# データ可視化のためのMatplotlib

## 外為ユーロ円のテクニカル分析

Yahooサイトから日次の[OHLC, Adj Close]の価格による分析を行います。

In [7]:
```
#!pip install pandas_datareader
import matplotlib.pyplot as plt
import  pandas_datareader as web
import matplotlib.dates as mdates
import datetime as dt
```

In [2]:
```
eurjpy_day = web.DataReader("EURJPY=X", 'yahoo')
eurjpy_day.head()
```

Out[2]:

| Date | High | Low | Open | Close | Volume | Adj Close |
|---|---|---|---|---|---|---|
| **2015-06-30** | 137.339996 | 136.065994 | 136.272995 | 136.259003 | 0.0 | 136.259003 |
| **2015-07-01** | 136.873001 | 136.102005 | 136.102005 | 136.108994 | 0.0 | 136.108994 |
| **2015-07-02** | 136.835999 | 136.061005 | 136.389008 | 136.406998 | 0.0 | 136.406998 |
| **2015-07-05** | 135.970001 | 134.600998 | 134.600998 | 134.567993 | 0.0 | 134.567993 |
| **2015-07-06** | 135.660004 | 133.511993 | 135.479996 | 135.460007 | 0.0 | 135.460007 |

In [3]:
```
eurjpy_day.tail()
```

Out[3]:

| Date | High | Low | Open | Close | Volume | Adj Close |
|---|---|---|---|---|---|---|
| **2020-06-23** | 120.703003 | 120.050003 | 120.386002 | 120.401001 | 0.0 | 120.401001 |
| **2020-06-24** | 120.607002 | 120.139999 | 120.420998 | 120.429001 | 0.0 | 120.429001 |
| **2020-06-25** | 120.289001 | 119.820000 | 120.211998 | 120.199997 | 0.0 | 120.199997 |
| **2020-06-28** | 121.348999 | 120.230003 | 120.339996 | 120.328003 | 0.0 | 120.328003 |
| **2020-06-30** | 121.176003 | 120.830002 | 120.839996 | 121.000000 | 0.0 | 121.000000 |

## 移動平均線ゴールデンクロスとデードクロス (トレンドリバーサル)

## 移動平均線100日,200日

In [4]:
```
sma200 = eurjpy_day["Adj Close"].rolling(200).mean()
sma100 = eurjpy_day["Adj Close"].rolling(100).mean()
print(sma200["2017-01"].head(10),sma100["2017-01"].head(10))
```

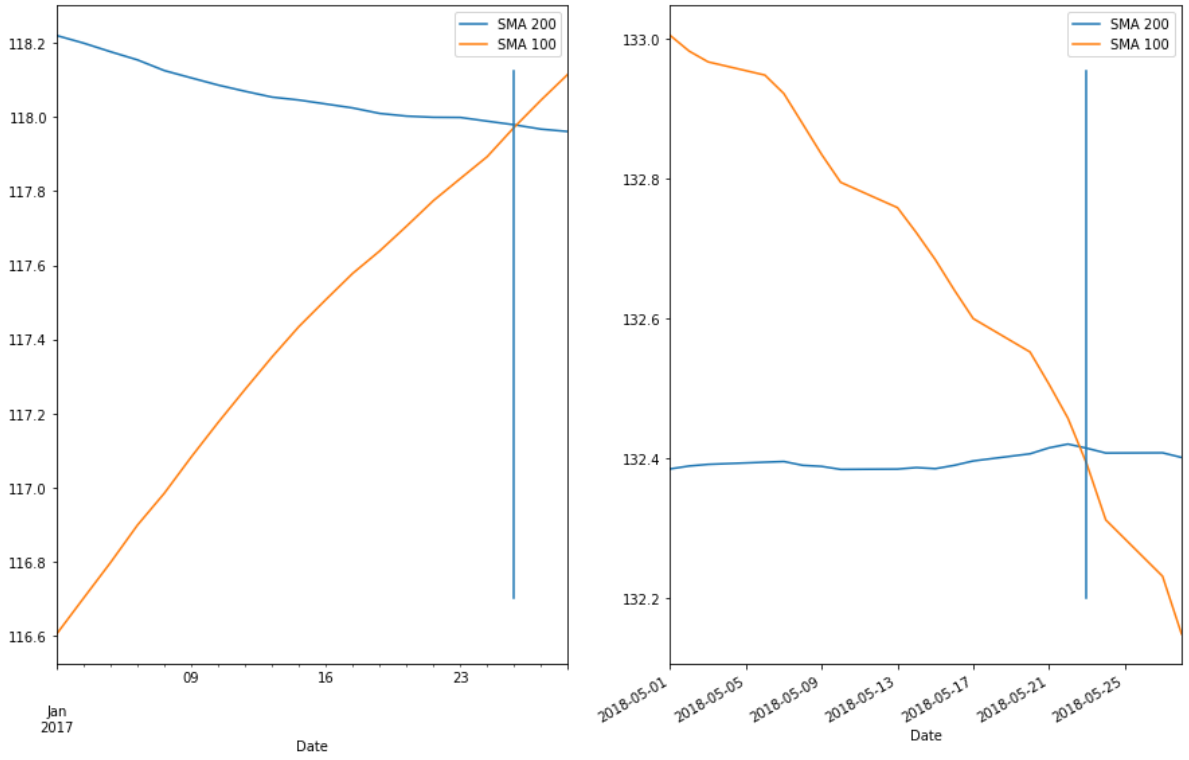
```
Date
2017-01-02    118.219550
2017-01-03    118.199085
2017-01-04    118.175885
2017-01-05    118.153895
2017-01-06    118.124995
2017-01-09    118.105320
2017-01-10    118.085880
2017-01-11    118.069155
2017-01-12    118.053480
2017-01-13    118.045595
Name: Adj Close, dtype: float64 Date
2017-01-02    116.60567
2017-01-03    116.70233
2017-01-04    116.79821
2017-01-05    116.89902
2017-01-06    116.98520
2017-01-09    117.08288
2017-01-10    117.17610
2017-01-11    117.26527
2017-01-12    117.35245
2017-01-13    117.43454
Name: Adj Close, dtype: float64
```

In [5]:
```
print(sma200["2018-05"].head(10),sma100["2018-05"].head(10))
```

```
Date
2018-05-01    132.384405
2018-05-02    132.388555
2018-05-03    132.390900
2018-05-06    132.394025
2018-05-07    132.394990
2018-05-08    132.389505
2018-05-09    132.388000
2018-05-10    132.383705
2018-05-13    132.384170
2018-05-14    132.386325
Name: Adj Close, dtype: float64 Date
2018-05-01    133.00467
2018-05-02    132.98234
2018-05-03    132.96693
2018-05-06    132.94803
2018-05-07    132.92156
2018-05-08    132.87796
2018-05-09    132.83424
2018-05-10    132.79458
2018-05-13    132.75834
2018-05-14    132.72235
Name: Adj Close, dtype: float64
```

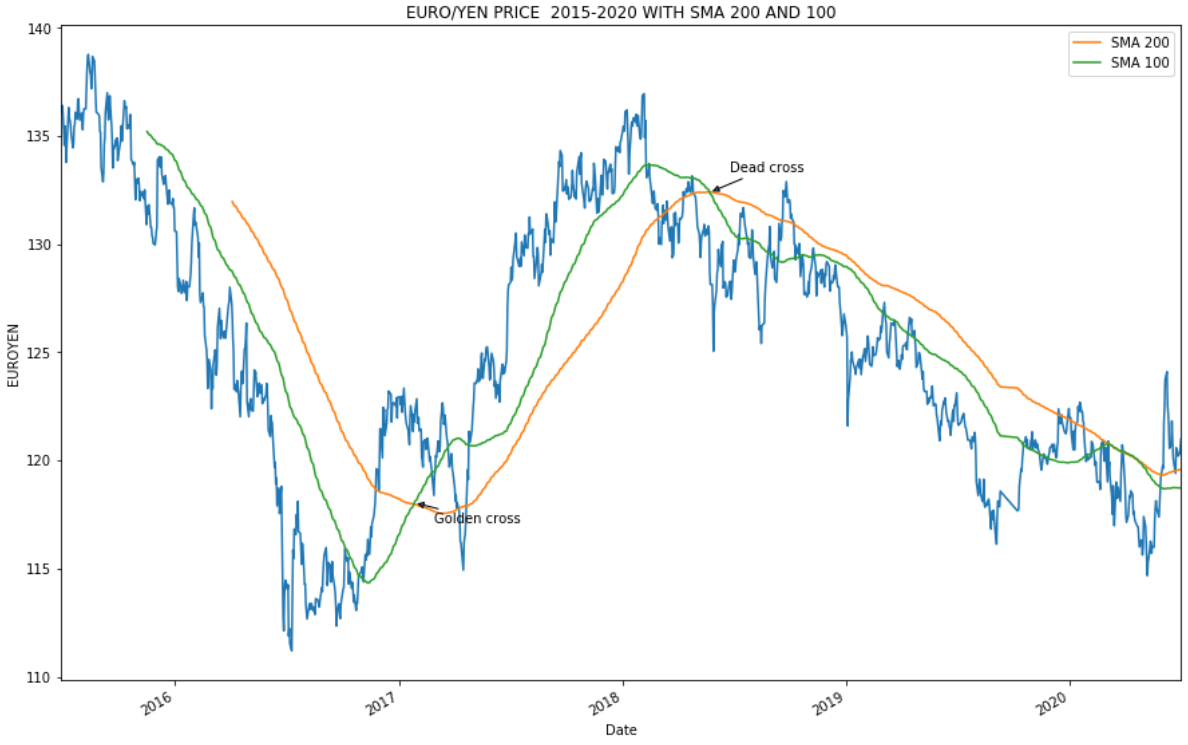
In [8]:
```
plt.figure(figsize=(15,10))
plt.subplot(121)
sma200["2017-01"].head(20).plot(label="SMA 200")
sma100["2017-01"].head(20).plot(label="SMA 100")
plt.axvline(dt.datetime(2017,1,25),ymin=0.1, ymax=0.9)
plt.legend()
plt.subplot(122)
sma200["2018-05"].head(20).plot(label="SMA 200")
sma100["2018-05"].head(20).plot(label="SMA 100")
plt.axvline(dt.datetime(2018,5,23),ymin=0.1, ymax=0.9)
plt.legend()
```

Out[8]: <matplotlib.legend.Legend at 0x2aee1d82e80>



In [9]:
```
plt.figure(figsize=(15,10))
plt.title("EURO/YEN PRICE  2015-2020 WITH SMA 200 AND 100")
ax = eurjpy_day["Adj Close"].plot()
plt.ylabel("EUROYEN")
eurjpy_day["Adj Close"].rolling(200).mean().plot()
eurjpy_day["Adj Close"].rolling(100).mean().plot()
lines, labels = ax.get_legend_handles_labels()
ax.legend(lines[1:],["SMA 200","SMA 100"], loc='best')
ax.annotate("Golden cross", (mdates.date2num(dt.datetime(2017,1,26)),sma100["2017-01-26"]),
            xytext=(15, -15), textcoords='offset points',
            arrowprops=dict(arrowstyle='-|>'))
ax.annotate("Dead cross", (mdates.date2num(dt.datetime(2018,5,23)),sma100["2018-05-23"]),
            xytext=(15, 15), textcoords='offset points',
            arrowprops=dict(arrowstyle='-|>'))
```

Out[9]:
```
Text(15, 15, 'Dead cross')
```




In [10]:
```
eurjpy_day.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1283 entries, 2015-06-24 to 2020-06-24
Data columns (total 6 columns):
High         1283 non-null float64
Low          1283 non-null float64
Open         1283 non-null float64
Close        1283 non-null float64
Volume       1283 non-null float64
Adj Close    1283 non-null float64
dtypes: float64(6)
memory usage: 110.2 KB
```

## Resampling

In [23]:
```
import pandas as pd
```

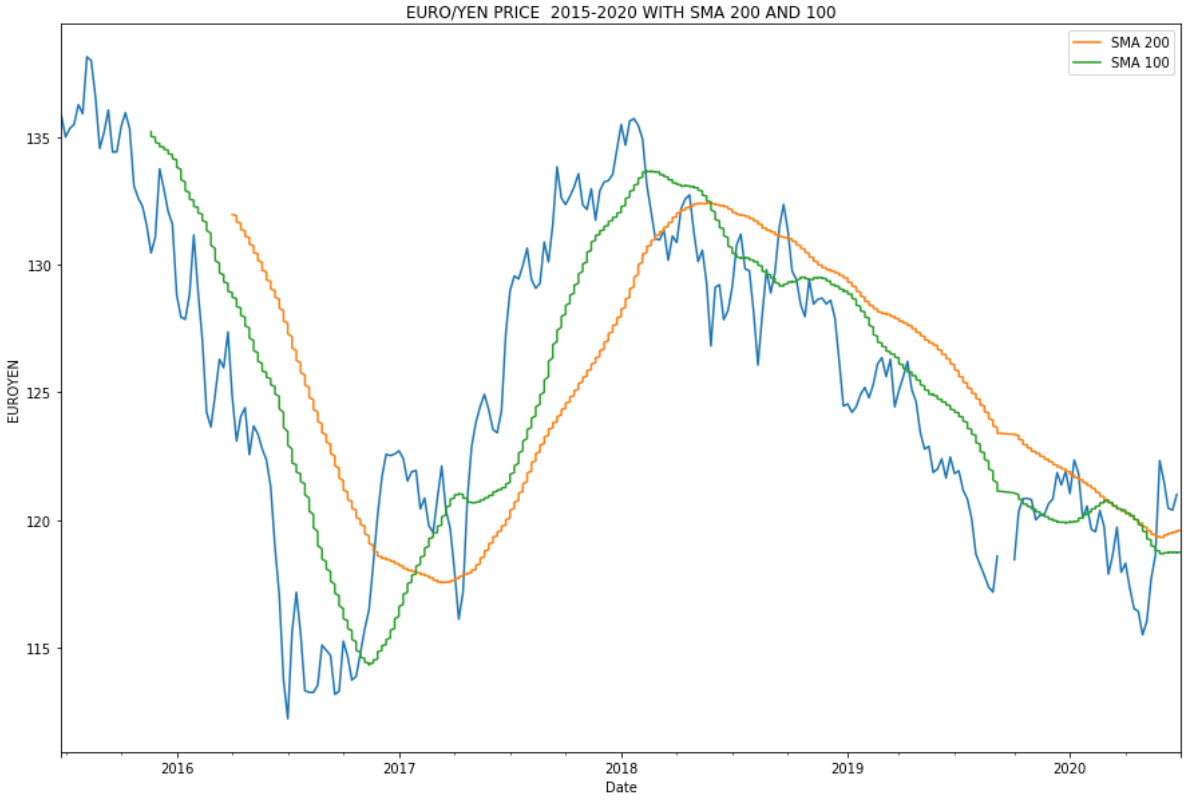
In [27]:
```
eurjpy_week=eurjpy_day.resample('W',label='left', loffset=pd.DateOffset(days=1)).mean()
eurjpy_week.tail()
```

Out[27]:

| Date | High | Low | Open | Close | Volume | Adj Close |
|---|---|---|---|---|---|---|
| **2020-06-01** | 123.287602 | 121.739799 | 122.288199 | 122.329399 | 0.0 | 122.329399 |
| **2020-06-08** | 121.961002 | 120.944598 | 121.536200 | 121.516199 | 0.0 | 121.516199 |
| **2020-06-15** | 120.883400 | 119.856599 | 120.437199 | 120.463000 | 0.0 | 120.463000 |
| **2020-06-22** | 120.804601 | 120.035001 | 120.379199 | 120.388200 | 0.0 | 120.388200 |
| **2020-06-29** | 121.176003 | 120.830002 | 120.839996 | 121.000000 | 0.0 | 121.000000 |

In [29]:
```
plt.figure(figsize=(15,10))
plt.title("EURO/YEN WEEKLY PRICE  2015-2020 WITH SMA 200 AND 100")
ax = eurjpy_week["Adj Close"].plot()
plt.ylabel("EUROYEN")
eurjpy_day["Adj Close"].rolling(200).mean().plot()
eurjpy_day["Adj Close"].rolling(100).mean().plot()
lines, labels = ax.get_legend_handles_labels()
ax.legend(lines[1:],["SMA 200","SMA 100"], loc='best')
#ax.annotate("Golden cross", (mdates.date2num(dt.datetime(2017,1,26)),sma100["2017-01-26"]),
#             xytext=(15, -15), textcoords='offset points',
#             arrowprops=dict(arrowstyle='-|>'))
#ax.annotate("Dead cross", (mdates.date2num(dt.datetime(2018,5,23)),sma100["2018-05-23"]),
#             xytext=(15, 15), textcoords='offset points',
#             arrowprops=dict(arrowstyle='-|>'))
```

Out[29]: <matplotlib.legend.Legend at 0x2aee4d67860>



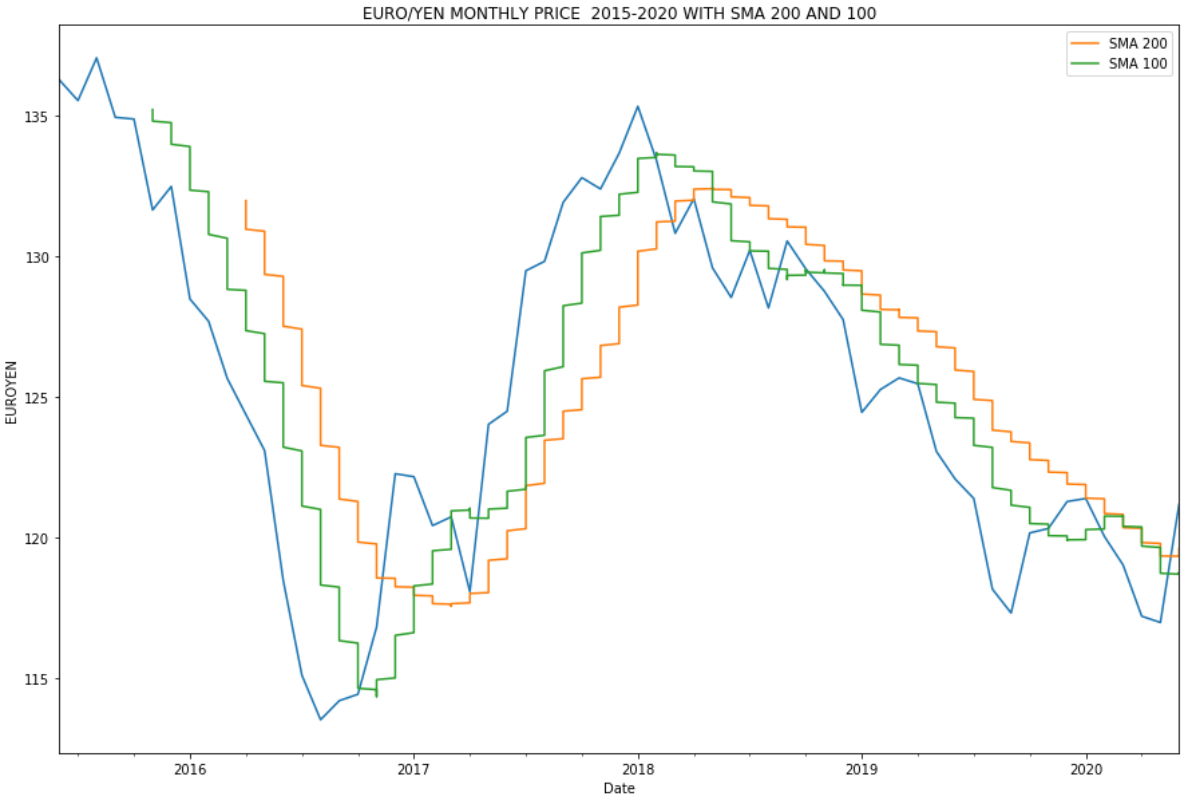
In [30]:
```
eurjpy_month=eurjpy_day.resample('M',label='left', loffset=pd.DateOffset(days=1)).mean()
eurjpy_month.tail()
```

Out[30]:

| Date | High | Low | Open | Close | Volume | Adj Close |
|---|---|---|---|---|---|---|
| **2020-02-01** | 120.408450 | 119.610350 | 120.021352 | 120.025050 | 0.0 | 120.025050 |
| **2020-03-01** | 119.689696 | 118.111218 | 119.009130 | 119.011043 | 0.0 | 119.011043 |
| **2020-04-01** | 117.541409 | 116.731681 | 117.186636 | 117.189273 | 0.0 | 117.189273 |
| **2020-05-01** | 117.387619 | 116.609000 | 116.966810 | 116.966572 | 0.0 | 116.966572 |
| **2020-06-01** | 121.707573 | 120.652857 | 121.144952 | 121.165904 | 0.0 | 121.165904 |

In [32]:
```
plt.figure(figsize=(15,10))
plt.title("EURO/YEN MONTHLY PRICE  2015-2020 WITH SMA 200 AND 100")
ax = eurjpy_month["Adj Close"].plot()
plt.ylabel("EUROYEN")
eurjpy_day["Adj Close"].rolling(200).mean().plot()
eurjpy_day["Adj Close"].rolling(100).mean().plot()
lines, labels = ax.get_legend_handles_labels()
ax.legend(lines[1:],["SMA 200","SMA 100"], loc='best')
```

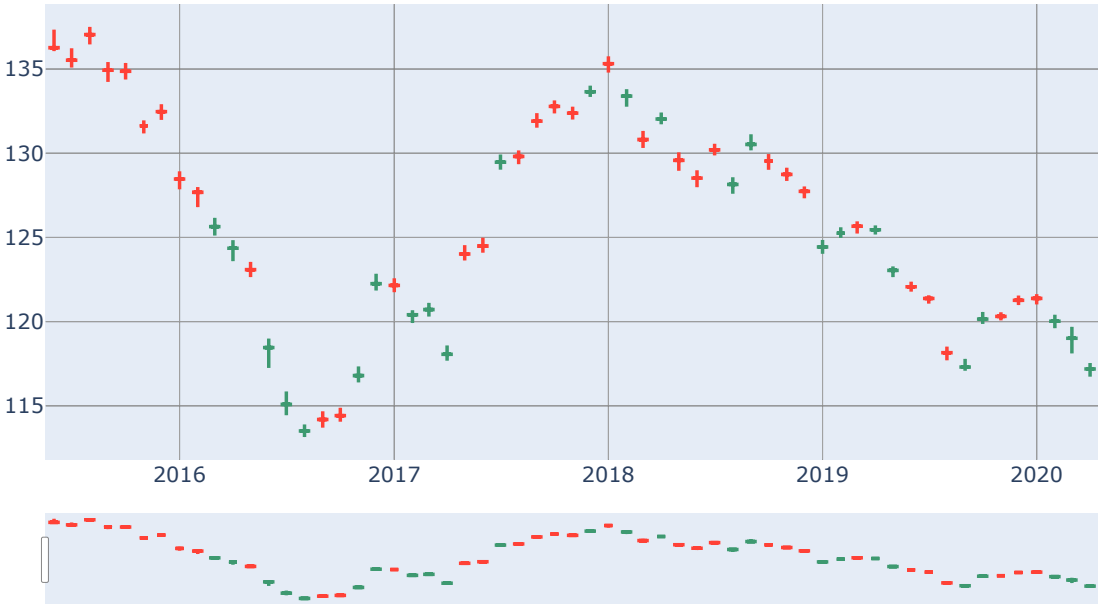
Out[32]: <matplotlib.legend.Legend at 0x2aee5170be0>




## Plotly candlestick

In [33]:
```
import plotly.graph_objects as go
```

In [34]:
```
fig = go.Figure(data=[go.Candlestick(x=eurjpy_month.index,
                open=eurjpy_month['Open'],
                high=eurjpy_month['High'],
                low=eurjpy_month['Low'],
                close=eurjpy_month['Close'])])

fig.show()
```

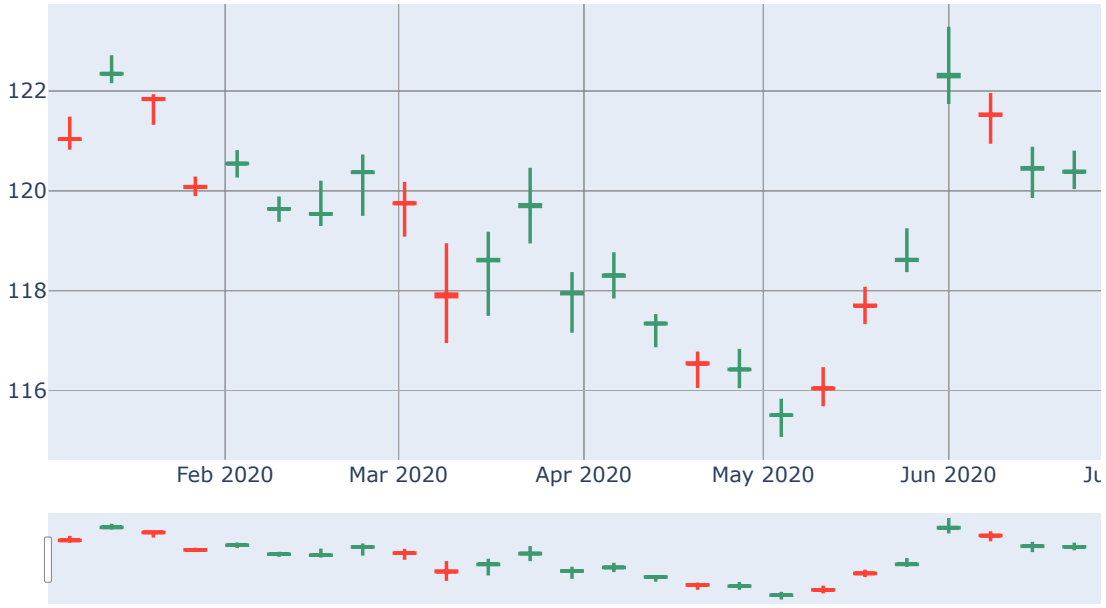



In [35]:
```
df2020=eurjpy_week['2020']
df2020.head()
```

Out[35]:

| Date | High | Low | Open | Close | Volume | Adj Close |
|---|---|---|---|---|---|---|
| **2020-01-06** | 121.487001 | 120.825401 | 121.038799 | 121.033800 | 0.0 | 121.033800 |
| **2020-01-13** | 122.717200 | 122.157201 | 122.344400 | 122.349599 | 0.0 | 122.349599 |
| **2020-01-20** | 121.935199 | 121.323000 | 121.846201 | 121.833998 | 0.0 | 121.833998 |
| **2020-01-27** | 120.286401 | 119.895799 | 120.085600 | 120.071001 | 0.0 | 120.071001 |
| **2020-02-03** | 120.816600 | 120.266200 | 120.545203 | 120.548201 | 0.0 | 120.548201 |

In [36]:
```
fig = go.Figure(data=[go.Candlestick(x=df2020.index,
                open=df2020['Open'],
                high=df2020['High'],
                low=df2020['Low'],
                close=df2020['Close'])])

fig.show()
```




In [37]:
```
df2020MayJune=eurjpy_day['2020-05':]
df2020MayJune.head()
```

Out[37]:

| Date | High | Low | Open | Close | Volume | Adj Close |
|---|---|---|---|---|---|---|
| **2020-05-03** | 116.990997 | 116.495003 | 116.969002 | 116.972000 | 0.0 | 116.972000 |
| **2020-05-04** | 116.510002 | 115.449997 | 116.330002 | 116.331001 | 0.0 | 116.331001 |
| **2020-05-05** | 115.369003 | 114.455002 | 115.369003 | 115.377998 | 0.0 | 115.377998 |
| **2020-05-06** | 115.057999 | 114.592003 | 114.696999 | 114.675003 | 0.0 | 114.675003 |
| **2020-05-07** | 115.727997 | 115.039001 | 115.285004 | 115.263000 | 0.0 | 115.263000 |

In [38]:
```
fig = go.Figure(data=[go.Candlestick(x=df2020MayJune.index,
                open=df2020MayJune['Open'],
                high=df2020MayJune['High'],
                low=df2020MayJune['Low'],
                close=df2020MayJune['Close'])])

fig.show()
```


In [40]:
```
# Yearly resampling using grouper
import numpy as np
```

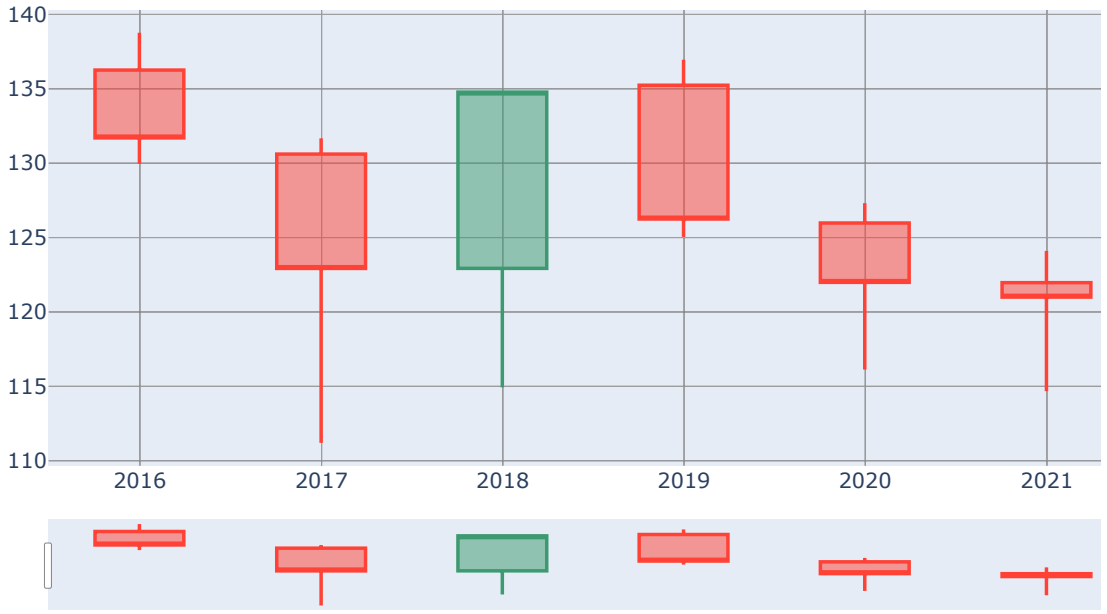
In [41]:
```
# work off a copy of the dataset to preserve the original
ohlc_df = eurjpy_day.copy()

# aggregation frequency
time_grouper = 'Y' # try 'Y', '5Y', 'M', etc

#ohlc_df = ohlc_df.set_index('Date')
tmp_open = ohlc_df.groupby(
    pd.Grouper(freq=time_grouper))['Close'].nth([0])
tmp_close = ohlc_df.groupby(
    pd.Grouper(freq=time_grouper))['Close'].nth([-1])
ohlc_df = ohlc_df.groupby(
    pd.Grouper(freq=time_grouper))['Close'].aggregate(
    {'Low': np.min,'High':np.max}).reset_index()
ohlc_df['Open'] = tmp_open.values
ohlc_df['Close'] = tmp_close.values

ohlc_df = ohlc_df[['Date', 'Open', 'High', 'Low', 'Close']]
ohlc_df.tail()
```

```
D:¥Anaconda3¥lib¥site-packages¥ipykernel_launcher.py:14: FutureWarning:

using a dict on a Series for aggregation
is deprecated and will be removed in a future version. Use                named aggregation inst
ead.

    >>> grouper.agg(name_1=func_1, name_2=func_2)
```

Out[41]:

| | Date | Open | High | Low | Close |
|---|---|---|---|---|---|
| **1** | 2016-12-31 | 130.612000 | 131.669998 | 111.202003 | 122.926003 |
| **2** | 2017-12-31 | 122.930000 | 134.778000 | 114.931999 | 134.778000 |
| **3** | 2018-12-31 | 135.240005 | 136.955002 | 125.045998 | 126.249001 |
| **4** | 2019-12-31 | 125.980003 | 127.309998 | 116.129997 | 121.989998 |
| **5** | 2020-12-31 | 121.970001 | 124.103996 | 114.675003 | 121.000000 |

In [42]:
```
fig = go.Figure(data=[go.Candlestick(x=ohlc_df['Date'],
                open=ohlc_df['Open'],
                high=ohlc_df['High'],
                low=ohlc_df['Low'],
                close=ohlc_df['Close'])])

fig.show()
```




# 日経平均値225の分析

In [15]:
```
n225= web.DataReader("^N225", 'yahoo')
```

In [20]:
```
print(n225[["Adj Close"]].head(),n225[["Adj Close"]].tail())
```

```
              Adj Close
Date
2015-07-02  20522.500000
2015-07-03  20539.789062
2015-07-06  20112.119141
2015-07-07  20376.589844
2015-07-08  19737.640625                Adj Close
Date
2020-06-24  22534.320312
2020-06-25  22259.789062
2020-06-26  22512.080078
2020-06-29  21995.039062
2020-06-30  22422.970703
```

In [22]:
```
n225_sma200 = n225["Adj Close"].rolling(200).mean()
n225_sma100 = n225["Adj Close"].rolling(100).mean()
#print(sma200["2017-01"].head(20), sma100["2017-01"].head(20))
```

In [ ]:
```
```

In [ ]: