PandasとMatplotlibの活用

外為クロス円のデータ分析: 月次

In [1]:
```
#https://www.mizuhobank.co.jp/market/historical.htmlからg月次外為クロス円(m_quotes.csv)をダウンロードする
import pandas as pd

fx_historical = pd.read_csv("m_quote.csv")
fx_historical
```

Out[1]:

	Unnamed: 0	USD	GBP	EUR	CAD	CHF	SEK	DKK	NOK	AUD	...	HUF	CZK	PLN	RUB
0	2002/4/30	131.15	189.01	115.97	82.83	79.13	12.73	15.61	15.20	70.24	...	*****	*****	*****	NaN
1	2002/5/31	126.44	184.56	115.88	81.58	79.60	12.57	15.59	15.43	69.58	...	*****	*****	*****	NaN
2	2002/6/28	123.53	183.00	117.83	80.64	80.09	12.94	15.86	15.92	70.29	...	*****	*****	*****	NaN
3	2002/7/31	118.05	183.64	117.23	76.52	80.16	12.67	15.79	15.83	65.42	...	*****	*****	*****	NaN
4	2002/8/30	119.08	183.14	116.45	75.85	79.60	12.59	15.69	15.68	64.47	...	*****	*****	*****	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
213	2020/1/31	109.39	142.93	121.35	83.60	112.82	11.50	16.24	12.21	74.95	...	0.36	4.82	28.57	1.77
214	2020/2/28	109.99	142.71	119.99	82.82	112.68	11.36	16.06	11.85	73.41	...	0.36	4.79	28.06	1.72
215	2020/3/31	107.54	133.35	119.14	77.33	112.37	10.98	15.95	10.61	67.06	...	0.35	4.51	26.93	1.46
216	2020/4/30	107.96	133.97	117.37	76.74	111.26	10.76	15.73	10.35	67.82	...	0.33	4.3	25.83	1.44
217	2020/5/29	107.35	131.83	116.99	76.89	110.64	11.04	15.69	10.65	70.03	...	0.33	4.29	25.83	1.48

218 rows × 37 columns

- [USD、EUR、GBP、CHF、AUD、NZD]通貨のみを選択
- 日付列名を「Date」に変更し、インデックスとして設定します。
- 日付の形式を「MM-YYYY」に変更します

In [2]:
```
fx_historical[['USD','EUR','GBP','CHF','AUD','NZD']]
```

Out[2]:

	USD	EUR	GBP	CHF	AUD	NZD
0	131.15	115.97	189.01	79.13	70.24	58.08
1	126.44	115.88	184.56	79.60	69.58	58.38
2	123.53	117.83	183.00	80.09	70.29	60.48
3	118.05	117.23	183.64	80.16	65.42	56.82
4	119.08	116.45	183.14	79.60	64.47	55.28
...	...	...	...	...	...	...
213	109.39	121.35	142.93	112.82	74.95	72.20
214	109.99	119.99	142.71	112.68	73.41	70.44
215	107.54	119.14	133.35	112.37	67.06	65.13
216	107.96	117.37	133.97	111.26	67.82	64.76
217	107.35	116.99	131.83	110.64	70.03	65.38

218 rows × 6 columns

In [3]:
```
fxhm =  fx_historical.rename(columns={"Unnamed: 0": "Date"}, errors="raise")
```

In [4]:
```
fxhm.head()
```

Out[4]:

	Date	USD	GBP	EUR	CAD	CHF	SEK	DKK	NOK	AUD	...	HUF	CZK	PLN	RUB	TRY
0	2002/4/30	131.15	189.01	115.97	82.83	79.13	12.73	15.61	15.20	70.24	...	*****	*****	*****	NaN	NaN
1	2002/5/31	126.44	184.56	115.88	81.58	79.60	12.57	15.59	15.43	69.58	...	*****	*****	*****	NaN	NaN
2	2002/6/28	123.53	183.00	117.83	80.64	80.09	12.94	15.86	15.92	70.29	...	*****	*****	*****	NaN	NaN
3	2002/7/31	118.05	183.64	117.23	76.52	80.16	12.67	15.79	15.83	65.42	...	*****	*****	*****	NaN	NaN
4	2002/8/30	119.08	183.14	116.45	75.85	79.60	12.59	15.69	15.68	64.47	...	*****	*****	*****	NaN	NaN

5 rows × 37 columns

In [5]:
```
fxhm1=fxhm[['Date','USD','EUR','GBP','CHF','AUD','NZD']]
fxhm1.head()
```

Out[5]:

	Date	USD	EUR	GBP	CHF	AUD	NZD
0	2002/4/30	131.15	115.97	189.01	79.13	70.24	58.08
1	2002/5/31	126.44	115.88	184.56	79.60	69.58	58.38
2	2002/6/28	123.53	117.83	183.00	80.09	70.29	60.48
3	2002/7/31	118.05	117.23	183.64	80.16	65.42	56.82
4	2002/8/30	119.08	116.45	183.14	79.60	64.47	55.28

In [6]:
```
fxhm1['Date']=pd.to_datetime(fxhm1['Date'])
fxhm1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 218 entries, 0 to 217
Data columns (total 7 columns):
Date    218 non-null datetime64[ns]
USD     218 non-null float64
EUR     218 non-null float64
GBP     218 non-null float64
CHF     218 non-null float64
AUD     218 non-null float64
NZD     218 non-null float64
dtypes: datetime64[ns](1), float64(6)
memory usage: 12.0 KB

D:¥Anaconda3¥lib¥site-packages¥ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/inde
xing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

In [7]:
```
fxhm1=fxhm1.set_index(fxhm1['Date']).drop('Date', 1)
```

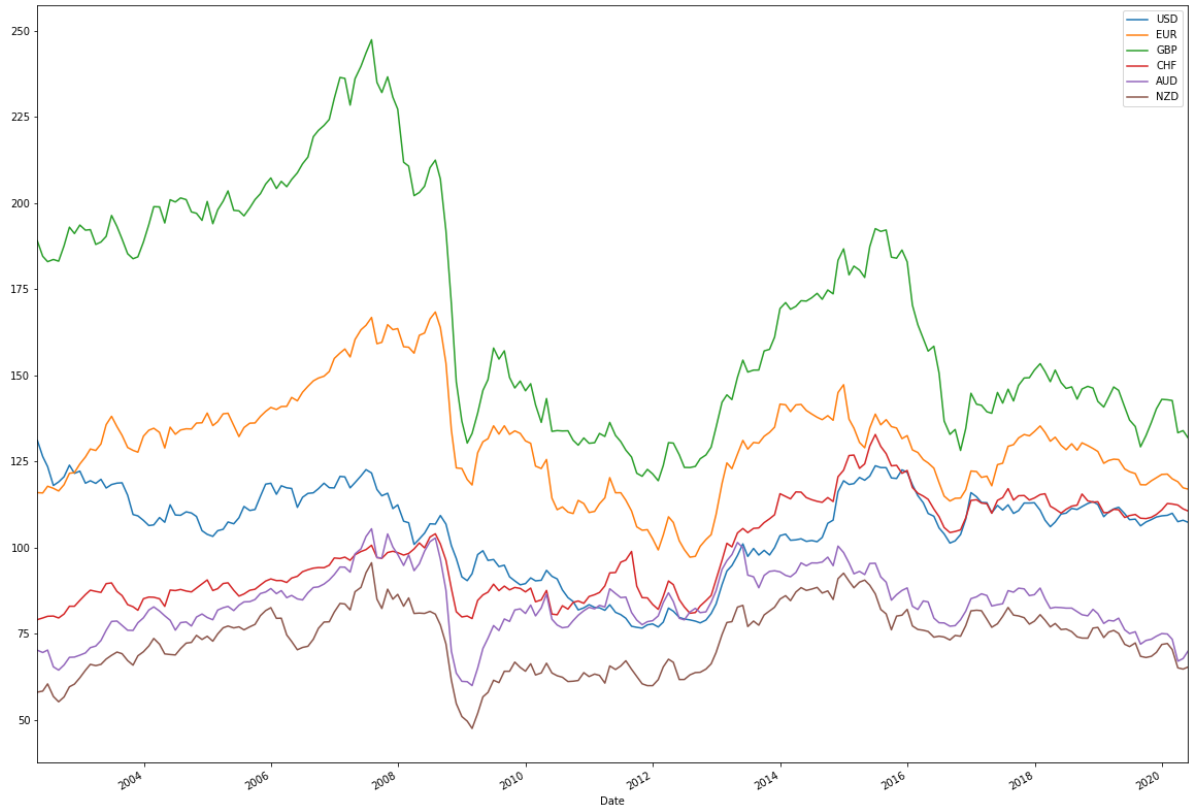
In [8]:
```
fxhm1.head()
```

Out[8]:

Date	USD	EUR	GBP	CHF	AUD	NZD
2002-04-30	131.15	115.97	189.01	79.13	70.24	58.08
2002-05-31	126.44	115.88	184.56	79.60	69.58	58.38
2002-06-28	123.53	117.83	183.00	80.09	70.29	60.48
2002-07-31	118.05	117.23	183.64	80.16	65.42	56.82
2002-08-30	119.08	116.45	183.14	79.60	64.47	55.28

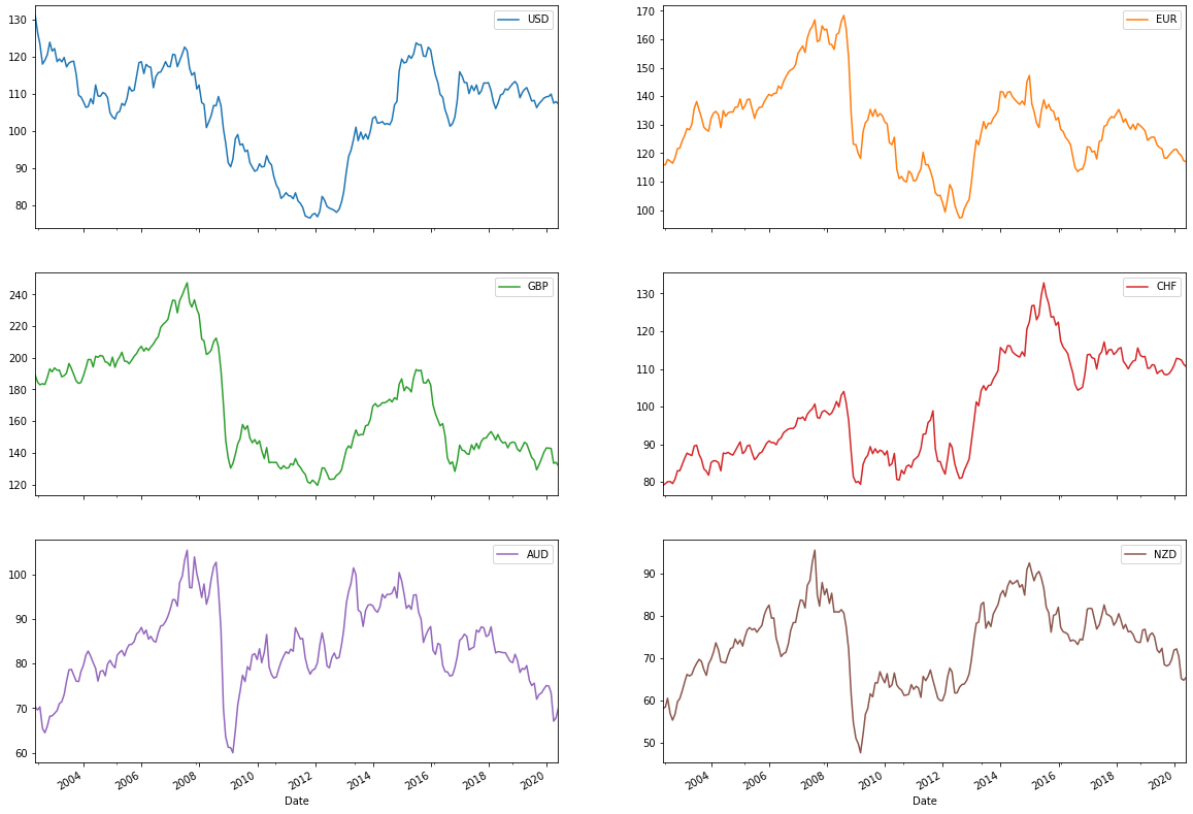
In [9]:
```
%matplotlib inline
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = (20,15) # この設定があれば、全てのplotに提供される
```

In [10]:
```
#plt.figure(figsize=(20,10))
fxhm1.plot()
plt.show()
```



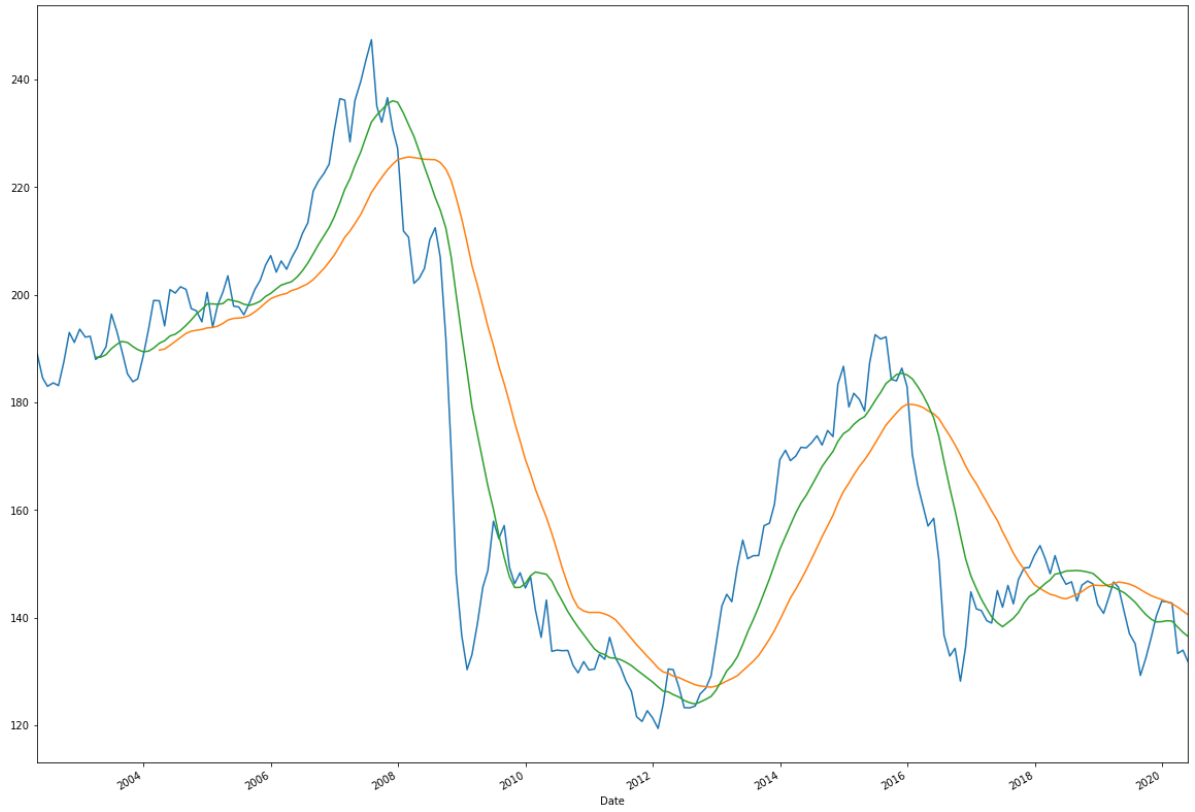
In [11]:
```
fxhm1.plot(subplots=True, layout=(3,2))
```

Out[11]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000002C7334C7128>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002C7334F9DD8>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000002C733534278>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002C7335676D8>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000002C73359AB38>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002C7335CCF98>]],
      dtype=object)
```



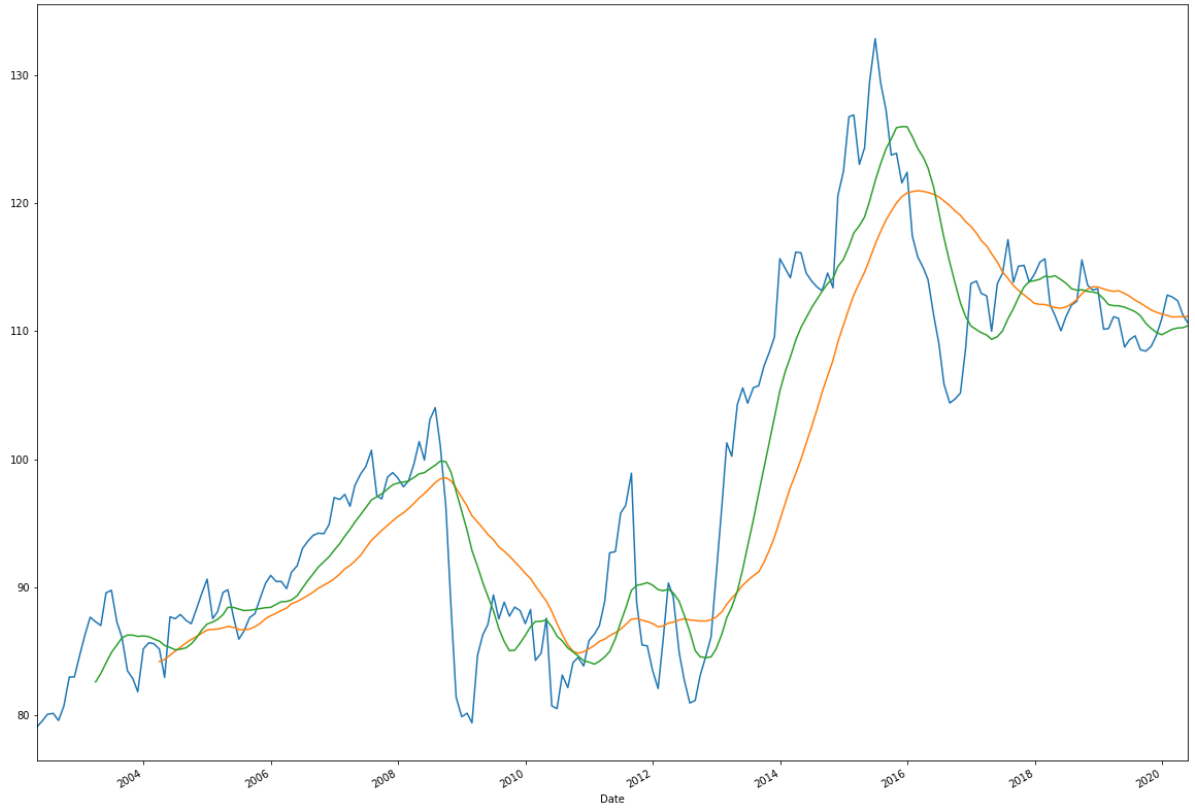
In [12]:
```
fxhm1['GBP'].plot()
fxhm1['GBP'].rolling(24).mean().plot() # 15 months
fxhm1['GBP'].rolling(12).mean().plot()  # 3 months
```

Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x2c7336ece80>



In [13]:
```
fxhm1['CHF'].plot()
fxhm1['CHF'].rolling(24).mean().plot() # 15 months
fxhm1['CHF'].rolling(12).mean().plot()  # 3 months
```

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x2c7337a05c0>



In [32]:
```
#fxhm1.index = fxhm1.index.strftime('%m-%Y')
#fxhm1.head()
```

In [14]:
```
fxhm1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 218 entries, 2002-04-30 to 2020-05-29
Data columns (total 6 columns):
USD    218 non-null float64
EUR    218 non-null float64
GBP    218 non-null float64
CHF    218 non-null float64
AUD    218 non-null float64
NZD    218 non-null float64
dtypes: float64(6)
memory usage: 11.9 KB
```

相関性

In [15]:
```
from __future__ import print_function
import numpy as np
```

In [16]:
```
usd_returns = np.diff(fxhm1['USD']) / fxhm1['USD'][ : -1]
```

In [17]:
```
gbp_returns = np.diff(fxhm1['GBP']) / fxhm1['GBP'][ : -1]
```

In [18]:
```
covariance = np.cov(usd_returns, gbp_returns)
print("Covariance", covariance)
```

```
Covariance [[0.00050726 0.00044505]
 [0.00044505 0.00086129]]
```

In [19]:
```
print("Covariance diagonal", covariance.diagonal())
print("Covariance trace", covariance.trace())
```

```
Covariance diagonal [0.00050726 0.00086129]
Covariance trace 0.0013685555531115514
```

In [20]:
```
print(covariance/ (usd_returns.std() * gbp_returns.std()))
```

```
[[0.76743219 0.67330703]
 [0.67330703 1.30304672]]
```

In [21]:
```
print("Correlation coefficient", np.corrcoef(usd_returns, gbp_returns))
```

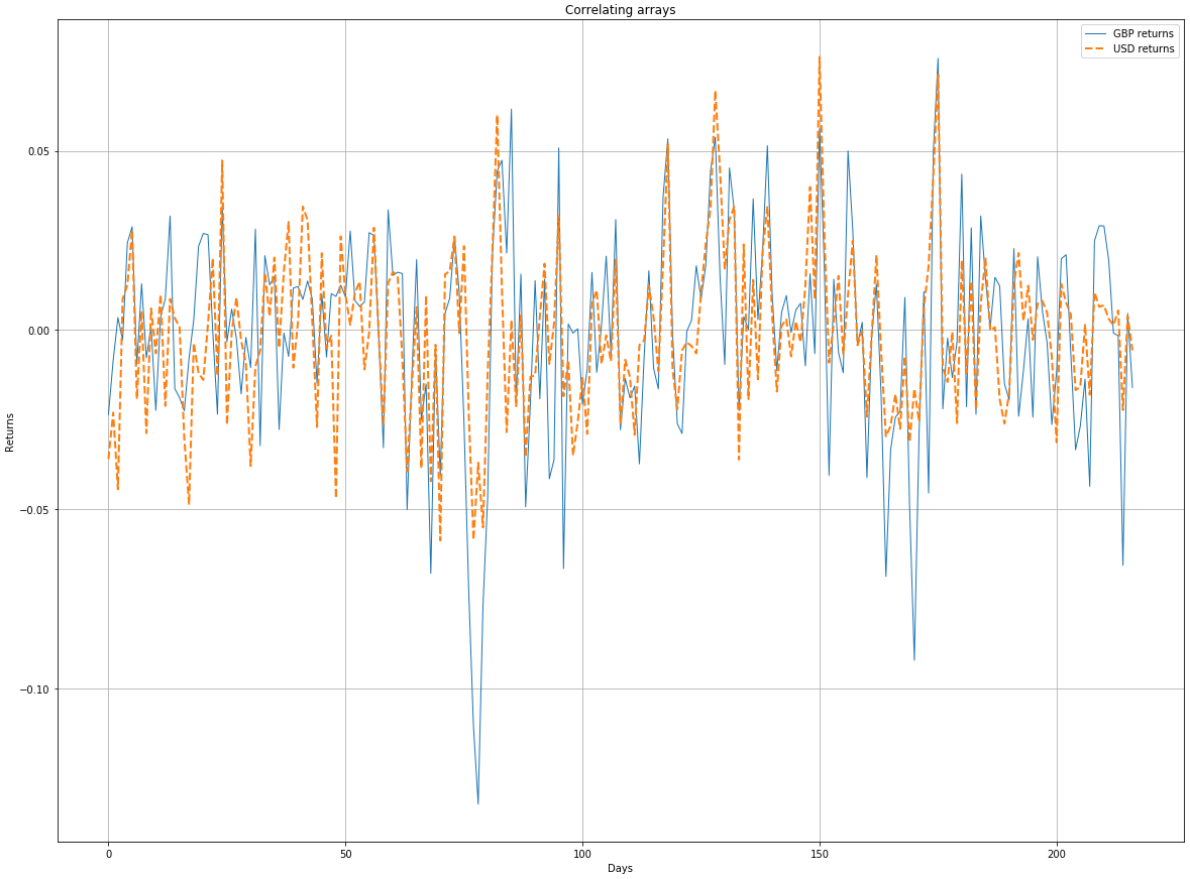
```
Correlation coefficient [[1.         0.67330703]
 [0.67330703 1.        ]]
```

GBPとEURの通貨は円に対して、相関しないようです。相関係数は0.673なので

In [22]:
```
difference = fxhm1['GBP'] - fxhm1['USD']
avg = np.mean(difference)
dev = np.std(difference)

print("Out of sync", np.abs(difference[-1] - avg) > 2 * dev)
```

```
Out of sync False
```

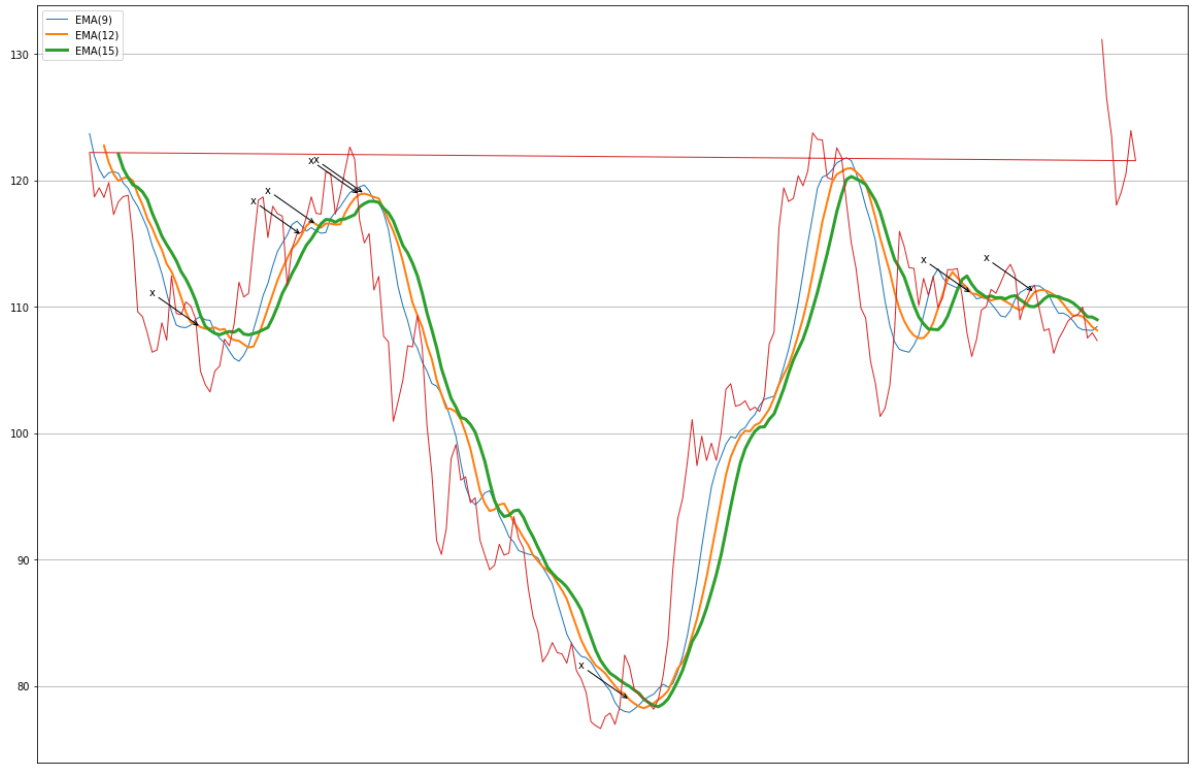
In [23]:
```
t = np.arange(len(gbp_returns))
plt.plot(t, gbp_returns, lw=1, label='GBP returns')
plt.plot(t, usd_returns, '--', lw=2, label='USD returns')
plt.title('Correlating arrays')
plt.xlabel('Days')
plt.ylabel('Returns')
plt.grid()
plt.legend(loc='best')
plt.show()
```



平滑化　(Exponential Smoothing with 9, 12, and 15 months)

In [39]:

```
from matplotlib.dates import DateFormatter
from matplotlib.dates import DayLocator
from matplotlib.dates import MonthLocator
from datetime import date

dates = fxhm['Date']
#dates=pd.to_datetime(dates)
close = fxhm['USD']

fig = plt.figure()
ax = fig.add_subplot(111)

emas = []
for i in range(9, 18, 3):
    weights = np.exp(np.linspace(-1., 0., i))
    weights /= weights.sum()

    ema = np.convolve(weights, close)[i-1:-i+1]
    idx = (i - 6)/3
    ax.plot(dates[i-1:], ema, lw=idx, label="EMA(%s)" % (i))
    data = np.column_stack((dates[i-1:], ema))
    emas.append(np.rec.fromrecords(data, names=["dates", "ema"]))

# emaの交点を自動検出
first = emas[0]["ema"].flatten()
second = emas[2]["ema"].flatten()
bools = np.abs(first[-len(second):] - second)/second < 0.001 #0.0001
xpoints = np.compress(bools, emas[1])

for xpoint in xpoints:
    ax.annotate('x', xy=xpoint, textcoords='offset points',
                xytext=(-50, 30),
                arrowprops=dict(arrowstyle="->"))

leg = ax.legend(loc='best', fancybox=True)

alldays = DayLocator()
months = MonthLocator()
month_formatter = DateFormatter("%b %Y")
ax.plot(dates, close, lw=1.0, label="Close")
ax.xaxis.set_major_locator(months)
ax.xaxis.set_minor_locator(alldays)
ax.xaxis.set_major_formatter(month_formatter)
ax.grid(True)
fig.autofmt_xdate()
plt.show()
```



In [25]:
```
xpoints
```

Out[25]:
```
rec.array([],
          dtype=[('dates', '<U10'), ('ema', '<f8')])
```

In [26]:
```
len(first)
```

Out[26]:
```
210
```

In [27]:
```
len(second)
```

Out[27]:
```
207
```

In [28]:
```
bools = np.abs(first[-len(second):] - second)/second < 0.0001
#xpoints = np.compress(bools, emas[1])
len(bools)
```

Out[28]:
```
207
```

In [29]:
```
bools[:10]
```

Out[29]:
```
array([False, False, False, False, False, False, False, False, False,
       False])
```

In [40]:
```
#print (bools != False) #
```

In [ ]:
```

```