Exchange _Rate_Viz

March 19, 2025

1 Storytelling Data Visualization on Exchange Rate

1.1 Importing the required Libraries

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.style as style
style.use('bmh')
```

1.2 Read in the Dataset and Inspect the Dataset

[2]:
```
exchange_rates = pd.read_csv('euro-daily-hist_1999_2020.csv')
print(exchange_rates.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5699 entries, 0 to 5698
Data columns (total 41 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Period\Unit:               5699 non-null   object
 1   [Australian dollar ]       5699 non-null   object
 2   [Bulgarian lev ]           5297 non-null   object
 3   [Brazilian real ]          5431 non-null   object
 4   [Canadian dollar ]         5699 non-null   object
 5   [Swiss franc ]             5699 non-null   object
 6   [Chinese yuan renminbi ]   5431 non-null   object
 7   [Cypriot pound ]           2346 non-null   object
 8   [Czech koruna ]            5699 non-null   object
 9   [Danish krone ]            5699 non-null   object
 10  [Estonian kroon ]          3130 non-null   object
 11  [UK pound sterling ]       5699 non-null   object
 12  [Greek drachma ]           520 non-null    object
 13  [Hong Kong dollar ]        5699 non-null   object
 14  [Croatian kuna ]           5431 non-null   object
 15  [Hungarian forint ]        5699 non-null   object
 16  [Indonesian rupiah ]       5699 non-null   object
 17  [Israeli shekel ]         5431 non-null   object
 18  [Indian rupee ]           5431 non-null   object
 19  [Iceland krona ]          3292 non-null   float64
 20  [Japanese yen ]           5699 non-null   object
 21  [Korean won ]             5699 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 ]           5699 non-null   object
 26  [Malaysian ringgit ]      5699 non-null   object
 27  [Norwegian krone ]        5699 non-null   object
 28  [New Zealand dollar ]     5699 non-null   object
 29  [Philippine peso ]        5699 non-null   object
 30  [Polish zloty ]           5699 non-null   object
 31  [Romanian leu ]           5637 non-null   float64
 32  [Russian rouble ]         5699 non-null   object
 33  [Swedish krona ]          5699 non-null   object
 34  [Singapore dollar ]       5699 non-null   object
 35  [Slovenian tolar ]        2085 non-null   object
 36  [Slovak koruna ]          2608 non-null   object
 37  [Thai baht ]              5699 non-null   object
 38  [Turkish lira ]           5637 non-null   float64
 39  [US dollar ]              5699 non-null   object
 40  [South African rand ]     5699 non-null   object
dtypes: float64(3), object(38)
memory usage: 1.8+ MB
None
```

1.3 Display the five records of the Dataset

[3]:
```
print(exchange_rates.head())
```

```
  Period\Unit: [Australian dollar ] [Bulgarian lev ] [Brazilian real ]  \
0   2021-01-08               1.5758           1.9558            6.5748
1   2021-01-07               1.5836           1.9558            6.5172
2   2021-01-06               1.5824           1.9558            6.5119
3   2021-01-05               1.5927           1.9558            6.5517
4   2021-01-04               1.5928           1.9558            6.3241

  [Canadian dollar ] [Swiss franc ] [Chinese yuan renminbi ] [Cypriot pound ]  \
0             1.5543         1.0827                   7.9184              NaN
1             1.5601         1.0833                   7.9392              NaN
2             1.5640         1.0821                   7.9653              NaN
3             1.5651         1.0803                   7.9315              NaN
4             1.5621         1.0811                   7.9484              NaN

  [Czech koruna ] [Danish krone ]  … [Romanian leu ] [Russian rouble ]  \
0          26.163          7.4369  …          4.8708           90.8000
```

```
1          26.147          7.4392  …          4.8712          91.2000
2          26.145          7.4393  …          4.8720          90.8175
3          26.227          7.4387  …          4.8721          91.6715
4          26.141          7.4379  …          4.8713          90.3420

  [Swedish krona ] [Singapore dollar ] [Slovenian tolar ] [Slovak koruna ]  \
0          10.0510              1.6228                NaN              NaN
1          10.0575              1.6253                NaN              NaN
2          10.0653              1.6246                NaN              NaN
3          10.0570              1.6180                NaN              NaN
4          10.0895              1.6198                NaN              NaN

  [Thai baht ] [Turkish lira ] [US dollar ]  [South African rand ]
0      36.8480          9.0146       1.2250                18.7212
1      36.8590          8.9987       1.2276                18.7919
2      36.9210          9.0554       1.2338                18.5123
3      36.7760          9.0694       1.2271                18.4194
4      36.7280          9.0579       1.2296                17.9214

[5 rows x 41 columns]
```

1.4 Display the last five records of the Dataset

```
[4]: print(exchange_rates.tail())
```

```
     Period\Unit: [Australian dollar ] [Bulgarian lev ] [Brazilian real ]  \
5694   1999-01-08               1.8406              NaN               NaN
5695   1999-01-07               1.8474              NaN               NaN
5696   1999-01-06               1.8820              NaN               NaN
5697   1999-01-05               1.8944              NaN               NaN
5698   1999-01-04               1.9100              NaN               NaN

     [Canadian dollar ] [Swiss franc ] [Chinese yuan renminbi ]  \
5694             1.7643         1.6138                      NaN
5695             1.7602         1.6165                      NaN
5696             1.7711         1.6116                      NaN
5697             1.7965         1.6123                      NaN
5698             1.8004         1.6168                      NaN

     [Cypriot pound ] [Czech koruna ] [Danish krone ]  … [Romanian leu ]  \
5694          0.58187          34.938          7.4433  …          1.3143
5695          0.58187          34.886          7.4431  …          1.3092
5696          0.58200          34.850          7.4452  …          1.3168
5697          0.58230          34.917          7.4495  …          1.3168
5698          0.58231          35.107          7.4501  …          1.3111

     [Russian rouble ] [Swedish krona ] [Singapore dollar ]  \
5694           27.2075           9.1650              1.9537
```

```
5695           26.9876           9.1800             1.9436
5696           27.4315           9.3050             1.9699
5697           26.5876           9.4025             1.9655
5698           25.2875           9.4696             1.9554

     [Slovenian tolar ] [Slovak koruna ] [Thai baht ] [Turkish lira ]  \
5694           188.8400            42.560      42.5590           0.3718
5695           188.8000            42.765      42.1678           0.3701
5696           188.7000            42.778      42.6949           0.3722
5697           188.7750            42.848      42.5048           0.3728
5698           189.0450            42.991      42.6799           0.3723

     [US dollar ]  [South African rand ]
5694       1.1659                 6.7855
5695       1.1632                 6.8283
5696       1.1743                 6.7307
5697       1.1790                 6.7975
5698       1.1789                 6.9358

[5 rows x 41 columns]
```

1.5 Data Cleaning

1.5.1 1. Renaming Columns

[5]:
```
new_column_names = []
for column_name in exchange_rates.columns:
    column_name = column_name.replace('[', '')
    column_name = column_name.replace(' ]', '')
    column_name = column_name.replace(' ', '_')
    column_name = column_name.lower()
    new_column_names.append(column_name)
exchange_rates.columns = new_column_names
exchange_rates.rename(columns={'period\\unit:': 'time'}, inplace = True)
exchange_rates.columns
```

[5]:
```
Index(['time', 'australian_dollar', 'bulgarian_lev', 'brazilian_real',
       'canadian_dollar', 'swiss_franc', 'chinese_yuan_renminbi',
       'cypriot_pound', 'czech_koruna', 'danish_krone', 'estonian_kroon',
       'uk_pound_sterling', 'greek_drachma', 'hong_kong_dollar',
       'croatian_kuna', 'hungarian_forint', 'indonesian_rupiah',
       'israeli_shekel', 'indian_rupee', 'iceland_krona', 'japanese_yen',
       'korean_won', 'lithuanian_litas', 'latvian_lats', 'maltese_lira',
       'mexican_peso', 'malaysian_ringgit', 'norwegian_krone',
       'new_zealand_dollar', 'philippine_peso', 'polish_zloty', 'romanian_leu',
       'russian_rouble', 'swedish_krona', 'singapore_dollar',
       'slovenian_tolar', 'slovak_koruna', 'thai_baht', 'turkish_lira',
       'us_dollar', 'south_african_rand'],
      dtype='object')
```

1.5.2 2. Dropping Unnecessary Columns and Change the Datatype of 'time' column

[6]:
```
exchange_rates['time'] = pd.to_datetime(exchange_rates['time'])
euro_to_dollar = exchange_rates[['time', 'us_dollar']]
print(euro_to_dollar.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5699 entries, 0 to 5698
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   time       5699 non-null   datetime64[ns]
 1   us_dollar  5699 non-null   object
dtypes: datetime64[ns](1), object(1)
memory usage: 89.2+ KB
None
```

1.5.3 3. Sorting Records

[7]:
```
euro_to_dollar = euro_to_dollar.sort_values(['time'])
print(euro_to_dollar.head())
```

```
           time us_dollar
5698 1999-01-04    1.1789
5697 1999-01-05    1.1790
5696 1999-01-06    1.1743
5695 1999-01-07    1.1632
5694 1999-01-08    1.1659
```

1.5.4 4. Clean us_dollar column and change its datatype

[8]:
```
euro_to_dollar['us_dollar'].value_counts(dropna = False)
```

[8]:
```
us_dollar
-          62
1.2276      9
1.1215      8
1.1305      7
1.1797      6
           ..
1.2571      1
1.2610      1
1.2651      1
1.2632      1
1.2193      1
Name: count, Length: 3528, dtype: int64
```

[9]:
```
euro_to_dollar['us_dollar'] = euro_to_dollar[euro_to_dollar['us_dollar'] !=␣
↪'-']['us_dollar'].astype('float')
euro_to_dollar.reset_index(drop = True, inplace = True)
euro_to_dollar.dropna(inplace = True)
print(euro_to_dollar.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 5637 entries, 0 to 5698
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   time       5637 non-null   datetime64[ns]
 1   us_dollar  5637 non-null   float64
dtypes: datetime64[ns](1), float64(1)
memory usage: 132.1 KB
None
```

1.6 Rolling Mean

We will calculate the rolling means for the US_dollar column using a moving window of 30 days. We will add the rolling means to a new column named rolling_mean.

[10]:
```
euro_to_dollar['rolling_mean'] = euro_to_dollar['us_dollar'].rolling(30).mean()
print(euro_to_dollar.head())
```

```
        time  us_dollar  rolling_mean
0 1999-01-04     1.1789           NaN
1 1999-01-05     1.1790           NaN
2 1999-01-06     1.1743           NaN
3 1999-01-07     1.1632           NaN
4 1999-01-08     1.1659           NaN
```

We will show comparatively how the euro-dollar rate changed under the presidency of George W. Bush (2001-2009), Barack Obama (2009-2017), and Donald Trump (2017-2021)). We will use the line plot.

1.7 Plotting EURO-USD Exchange Rates Under the presidency of George W. Bush (2000 - 2008), Barrack Obama (2009 - 2016) and Donald J.Trump (2017 - 2020)

[11]:
```
# Initialise a figure
fig = plt.figure(figsize = (10, 6))

# Define subplots
ax1 = plt.subplot(2, 3, 1)
ax2 = plt.subplot(2, 3, 2)
ax3 = plt.subplot(2, 3, 3)
ax4 = plt.subplot(2, 3, (4,6))
axes = [ax1, ax2, ax3, ax4]
```

```
# Plot a number line in each subplot
for ax in axes:
    ax.plot(euro_to_dollar['time'], euro_to_dollar['rolling_mean'], color =␣
 ↪'gray', alpha = 0.5, lw = 0.7)
    ax.grid(False)

# George W. Bush (2001 - 2009)
bush_time = euro_to_dollar[(euro_to_dollar['time'].dt.year >= 1999) &␣
 ↪(euro_to_dollar['time'].dt.year < 2009)]
ax1.plot(bush_time['time'], bush_time['rolling_mean'], color = 'blue', lw = 1)
ax1.set_title('George Bush (2000 - 2008)', fontdict = {'size':12, 'weight':␣
 ↪'bold', 'color': 'blue'})
ax1.axvline(pd.to_datetime('2009-01-01'), lw = 0.5 , color = 'black', alpha = 0.
 ↪5)
ax1.set_yticks([0.8, 1.0, 1.2, 1.4, 1.6])
ax1.set_xticks([pd.to_datetime('2000-01-01'), pd.to_datetime('2009-01-01'),
                pd.to_datetime('2017-01-01'), pd.to_datetime('2021-01-01')],
              ['2000', '2009', '2017', '2021'])
ax1.tick_params(labelbottom = False, labeltop = True, bottom = False, top =␣
 ↪True, labelcolor = 'gray')

# Barrack Obama (2009 - 2017)
obama_time = euro_to_dollar[(euro_to_dollar['time'].dt.year >= 2009) &␣
 ↪(euro_to_dollar['time'].dt.year < 2017)]
ax2.plot(obama_time['time'], obama_time['rolling_mean'], color = 'g', lw = 1)
ax2.set_title('Barrack Obama (2009 - 2016)', fontdict = {'size':12, 'weight':␣
 ↪'bold', 'color': 'g'})
ax2.axvline(pd.to_datetime('2009-01-01'), lw = 0.5 , color = 'black', alpha = 0.
 ↪5)
ax2.axvline(pd.to_datetime('2017-01-01'), lw = 0.5 , color = 'black', alpha = 0.
 ↪5)
ax2.set_xticks([pd.to_datetime('2000-01-01'), pd.to_datetime('2009-01-01'),
                pd.to_datetime('2017-01-01'), pd.to_datetime('2021-01-01')],
              ['2000', '2009', '2017', '2021'])
ax2.tick_params(labelbottom = False, labeltop = True, bottom = False, top =␣
 ↪True, labelcolor = 'gray')
ax2.set_yticks([])

# Donald J. Trump (2017 - 2021)
trump_time = euro_to_dollar[(euro_to_dollar['time'].dt.year >= 2017) &␣
 ↪(euro_to_dollar['time'].dt.year < 2021)]
ax3.plot(trump_time['time'], trump_time['rolling_mean'], color = 'orange', lw =␣
 ↪1)
ax3.set_yticks([])
```

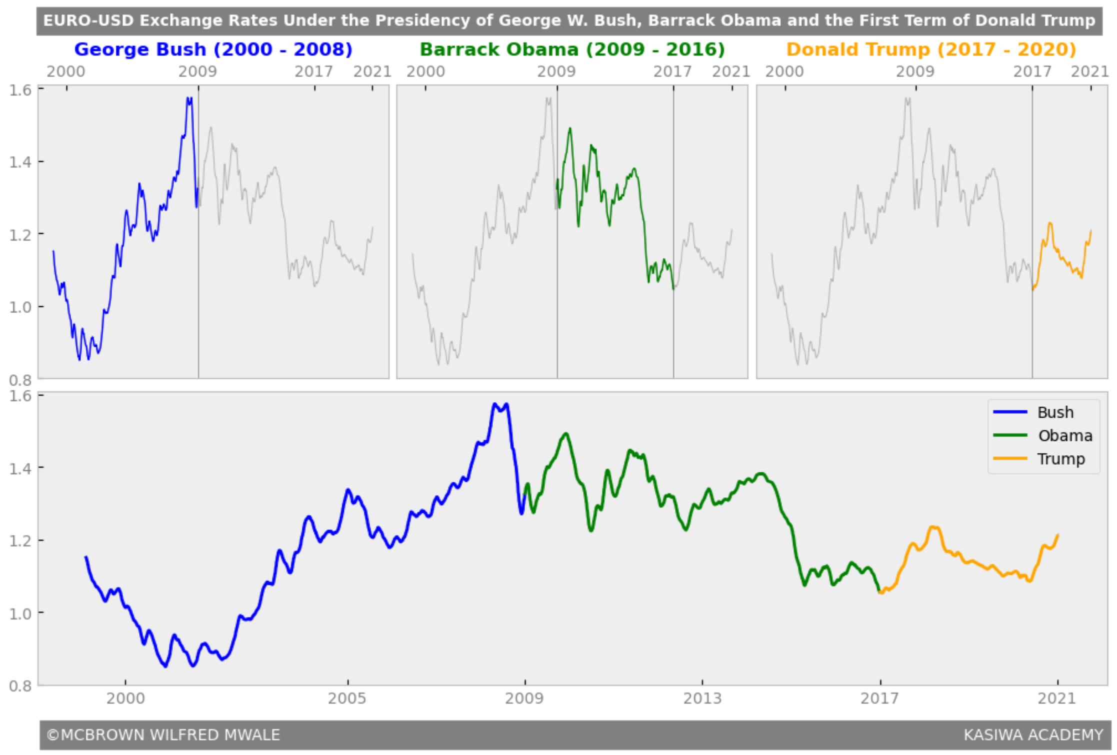
```
ax3.set_title('Donald Trump (2017 - 2020)', fontdict = {'size':12, 'weight':␣
↪'bold', 'color': 'orange'})
ax3.axvline(pd.to_datetime('2017-01-01'), lw = 0.5 , color = 'black', alpha = 0.
↪5)
ax3.set_xticks([pd.to_datetime('2000-01-01'), pd.to_datetime('2009-01-01'),
                pd.to_datetime('2017-01-01'), pd.to_datetime('2021-01-01')],
               ['2000', '2009', '2017', '2021'])
ax3.tick_params(labelbottom = False, labeltop = True, bottom = False, top =␣
↪True, labelcolor = 'gray')

# All presidency
ax4.plot(bush_time['time'], bush_time['rolling_mean'], color = 'blue', label =␣
↪'Bush')
ax4.plot(obama_time['time'], obama_time['rolling_mean'], color = 'g', label =␣
↪'Obama')
ax4.plot(trump_time['time'], trump_time['rolling_mean'], color = 'orange',␣
↪label = 'Trump')
ax4.legend()
ax4.set_xticks([pd.to_datetime('2000-01-01'), pd.to_datetime('2005-01-01'), pd.
↪to_datetime('2009-01-01'),
                pd.to_datetime('2013-01-01'), pd.to_datetime('2017-01-01'), pd.
↪to_datetime('2021-01-01')],
               ['2000', '2005', '2009', '2013', '2017', '2021'])
ax4.set_yticks([0.8, 1.0, 1.2, 1.4, 1.6])
ax4.tick_params(labelcolor = 'gray')

plt.tight_layout(pad = 0)
plt.text(x = 10280, y = 2.62, s = 'EURO-USD Exchange Rates Under the Presidency␣
↪of George W. Bush, Barrack Obama and the First Term of Donald Trump',
            fontdict = {'weight':'semibold', 'size': '10', 'color': 'white'},␣
↪backgroundcolor = 'gray')

plt.text(x = 10300, y = 0.65, s = '©MCBROWN WILFRED MWALE'+' '*141 + 'KASIWA␣
↪ACADEMY', backgroundcolor = 'gray', color = 'white')
plt.savefig('euro_usd_exchange_rates.png', bbox_inches = 'tight')

plt.show()
```


EURO-USD Exchange Rates Under the Presidency of George W. Bush, Barrack Obama and the First Term of Donald Trump
George Bush (2000 - 2008)
Barrack Obama (2009 - 2016)
Donald Trump (2017 - 2020)
Bush
Obama
Trump
©MCBROWN WILFRED MWALE
KASIWA ACADEMY