Storytelling Data Visualization on Exchange Rates

In my project, I emphasized explanatory data visualization, where I crafted graphs not just for personal exploration but to effectively convey information to others. Key aspects I delved into include applying information design principles such as maximizing the data-ink ratio for enhanced clarity. Additionally, I learned to incorporate storytelling elements into data visualizations using Matplotlib, guiding the audience's attention through pre-attentive attributes. A case study on the FiveThirtyEight style highlighted the utilization of Matplotlib's built-in styles for impactful visual communication.

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
df = pd.read_csv('euro-daily-hist_1999_2022.csv')
```

In [3]:

```
df.head()
```

Out[3]:

	Period\Unit:	[Australian dollar]	[Bulgarian lev]	[Brazilian real]	[Canadian dollar]	[Swiss franc]	[Chinese yuan renminbi]	[Cypriot pound]	[Czech koruna]	[Danish krone]	...	[Romanian leu]	[Russian rouble]	[Swedish krona]
0	2023-12-15	1.6324	1.9558	5.4085	1.4653	0.9488	7.7812	NaN	24.477	7.4556	...	4.9710	NaN	11.2125
1	2023-12-14	1.6288	1.9558	5.3349	1.4677	0.949	7.7866	NaN	24.408	7.4566	...	4.9712	NaN	11.18
2	2023-12-13	1.6452	1.9558	5.3609	1.4644	0.9452	7.7426	NaN	24.476	7.4566	...	4.9738	NaN	11.277
3	2023-12-12	1.6398	1.9558	5.3327	1.4656	0.9443	7.7447	NaN	24.42	7.4569	...	4.9732	NaN	11.2815
4	2023-12-11	1.642	1.9558	5.3169	1.4609	0.9478	7.7206	NaN	24.367	7.4563	...	4.9707	NaN	11.297

5 rows × 41 columns

In [4]:

```
print(f'No. of rows: {df.shape[0]}')
print(f'No. of columns: {df.shape[1]}')
```

```
No. of rows: 6456
No. of columns: 41
```

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6456 entries, 0 to 6455
Data columns (total 41 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Period\Unit:              6456 non-null   object 
 1   [Australian dollar ]      6456 non-null   object 
 2   [Bulgarian lev ]          6054 non-null   object 
 3   [Brazilian real ]         6188 non-null   object 
 4   [Canadian dollar ]        6456 non-null   object 
 5   [Swiss franc ]            6456 non-null   object 
 6   [Chinese yuan renminbi ]  6188 non-null   object 
 7   [Cypriot pound ]          2346 non-null   object 
 8   [Czech koruna ]           6456 non-null   object 
 9   [Danish krone ]           6456 non-null   object 
 10  [Estonian kroon ]         3130 non-null   object 
 11  [UK pound sterling ]      6456 non-null   object 
 12  [Greek drachma ]          520 non-null    object 
 13  [Hong Kong dollar ]       6456 non-null   object 
 14  [Croatian kuna ]          5941 non-null   object 
 15  [Hungarian forint ]       6456 non-null   object 
 16  [Indonesian rupiah ]      6456 non-null   object 
 17  [Israeli shekel ]         6188 non-null   object 
 18  [Indian rupee ]           6188 non-null   object 
 19  [Iceland krona ]          4049 non-null   float64
 20  [Japanese yen ]           6456 non-null   object 
 21  [Korean won ]             6456 non-null   object 
 22  [Lithuanian litas ]       4159 non-null   object 
 23  [Latvian lats ]           3904 non-null   object 
 24  [Maltese lira ]           2346 non-null   object 
 25  [Mexican peso ]           6456 non-null   object 
 26  [Malaysian ringgit ]      6456 non-null   object 
 27  [Norwegian krone ]        6456 non-null   object 
 28  [New Zealand dollar ]     6456 non-null   object 
 29  [Philippine peso ]        6456 non-null   object 
 30  [Polish zloty ]           6456 non-null   object 
 31  [Romanian leu ]           6394 non-null   float64
 32  [Russian rouble ]         5994 non-null   object 
 33  [Swedish krona ]          6456 non-null   object 
 34  [Singapore dollar ]       6456 non-null   object 
 35  [Slovenian tolar ]        2085 non-null   object 
 36  [Slovak koruna ]          2608 non-null   object 
 37  [Thai baht ]              6456 non-null   object 
 38  [Turkish lira ]           6394 non-null   float64
 39  [US dollar ]              6456 non-null   object 
 40  [South African rand ]     6456 non-null   object 
dtypes: float64(3), object(38)
memory usage: 2.0+ MB
```

In [6]:

```
df.rename(columns = {'[US dollar ]':'US_dollar' , 'Period\\Unit:':'Time'}, inplace = True)
```

In [7]:

```
df['Time'] = pd.to_datetime(df['Time'])
```

In [8]:

```
df.sort_values('Time' , inplace = True)
```

In [9]:

```
df.head()
```

Out[9]:

	Time	[Australian dollar]	[Bulgarian lev]	[Brazilian real]	[Canadian dollar]	[Swiss franc]	[Chinese yuan renminbi]	[Cypriot pound]	[Czech koruna]	[Danish krone]	...	[Romanian leu]	[Russian rouble]	[Swedish krona]	[
6455	1999-01-04	1.9100	NaN	NaN	1.8004	1.6168	NaN	0.58231	35.107	7.4501	...	1.3111	25.2875	9.4696	
6454	1999-01-05	1.8944	NaN	NaN	1.7965	1.6123	NaN	0.58230	34.917	7.4495	...	1.3168	26.5876	9.4025	
6453	1999-01-06	1.8820	NaN	NaN	1.7711	1.6116	NaN	0.58200	34.850	7.4452	...	1.3168	27.4315	9.3050	
6452	1999-01-07	1.8474	NaN	NaN	1.7602	1.6165	NaN	0.58187	34.886	7.4431	...	1.3092	26.9876	9.1800	
6451	1999-01-08	1.8406	NaN	NaN	1.7643	1.6138	NaN	0.58187	34.938	7.4433	...	1.3143	27.2075	9.1650	

5 rows × 41 columns

In [10]:

```
df2 = df[['Time' , 'US_dollar']].copy()
```

In [11]:

```
df2.head()
```

Out[11]:

	Time	US_dollar
6455	1999-01-04	1.1789
6454	1999-01-05	1.1790
6453	1999-01-06	1.1743
6452	1999-01-07	1.1632
6451	1999-01-08	1.1659

In [12]:

```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6456 entries, 6455 to 0
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype         
---  ------     --------------  -----         
 0   Time       6456 non-null   datetime64[ns]
 1   US_dollar  6456 non-null   object        
dtypes: datetime64[ns](1), object(1)
memory usage: 151.3+ KB
```

In [13]:

```
df2['US_dollar'].value_counts()
```

Out[13]:

```
-         62
1.2276     9
1.1215     8
1.0888     7
1.0868     7
          ..
1.4304     1
1.4350     1
1.4442     1
1.4389     1
1.0804     1
Name: US_dollar, Length: 3769, dtype: int64
```

In [14]:

```
df2.drop(df2['US_dollar'][df2['US_dollar'] == '-'].index, inplace = True)
```

In [15]:

```
df2.shape
```

Out[15]:

```
(6394, 2)
```

In [16]:

```
df2['US_dollar'] = df2['US_dollar'].astype(float)
df2['US_dollar']
```

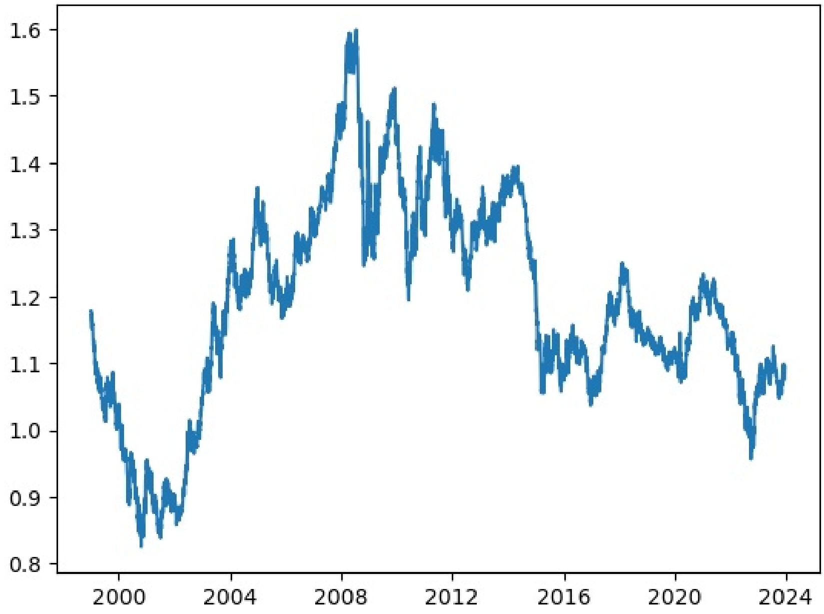
Out[16]:

```
6455    1.1789
6454    1.1790
6453    1.1743
6452    1.1632
6451    1.1659
         ...  
4       1.0757
3       1.0804
2       1.0787
1       1.0919
0       1.0946
Name: US_dollar, Length: 6394, dtype: float64
```

Rolling Mean

In [17]:

```
plt.plot(df2['Time'] , df2['US_dollar'])
plt.show()
```



When examining the shape of the line, we observe numerous small fluctuations rather than a smooth curve. These fluctuations, however, carry significance as they visually depict the daily variations in the exchange rate—alternating between upward and downward movements each day. Notably, the rate demonstrates distinct upward or downward trends over more extended periods, such as months or years.

Depending on our objectives, it may be undesirable to display this daily variation in our graph. In such cases, if our intention is to emphasize only the long-term trends, we can employ the rolling mean, also referred to as the moving average, to smooth out the graph.

In [18]:

```
values = pd.DataFrame()
values['daily_values'] = pd.Series(range(1,20,2))
values
```

Out[18]:

	daily_values
0	1
1	3
2	5
3	7
4	9
5	11
6	13
7	15
8	17
9	19

In [19]:

```
values['rolling_mean_2'] = values['daily_values'].rolling(2).mean()
values
```

Out[19]:

	daily_values	rolling_mean_2
0	1	NaN
1	3	2.0
2	5	4.0
3	7	6.0
4	9	8.0
5	11	10.0
6	13	12.0
7	15	14.0
8	17	16.0
9	19	18.0

In [20]:

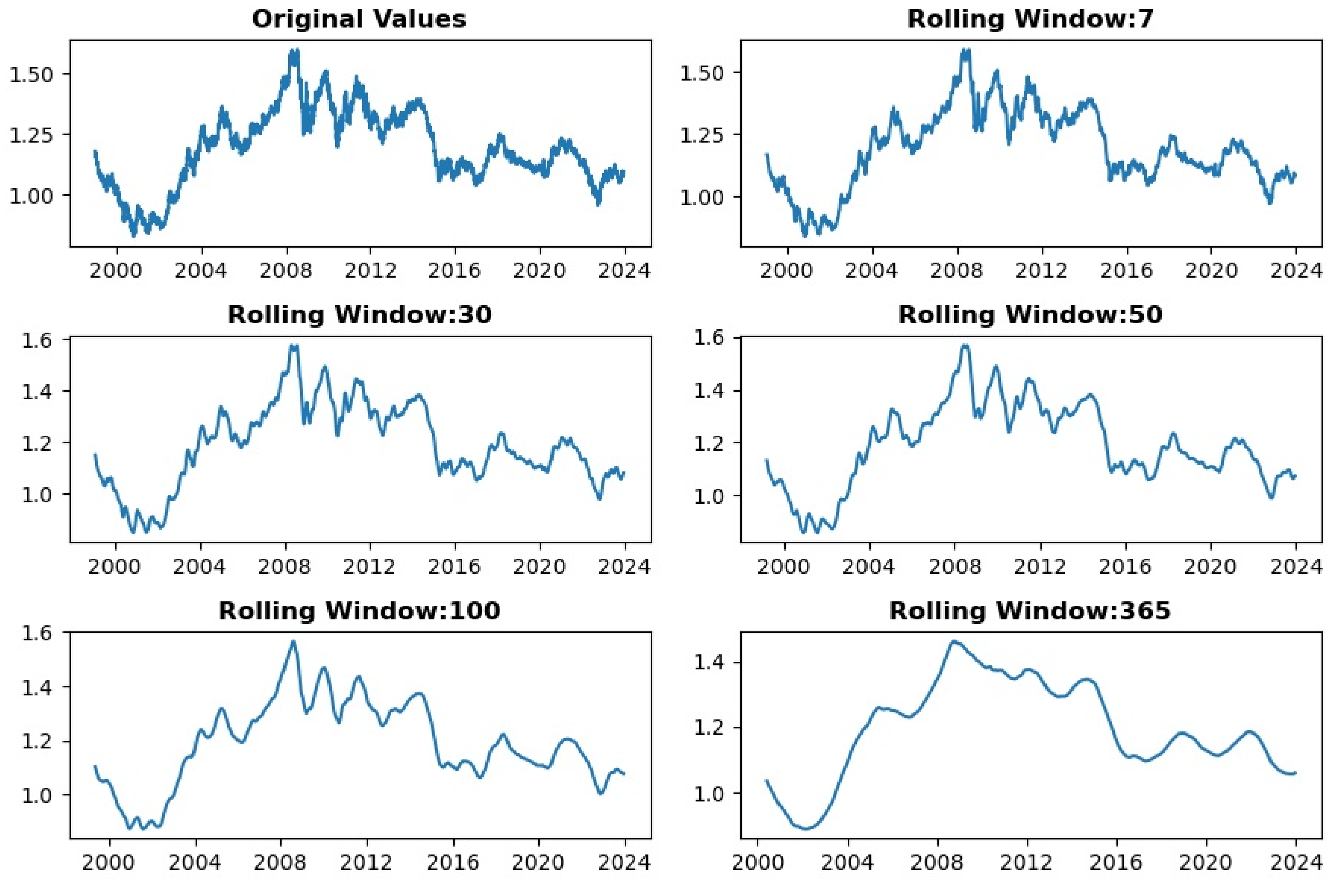
```
values['rolling_mean_3'] = values['daily_values'].rolling(3).mean()
values['rolling_mean_5'] = values['daily_values'].rolling(5).mean()

values
```

Out[20]:

	daily_values	rolling_mean_2	rolling_mean_3	rolling_mean_5
0	1	NaN	NaN	NaN
1	3	2.0	NaN	NaN
2	5	4.0	3.0	NaN
3	7	6.0	5.0	NaN
4	9	8.0	7.0	5.0
5	11	10.0	9.0	7.0
6	13	12.0	11.0	9.0
7	15	14.0	13.0	11.0
8	17	16.0	15.0	13.0
9	19	18.0	17.0	15.0

In [22]:

```
plt.figure(figsize=(9,6))


plt.subplot(3,2,1)
plt.plot(df2['Time'] , df2['US_dollar'])
plt.title('Original Values', weight = 'bold')

for i, rolling_mean in zip([2,3,4,5,6],
                           [7, 30, 50, 100, 365]):
    plt.subplot(3,2,i)
    plt.plot(df2['Time'] , df2['US_dollar'].rolling(rolling_mean).mean())
    plt.title('Rolling Window:' + str(rolling_mean), weight = 'bold')

plt.tight_layout()
plt.show()
```



Coming up with an idea

Here, are a few story ideas for our data:

- Demonstrate the euro-dollar rate fluctuations during the COVID-19 pandemic by presenting the 2020 data alongside the 2016-2019 baseline, utilizing a line plot.
- Explore the euro-dollar rate dynamics during the 2007-2008 financial crisis, including comparative data for 2006 and 2009, visualized through a line plot.
- Provide a comparative analysis of the euro-dollar rate changes under the last three US presidencies (George W. Bush: 2001-2009, Barack Obama: 2009-2017, and Donald Trump: 2017-2021) using a line plot to illustrate these trends.

In [27]:

```
df2['rolling_mean'] = df2['US_dollar'].rolling(30).mean()
df2
```

Out[27]:

	Time	US_dollar	rolling_mean
6455	1999-01-04	1.1789	NaN
6454	1999-01-05	1.1790	NaN
6453	1999-01-06	1.1743	NaN
6452	1999-01-07	1.1632	NaN
6451	1999-01-08	1.1659	NaN
...	...	...	...
4	2023-12-11	1.0757	1.080143
3	2023-12-12	1.0804	1.080760
2	2023-12-13	1.0787	1.081593
1	2023-12-14	1.0919	1.082453
0	2023-12-15	1.0946	1.083267

6394 rows × 3 columns

Storytelling Data Visualization

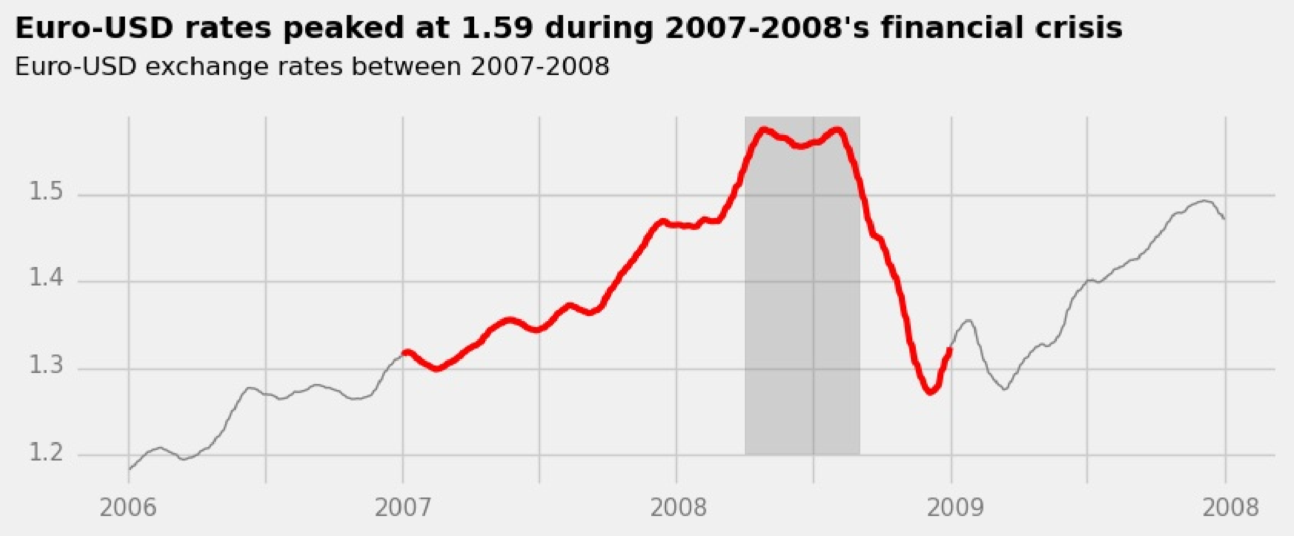
`Financial Crisis 2007-2008`

In [46]:

```
financial_crisis = df2.copy()[(df2['Time'].dt.year >= 2006) & (df2['Time'].dt.year <= 2009)]
financial_crisis_7_8 = df2.copy()[(df2['Time'].dt.year >= 2007) & (df2['Time'].dt.year <= 2008)]
```

In [64]:

```
import matplotlib.style as style
style.use('fivethirtyeight')

#Adding plots
fig,ax = plt.subplots(figsize = (9,3))
ax.plot(financial_crisis['Time'],
        financial_crisis['rolling_mean'],
        linewidth=1 , color = 'grey')

#Highlighting the 2007-2008 crises
ax.plot(financial_crisis_7_8['Time'],
        financial_crisis_7_8['rolling_mean'],
        linewidth=3 , color = 'red')

ax.set_xticklabels([])

x= 0.02
for year in ['2006', '2007', '2008', '2009', '2008']:
    ax.text(x, -0.08, year, alpha = 0.5, fontsize = 11, transform = plt.gca().transAxes)
    x += 0.22888

ax.set_yticklabels([])

y=0.07
for rate in ['1.2', '1.3', '1.4', '1.5']:
    ax.text(-.04, y, rate, alpha = 0.5, fontsize = 11, transform = plt.gca().transAxes)
    y += 0.2333


# Adding title and subtitle
ax.text(-.05, 1.2, "Euro-USD rates peaked at 1.59 during 2007-2008's financial crisis",
       weight = 'bold', transform = plt.gca().transAxes)
ax.text(-.05, 1.1, "Euro-USD exchange rates between 2007-2008",
       size = 12, transform = plt.gca().transAxes)

ax.axvspan(xmin = pd.to_datetime('2008-04-1'), xmax = pd.to_datetime('2008-09-1'), ymin = 0.09,
          alpha = 0.3 , color = "grey")


plt.show()
```



Observations:

- The highest point on the graph corresponds to the peak during the 2007-2008 financial crisis.
- At this peak, the Euro-USD exchange rate reached approximately `1.59`.
- Notably, there was a significant peak during the 2007-2008 financial crisis.
- Following that peak, there was a sharp decline in the exchange rate.
- The lowest point on the graph represents the time when the Euro was weakest against the USD.
- The exchange rate dropped significantly during this period.

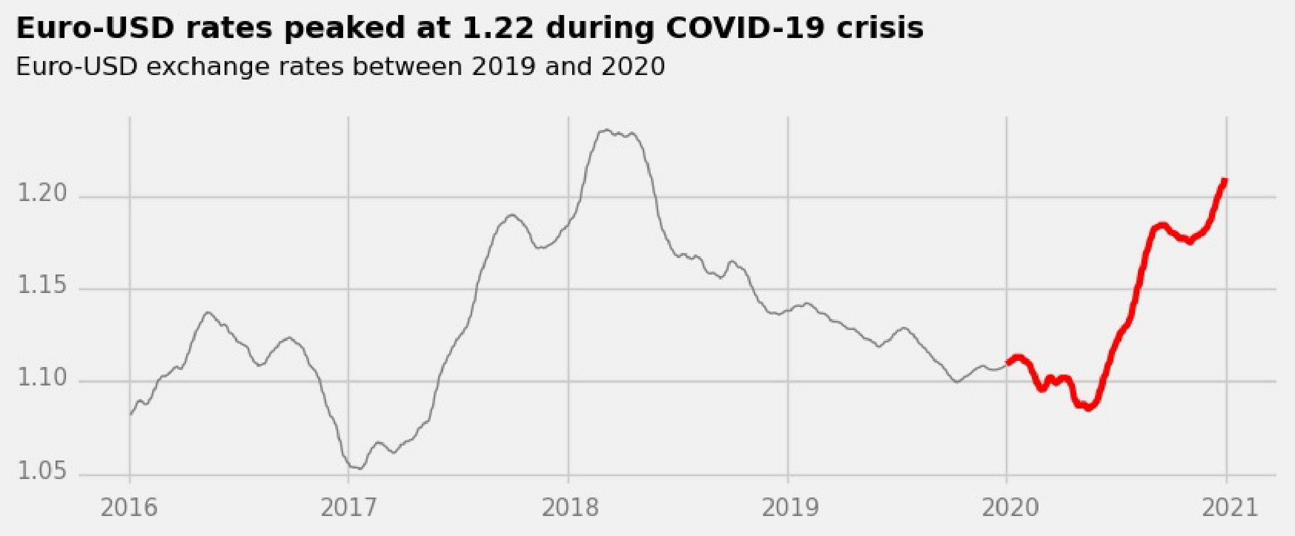
`Covid-19`

In [84]:

```
corona_crisis_20 = df2.loc[(df2['Time'] >=  '2020-01-01') & (df2['Time'] <=  '2020-12-31' )]
corona_crisis = df2.loc[(df2['Time'] >=  '2016-01-01') & (df2['Time'] <=  '2019-12-31' )]
```

In [85]:

```
import matplotlib.style as style
style.use('fivethirtyeight')

#Adding plots
fig,ax = plt.subplots(figsize = (9,3))
ax.plot(corona_crisis['Time'],
        corona_crisis['rolling_mean'],
        linewidth=1 , color = 'grey')

#Highlighting the 2007-2008 crises
ax.plot(corona_crisis_20['Time'],
        corona_crisis_20['rolling_mean'],
        linewidth=3 , color = 'red')

ax.set_xticklabels([])

x= 0.02
for year in ['2016', '2017', '2018', '2019', '2020', '2021']:
    ax.text(x, -0.08, year, alpha = 0.5, fontsize = 11, transform = plt.gca().transAxes)
    x += 0.183

ax.set_yticklabels([])

y=0.02
for rate in ['1.05', '1.10', '1.15', '1.20']:
    ax.text(-.05, y, rate, alpha = 0.5, fontsize = 11, transform = plt.gca().transAxes)
    y += 0.248


# Adding title and subtitle
ax.text(-.05, 1.2, "Euro-USD rates peaked at 1.22 during COVID-19 crisis",
       weight = 'bold', transform = plt.gca().transAxes)
ax.text(-.05, 1.1, "Euro-USD exchange rates between 2019 and 2020",
       size = 12, transform = plt.gca().transAxes)


plt.show()
```



Observations:

- The graph shows the fluctuation in the Euro-USD exchange rates over this period.
- Notably, there was a significant dip around late 2019 to early 2020, where the value dropped close to its lowest point.
- Following that, there was a sharp increase in the rate around mid-2020, peaking at approximately `1.22` (highlighted in red on the graph).
- This peak occurred during the COVID-19 crisis.
- Lowest Point: The lowest point on the graph corresponds to the time when the Euro was weakest against the USD.
- Peak: The peak at `1.22` indicates a strong Euro relative to the USD during the crisis.

`The US presidents Tenure`

In [100…

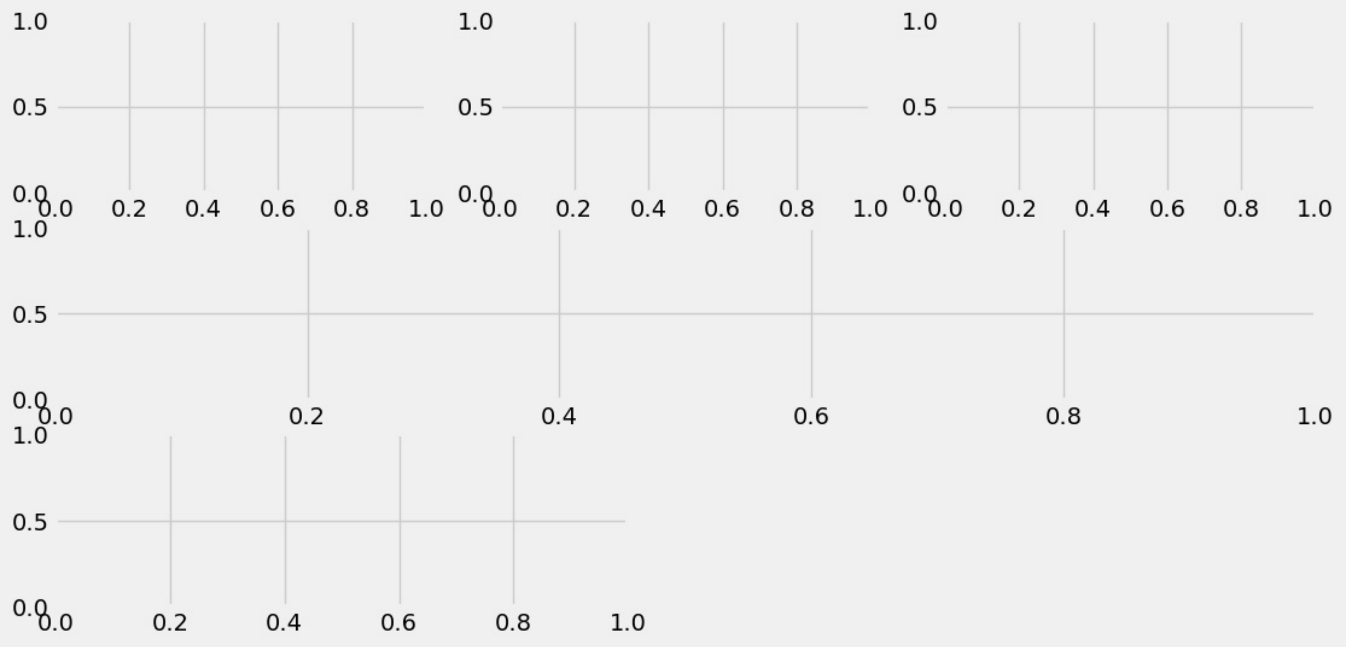
```
bush_obama_trump = df2.copy()[(df['Time'].dt.year >= 2001) & (df['Time'].dt.year < 2021)]
bush = bush_obama_trump.copy(
                              )[(bush_obama_trump['Time'].dt.year < 2009)]
obama = bush_obama_trump.copy(
                              )[(bush_obama_trump['Time'].dt.year >= 2009) &
                                (bush_obama_trump['Time'].dt.year < 2017)]
trump = bush_obama_trump.copy(
                              )[(bush_obama_trump['Time'].dt.year >= 2017) &
                                (bush_obama_trump['Time'].dt.year < 2021)]
```

Below, you'll notice we used matplotlib's functional approach to build the graph. We use this approach because it offers more flexibility in arranging the subplots.

- We first build three of the graphs on a 2-by-3 grid (this grid should hace six subplots, but we only three, the bottom one remains empty).
- We then build only the bottom graph of a 2-by-1 grid (this grid should have two subplots; the top rows remains empty)
- The two grids are merged, and we end up with three graphs on the top and one graph on the botton row.

In [101…

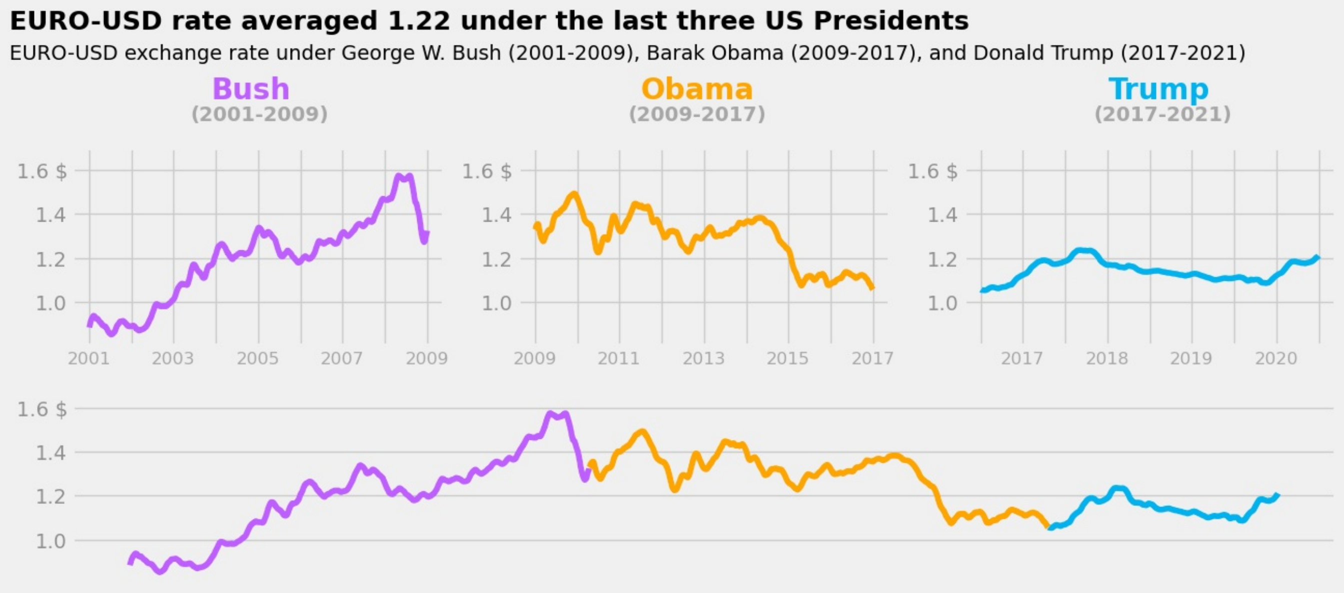
```
# Adding style
style.use('fivethirtyeight')

# Adding the subplots
plt.figure(figsize = (12,6))

# Pattern 1
ax1 = plt.subplot(3,3,1)
ax2 = plt.subplot(3,3,2)
ax3 = plt.subplot(3,3,3)

# Pattern 2
ax4 = plt.subplot(3,1,2)

# Pattern 3
ax5 = plt.subplot(3,2,5)
```



In [117…

```
# Adding style
style.use('fivethirtyeight')

# Adding the subplots
plt.figure(figsize = (14,8))

# Pattern 1
ax1 = plt.subplot(3,3,1)
ax2 = plt.subplot(3,3,2)
ax3 = plt.subplot(3,3,3)

# Pattern 2
ax4 = plt.subplot(3,1,2)


axes = [ax1, ax2, ax3, ax4]

# setting the axis
for ax in axes:
    ax.set_ylim(0.8,1.7)
    ax.set_yticks([1.0, 1.2, 1.4, 1.6])
    ax.set_yticklabels(['1.0', '1.2', '1.4', '1.6 $'], alpha = 0.4)


# Ax1: Bush
ax1.plot(bush['Time'], bush['rolling_mean'],
         color = '#BF5FFF')
ax1.set_xticklabels(['', '2001','' , '2003', '', '2005', '', '2007', '', '2009'],
                     alpha = 0.3, size = 12)
ax1.text(0.11, 2.45, 'Bush', fontsize=20, weight = 'bold', color = '#BF5FFF',
        transform = plt.gca().transAxes)
ax1.text(0.093, 2.34, '(2001-2009)', weight = 'bold', alpha = 0.3,
         transform = plt.gca().transAxes)

# Ax2: Obama
ax2.plot(obama['Time'], obama['rolling_mean'],
         color = '#ffa500')
ax2.set_xticklabels(['', '2009','' , '2011', '', '2013', '', '2015', '', '2017'],
                     alpha = 0.3, size = 12)
ax2.text(0.45, 2.45, 'Obama', fontsize=20, weight = 'bold', color = '#ffa500',
        transform = plt.gca().transAxes)
ax2.text(0.44, 2.34, '(2009-2017)', weight = 'bold', alpha = 0.3,
         transform = plt.gca().transAxes)

# Ax3: Trump
ax3.plot(trump['Time'], trump['rolling_mean'],
         color = '#00B2EE')
ax3.set_xticklabels(['','2017','' , '2018', '', '2019', '', '2020', '', '2021'],
                     alpha = 0.3, size = 12)
ax3.text(0.82, 2.45, 'Trump', fontsize=20, weight = 'bold', color = '#00B2EE',
        transform = plt.gca().transAxes)
ax3.text(0.808, 2.34, '(2017-2021)', weight = 'bold', alpha = 0.3,
         transform = plt.gca().transAxes)

# Ax4: Bush-Obama-Trump

ax4.plot(bush['Time'], bush['rolling_mean'], color = '#BF5FFF')
ax4.plot(obama['Time'], obama['rolling_mean'], color = '#ffa500')
ax4.plot(trump['Time'], trump['rolling_mean'], color = '#00B2EE')
```

```
ax4.set_xticks([])

# Adding a title and a subtitle

ax.text(-0.05, 2.8, "EURO-USD rate averaged 1.22 under the last three US Presidents", size = 18,
       weight = 'bold', transform = plt.gca().transAxes)
ax.text(-0.05, 2.65, "EURO-USD exchange rate under George W. Bush (2001-2009), Barak Obama (2009-2017),\
       and Donald Trump (2017-2021)", size = 14, transform = plt.gca().transAxes)

plt.tight_layout()
plt.show()
```



Observations:

- George W. Bush (2001–2009): The exchange rate increased from around `0.8` to `1.6` during his presidency.
- Barack Obama (2009–2017): The rate started at about `1.4` and decreased to approximately `1.05`.
- Donald Trump (2017–2021): The rate remained relatively stable, fluctuating between approximately `1.05` and `1.25`.
- The average EURO-USD exchange rate across the last three US Presidents was approximately `1.22`

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js