In [1]:
```
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
```

This is a visualization project where EURO/USD data will be analized to find out meaningfull insights

In [2]:
```
ex_rates  = pd.read_csv('euro-daily-hist_1999_2022.csv')
ex_rates.shape
```

Out[2]: (6456, 41)

In [3]:
```
ex_rates.head()
```

Out[3]:

	Period\Unit:	[Australian dollar]	[Bulgarian lev]	[Brazilian real]	[Canadian dollar]	[Swiss franc]	[Chinese yuan renminbi]	[Cypriot pound]	[Czech koruna]
0	2023-12-15	1.6324	1.9558	5.4085	1.4653	0.9488	7.7812	NaN	24.477
1	2023-12-14	1.6288	1.9558	5.3349	1.4677	0.949	7.7866	NaN	24.408
2	2023-12-13	1.6452	1.9558	5.3609	1.4644	0.9452	7.7426	NaN	24.476
3	2023-12-12	1.6398	1.9558	5.3327	1.4656	0.9443	7.7447	NaN	24.42
4	2023-12-11	1.642	1.9558	5.3169	1.4609	0.9478	7.7206	NaN	24.367

5 rows × 41 columns

In [4]:
```
ex_rates.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
```

DATA Cleaning

In [5]:
```
ex_rates.rename(columns={'[US dollar ]': "US_dollar", 'Period\\Unit:': 'Tir
```

In [6]:
```
ex_rates.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6456 entries, 0 to 6455
Data columns (total 41 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Time                      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 [7]:
```
ex_rates['Time'] = pd.to_datetime(ex_rates['Time'])
```

In [8]:
```
ex_rates.sort_values('Time',inplace= True)
```

In [9]:
```
ex_rates.head()
```

Out[9]:

	Time	[Australian dollar]	[Bulgarian lev]	[Brazilian real]	[Canadian dollar]	[Swiss franc]	[Chinese yuan renminbi]	[Cypriot pound]	[Czech koruna]	[D k
6455	1999-01-04	1.9100	NaN	NaN	1.8004	1.6168	NaN	0.58231	35.107	7
6454	1999-01-05	1.8944	NaN	NaN	1.7965	1.6123	NaN	0.58230	34.917	7
6453	1999-01-06	1.8820	NaN	NaN	1.7711	1.6116	NaN	0.58200	34.850	7
6452	1999-01-07	1.8474	NaN	NaN	1.7602	1.6165	NaN	0.58187	34.886	7
6451	1999-01-08	1.8406	NaN	NaN	1.7643	1.6138	NaN	0.58187	34.938	7

5 rows × 41 columns

In [10]:
```
euro_to_usd= ex_rates[['Time','US_dollar']].copy()
```

In [11]:
```
euro_to_usd['US_dollar'].value_counts()
```

Out[11]:
```
-         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 [12]:
```
euro_to_usd= euro_to_usd[euro_to_usd['US_dollar']!= '-']
euro_to_usd['US_dollar'].value_counts()
```

Out[12]:
```
1.2276    9
1.1215    8
1.0888    7
1.0868    7
1.1305    7
         ..
1.4304    1
1.4350    1
1.4442    1
1.4389    1
1.0804    1
Name: US_dollar, Length: 3768, dtype: int64
```

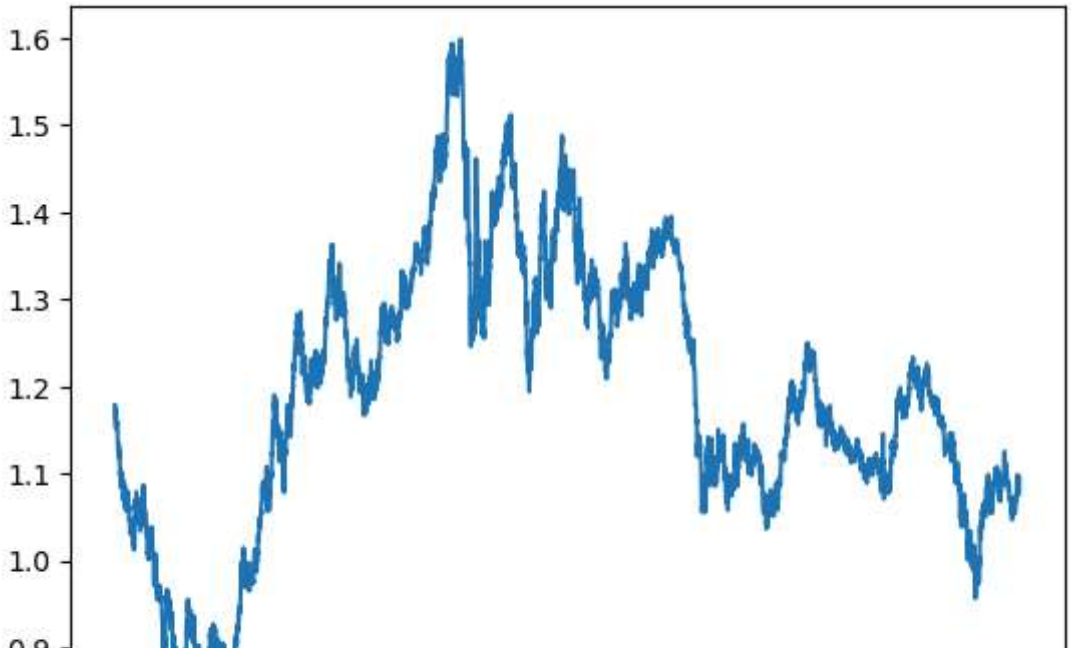
In [13]:
```
euro_to_usd['US_dollar']= euro_to_usd['US_dollar'].astype(float)
euro_to_usd.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6394 entries, 6455 to 0
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype         
---  ------     --------------  -----         
 0   Time       6394 non-null   datetime64[ns]
 1   US_dollar  6394 non-null   float64       
dtypes: datetime64[ns](1), float64(1)
memory usage: 149.9 KB
```

In [14]:
```
import matplotlib.pyplot as plt
```

Ploting

In [15]:
```
plt.plot(euro_to_usd['Time'],euro_to_usd['US_dollar'])
plt.show()
```



In [16]:
```
values= pd.DataFrame()
values['daily_values'] = pd.Series(range(1,20,2))
values
```

Out[16]:

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

In [17]:
```
values['rolling_mean2']= values['daily_values'].rolling(2).mean()
values
```

Out[17]:

	daily_values	rolling_mean2
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

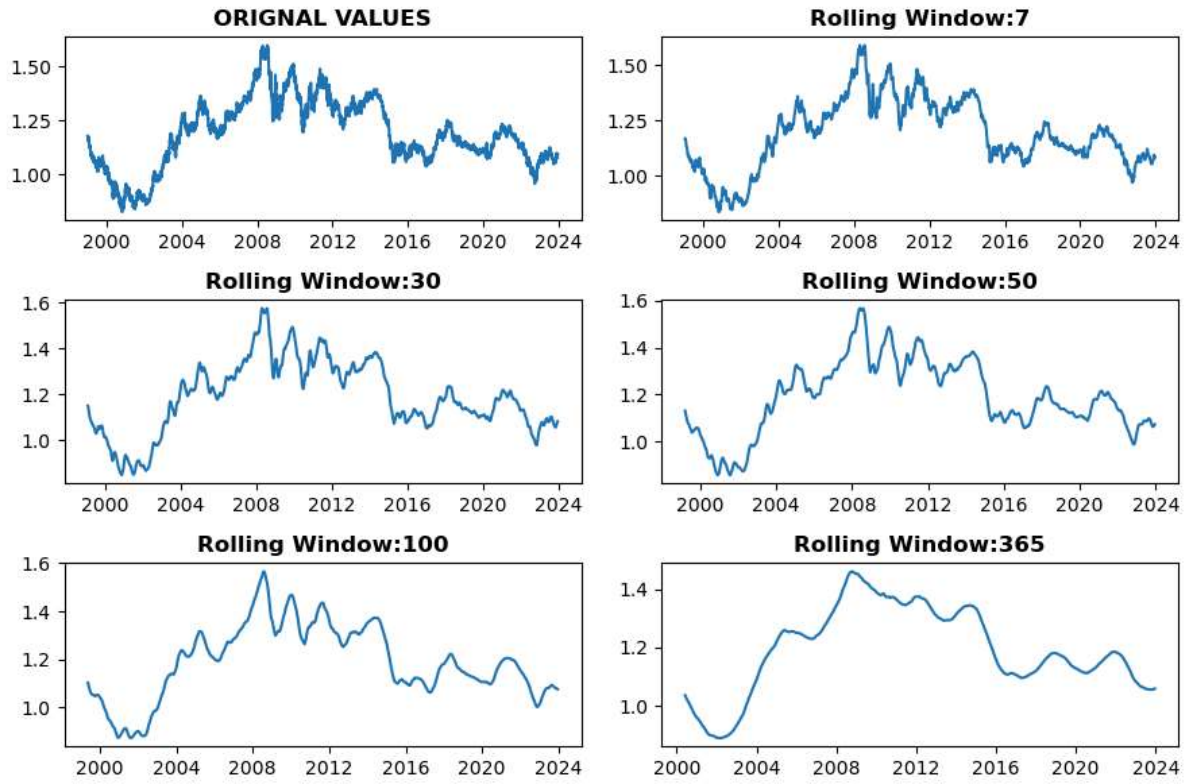
In [18]:
```
values['rolling_mean3']= values['daily_values'].rolling(3).mean()
values['rolling_mean5']= values['daily_values'].rolling(5).mean()
values
```

Out[18]:

	daily_values	rolling_mean2	rolling_mean3	rolling_mean5
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

Applying Rolling mean for smothing the data

In [19]:
```
plt.figure(figsize= (9,6))
plt.subplot(3,2,1)
plt.plot(euro_to_usd['Time'], euro_to_usd["US_dollar"])
plt.title("ORIGNAL 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(euro_to_usd['Time'], euro_to_usd['US_dollar'].rolling(rolling_mean
    plt.title('Rolling Window:'+str(rolling_mean), weight='bold')
plt.tight_layout()
plt.show()
```



In [20]:
```
euro_to_usd['rolling_mean']= euro_to_usd['US_dollar'].rolling(30).mean()
euro_to_usd
```

Out[20]:

	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

Analysing and Ploting Financial Crisis of 2008

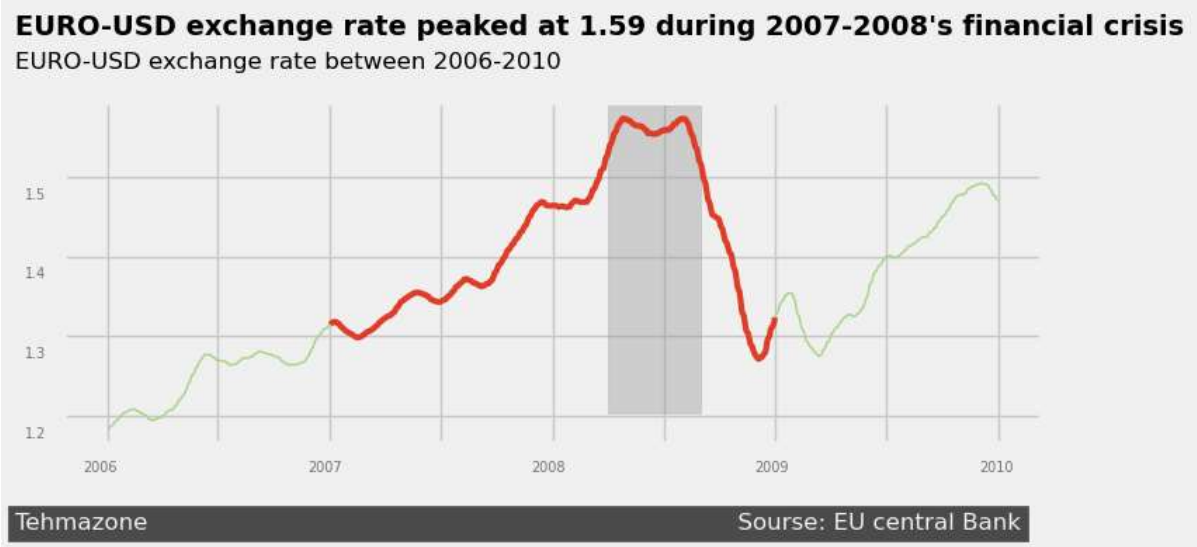
In [21]:
```
financial_crisis= euro_to_usd.copy()[(euro_to_usd['Time'].dt.year >=2006) & (eu
financial_crisis
financial_crisis_7_8= euro_to_usd.copy()[(euro_to_usd['Time'].dt.year >=2007) &
financial_crisis_7_8
```

Out[21]:

	Time	US_dollar	rolling_mean
4369	2007-01-02	1.3270	1.314257
4368	2007-01-03	1.3231	1.315780
4367	2007-01-04	1.3106	1.316663
4366	2007-01-05	1.3084	1.317563
4365	2007-01-08	1.3006	1.317963
...	...	...	...
3854	2008-12-23	1.3978	1.303717
3853	2008-12-24	1.4005	1.308633
3850	2008-12-29	1.4270	1.314450
3849	2008-12-30	1.4098	1.319193
3848	2008-12-31	1.3917	1.323383

511 rows × 3 columns

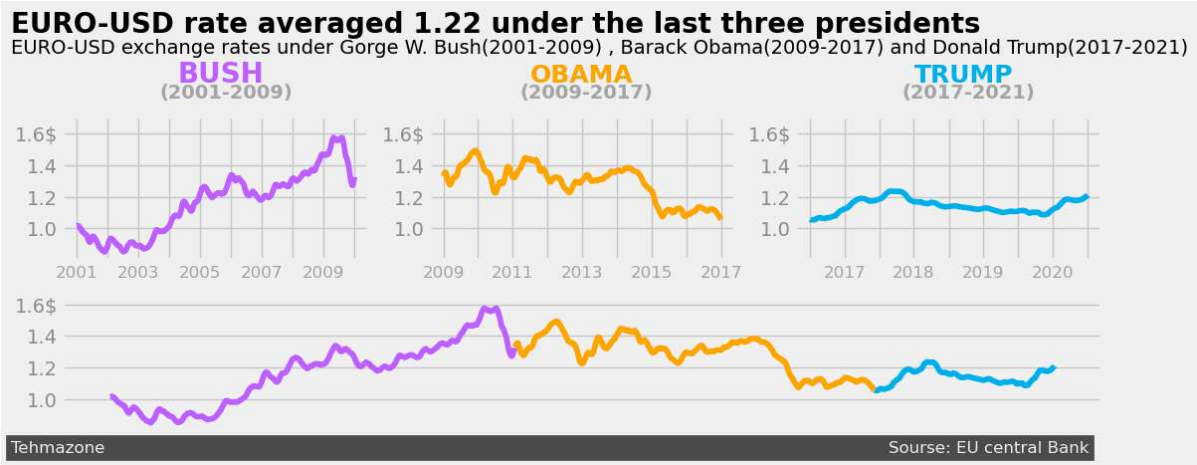
In [22]:
```
import matplotlib.style as style
style.use('fivethirtyeight')

fig,ax = plt.subplots(figsize=( 8,3))
ax.plot(financial_crisis["Time"], financial_crisis["rolling_mean"], linewidth=1

ax.plot(financial_crisis_7_8["Time"], financial_crisis_7_8["rolling_mean"], lin

ax.set_xticklabels([])
x = 0.02
for year in ['2006','2007','2008','2009','2010']:
    ax.text(x, -0.08, year, alpha= 0.5 , fontsize= 7, transform= plt.gca().tran
    x = x + 0.22888

ax.set_yticklabels([])
y = 0.02
for rate in ['1.2','1.3','1.4','1.5']:
    ax.text(-0.04,y, rate, alpha= 0.5 , fontsize= 7, transform= plt.gca().trans
    y = y+ 0.2333
ax.text(-0.05,1.2, "EURO-USD exchange rate peaked at 1.59 during 2007-2008's fi
ax.text(-0.05,1.1, "EURO-USD exchange rate between 2006-2010", size = 12, trans

ax.text(-0.05,-0.25, "Tehmazone" + ' '*80 +'Sourse: EU central Bank',color= "#

ax.axvspan(xmin=pd.to_datetime("2008-04-1"), xmax=pd.to_datetime("2008-09-1"),

plt.show()
```



Euro/Usd variation during last 3 US presidents

In [23]:
```
bush_obama_trump= euro_to_usd.copy()[(euro_to_usd['Time'].dt.year >= 2000) & (
bush= bush_obama_trump.copy()[(bush_obama_trump['Time'].dt.year < 2009)]
obama= bush_obama_trump.copy()[(bush_obama_trump['Time'].dt.year >= 2009) & (bu
trump= bush_obama_trump.copy()[(bush_obama_trump['Time'].dt.year >= 2017) & (bu
```

In [24]:
```
style.use('fivethirtyeight')
plt.figure(figsize=(12,6))
ax1 = plt.subplot(3,3,1)
ax2 = plt.subplot(3,3,2)
ax3 = plt.subplot(3,3,3)
ax4 = plt.subplot(3,1,2)
axes = [ax1,ax2,ax3,ax4]

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.plot(bush['Time'], bush['rolling_mean'], color= '#bf5fff')
ax1.set_xticklabels(['','2001','','2003','','2005','','2007','','2009'], alpha=

ax1.text(0.11,2.45,'BUSH', fontsize= 20, weight='bold', color= '#bf5fff', trans
ax1.text(0.093,2.34,'(2001-2009)', weight='bold', alpha = 0.3, transform= plt.g

ax2.plot(obama['Time'], obama['rolling_mean'], color= '#ffa500')
ax2.set_xticklabels(['','2009','','2011','','2013','','2015','','2017'], alpha=

ax2.text(0.45,2.45,'OBAMA', fontsize= 18, weight='bold', color= '#ffa500', tran
ax2.text(0.44,2.34,'(2009-2017)', weight='bold', alpha = 0.3, transform= plt.g

ax3.plot(trump['Time'], trump['rolling_mean'], color= '#00B2EE')
ax3.set_xticklabels(['','2017','','2018','','2019','','2020','','2021'], alpha=

ax3.text(0.82,2.45,'TRUMP', fontsize= 18, weight='bold', color= '#00B2EE', tran
ax3.text(0.808,2.34,'(2017-2021)', weight='bold', alpha = 0.3, transform= plt.g

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([])

ax1.text(-0.05,2.8,"EURO-USD rate averaged 1.22 under the last three presidents
ax1.text(-0.05,2.65,"EURO-USD exchange rates under Gorge W. Bush(2001-2009) , E

ax.text(-0.05,-0.15, "Tehmazone" + ' '*150 +'Sourse: EU central Bank',color= "#

plt.show()
```


EURO-USD rate averaged 1.22 under the last three presidents
EURO-USD exchange rates under Gorge W. Bush(2001-2009) , Barack Obama(2009-2017) and Donald Trump(2017-2021)
BUSH
(2001-2009)
OBAMA
(2009-2017)
TRUMP
(2017-2021)
1.6$
1.4
1.2
1.0
2001 2003 2005 2007 2009
2009 2011 2013 2015 2017
2017 2018 2019 2020
Tehmazone
Sourse: EU central Bank

In [ ]: