years")
plt.ylabel("Percent change")
```

```
Text(0, 0.5, 'Percent change')
```




```
corr = returns.rolling(100, min_periods=100).corr(spx_ret)
corr.plot(figsize=(12,10),color=['red','blue','yellow'])

plt.xlabel("Years")
plt.ylabel("Percent change")
```

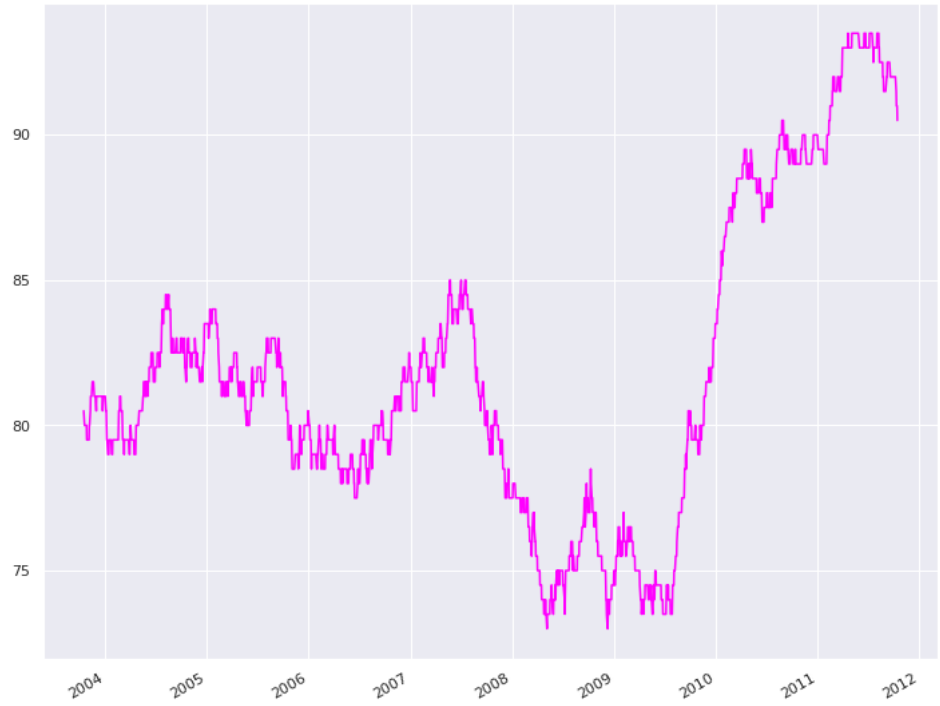
```
Text(0, 0.5, 'Percent change')
```




Using apply to define user defined window functions

```
from scipy.stats import percentileofscore
score_at_2percent = lambda x: percentileofscore(x, 0.02)
result = returns.AAPL.rolling(200).apply(score_at_2percent)
result.plot(figsize=(12,10),color='magenta')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8860963850>
```



✓ 0s completed at 11:15 PM