Datos Macroeconómicos Internacionales a través de la API de FRED

Unidad de Análisis y Estudios Fiscales

Fecha: 15 de Noviembre

Authors: - Cesar Ramos - Martín Romero - Alfredo Alanoca

1.0.1 Getting Started

1.0.2 Brief Information about Federal Reserve Economic Data (FRED).

Purpose of this tool: Get macroeconomic data with a streamlined process.

- FRED is an extensive economic database, mantained by Federal Reserve Bank of St. Louis.
- Provides acces to a vast collection of economic data
- Includes macro indicators, financial market data

For more details, see the link: fredapi

1.1 Getting Started

Install the the *fredapi* package with pipy

Before getting started with the code, the first step is Get an **API Key** from the link.

1.1.1 Packages and Libraries

[ ]:

```
# Importación de librerias
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
import plotly.graph_objects as go
from google.colab import drive
import numpy as np
import plotly.express as px
import warnings
import locale
import plotly.figure_factory as ff
```

```
[ ]: from fredapi import Fred
     #key_fred_api = "7535b6698b2a167a685edb0fb55dcde0"
     key_fred_api = "d38d5676502b4a1eb4458d7a0e332df8"
     f = Fred(api_key = key_fred_api)
```

1.1.2 Example

```
[ ]: from fredapi import Fred
     import pandas as pd
     import numpy as np
     import hvplot.pandas

     # Retrieve the raw data from FRED
     df = f.get_series('UNRATE', observation_start='2006-1-1', frequency = 'm')

     # Plot
     df.hvplot(title = "Unemployment Rate")
```

```
[ ]: :Curve   [index]   (0)
```

1.1.3 Search Function

- The output is a huge dataframe that contains information on each series:

```
[ ]: f.search("VIX").head()
```

```
[ ]:                  id realtime_start realtime_end
     series id
     VIXCLS       VIXCLS     2024-11-27   2024-11-27  \
     VXVCLS       VXVCLS     2024-11-27   2024-11-27
     GVZCLS       GVZCLS     2024-11-27   2024-11-27
     OVXCLS       OVXCLS     2024-11-27   2024-11-27
     VXAPLCLS   VXAPLCLS     2024-11-27   2024-11-27

                                                title observation_start
     series id
     VIXCLS                CBOE Volatility Index: VIX        1990-01-02  \
     VXVCLS     CBOE S&P 500 3-Month Volatility Index        2007-12-04
     GVZCLS            CBOE Gold ETF Volatility Index        2008-06-03
     OVXCLS       CBOE Crude Oil ETF Volatility Index        2007-05-10
     VXAPLCLS                CBOE Equity VIX on Apple        2010-06-01

               observation_end     frequency frequency_short  units units_short
     series id
     VIXCLS         2024-11-26  Daily, Close               D  Index       Index  \
     VXVCLS         2024-11-26  Daily, Close               D  Index       Index
     GVZCLS         2024-11-26  Daily, Close               D  Index       Index
     OVXCLS         2024-11-26  Daily, Close               D  Index       Index
```

```
VXAPLCLS       2024-11-26  Daily, Close               D  Index       Index

               seasonal_adjustment seasonal_adjustment_short
series id
VIXCLS     Not Seasonally Adjusted                       NSA  \
VXVCLS     Not Seasonally Adjusted                       NSA
GVZCLS     Not Seasonally Adjusted                       NSA
OVXCLS     Not Seasonally Adjusted                       NSA
VXAPLCLS   Not Seasonally Adjusted                       NSA

                        last_updated popularity
series id
VIXCLS     2024-11-27 08:36:07-06:00         75  \
VXVCLS     2024-11-27 08:36:03-06:00         54
GVZCLS     2024-11-27 08:36:09-06:00         54
OVXCLS     2024-11-27 08:36:09-06:00         54
VXAPLCLS   2024-11-27 08:36:07-06:00         23

                                                       notes
series id
VIXCLS     VIX measures market expectation of near term v…
VXVCLS     Copyright, 2016, Chicago Board Options Exchang…
GVZCLS     Exchange Traded Funds (ETFs) are shares of tru…
OVXCLS     Exchange Traded Funds (ETFs) are shares of tru…
VXAPLCLS   Copyright, 2016, Chicago Board Options Exchang…
```

From this, we can isolate the specific data series that we'd like to work with.

[ ]:

```
# Federal Funds Effective Rate
# f.search("Federal Funds Effective Rate (FEDFUNDS)")
f.search("FEDFUNDS")
```

[ ]:

```
                 id realtime_start realtime_end                         title
series id
FEDFUNDS   FEDFUNDS     2024-11-27   2024-11-27  Federal Funds Effective Rate  \

          observation_start observation_end frequency frequency_short
series id
FEDFUNDS         1954-07-01      2024-10-01   Monthly               M  \

             units units_short      seasonal_adjustment
series id
FEDFUNDS   Percent           %  Not Seasonally Adjusted  \

          seasonal_adjustment_short              last_updated popularity
series id
FEDFUNDS                        NSA 2024-11-01 15:18:45-05:00         98  \
                                                   notes
series id
FEDFUNDS   Averages of daily figures.   For additional hi…
```

1.1.4 Creating a structured data frame from raw-data — GDP and Yields

Series ID names can be found on the page of the specific series.

[ ]:
```
columns = ['AUS','JAP','USA','GBR','KOR']

aus = f.get_series('NGDPRSAXDCAUQ')
jap = f.get_series('JPNRGDPEXP')
us = f.get_series('GDPC1')
uk = f.get_series('NGDPRSAXDCGBQ')
korea = f.get_series('NGDPRSAXDCKRQ')

#Combine all the single series, only take common dates using "inner"
gdp_global = pd.concat([aus, jap, us, uk, korea], join='inner', axis=1)

# Creare YoY Real GDP (divided by 4 because we have quarterly data) & Drop NA's
gdp_global = ((gdp_global / gdp_global.shift(4)) - 1).dropna(axis = 0)

gdp_global.columns = columns

# Plot
gdp_global.hvplot(title = 'Global GDP', grid = True)
```

[ ]:
```
:NdOverlay   [Variable]
   :Curve   [index]   (value)
```

[ ]:
```
# Extract the 10Y Government Bond Yields for each country (make quarterly):
aus_y = f.get_series('IRLTLT01AUM156N', frequency = 'q')
jap_y = f.get_series('IRLTLT01JPM156N', frequency = 'q')
us_y = f.get_series('IRLTLT01USM156N', frequency = 'q')
uk_y = f.get_series('IRLTLT01GBM156N', frequency = 'q')
korea_y = f.get_series('IRLTLT01KRM156N', frequency = 'q')

# Combine again
global_10y_yields = pd.concat([aus_y, jap_y, us_y, uk_y, korea_y],␣
 ↪join='inner', axis=1)

global_10y_yields.columns = columns

# Plot
global_10y_yields.hvplot(title = 'Global 10Y Yields', grid = True)
```

[ ]:
```
:NdOverlay   [Variable]
   :Curve   [index]   (value)
```

To build out the database for each country, we can use "multi-level indexing" on the columns, let's see below:

[ ]:
```
# keys allows for multi-level columns
country_df = pd.concat([gdp_global, global_10y_yields], axis=1,
                        keys=['GDP', '10Y Yields'])

country_df = pd.DataFrame(country_df)
country_df.head()
```

[ ]:
```
                 GDP                                              10Y Yields
                 AUS       JAP       USA       GBR       KOR            AUS
2000-10-01       NaN       NaN       NaN       NaN       NaN       5.880219  \
2001-01-01  0.018098  0.022046  0.021975  0.029439  0.050283   5.285803
2001-04-01  0.017376  0.009821  0.009963  0.026428  0.049944   5.805817
2001-07-01  0.026286 -0.001375  0.004892  0.025295  0.034899   5.808712
2001-10-01  0.042425 -0.014437  0.001673  0.022066  0.054131   5.560726



                 JAP   USA       GBR       KOR
2000-10-01  1.734333  5.57  5.076000  7.763333
2001-01-01  1.364000  5.05  4.791633  6.686667
2001-04-01  1.239000  5.27  5.085733  7.486667
2001-07-01  1.331333  4.98  5.058900  6.436667
2001-10-01  1.341667  4.77  4.781733  6.810000
```

1.1.5 *Info and Labels*

[ ]:
```
aus = f.get_series('NGDPRSAXDCAUQ')
jap = f.get_series('JPNRGDPEXP')
us = f.get_series('GDPC1')
uk = f.get_series('NGDPRSAXDCGBQ')
korea = f.get_series('NGDPRSAXDCKRQ')
info = f.search('NGDPRSAXDCAUQ')
info
```

[ ]:
```
                          id realtime_start realtime_end
series id
NGDPRSAXDCAUQ  NGDPRSAXDCAUQ     2024-11-27   2024-11-27  \

                                                  title observation_start
series id
NGDPRSAXDCAUQ  Real Gross Domestic Product for Australia        1959-07-01  \

              observation_end  frequency frequency_short
series id
NGDPRSAXDCAUQ      2024-04-01  Quarterly               Q  \
                                       units                units_short
series id
NGDPRSAXDCAUQ  Millions of Domestic Currency  Mil. of Domestic Currency  \

               seasonal_adjustment seasonal_adjustment_short
series id
NGDPRSAXDCAUQ  Seasonally Adjusted                        SA  \

                           last_updated popularity notes
series id
NGDPRSAXDCAUQ 2024-10-07 08:39:04-05:00         47  None
```

[ ]:

```python
import pandas as pd

# Dictionary of series IDs with country names
series_dict = {
    'Australia': 'NGDPRSAXDCAUQ',
    'Japan': 'JPNRGDPEXP',
    'United States': 'GDPC1',
    'United Kingdom': 'NGDPRSAXDCGBQ',
    'South Korea': 'NGDPRSAXDCKRQ'
}

# List to store metadata for each series
metadata_list = []

# Loop through each series and fetch metadata
for country, series_id in series_dict.items():
    # Fetch series information from FRED
    series_info = f.search(series_id)

    # Extract metadata fields and store them in a dictionary
    metadata = {
        'Country': country,
        'Series ID': series_id,
        #'Title': title,
        'Title': series_info['title'],
        'Frequency': series_info['frequency'],
        'Units': series_info['units'],
        'Description': series_info['notes'],
        'Popularity':  series_info['popularity'],
    }

    # Append metadata dictionary to list
    metadata_list.append(metadata)

# Convert the metadata list to a DataFrame
```

```python
metadata_df = pd.DataFrame(metadata_list)

# Display the metadata DataFrame
metadata_df
```

[ ]:

```
          Country      Series ID
0       Australia  NGDPRSAXDCAUQ  \
1           Japan     JPNRGDPEXP
2   United States          GDPC1
3  United Kingdom  NGDPRSAXDCGBQ
4     South Korea  NGDPRSAXDCKRQ

                                              Title
0  series id
NGDPRSAXDCAUQ    Real Gross Domestic…  \
1  series id
JPNRGDPEXP    Real Gross Domestic Pr…
2  series id
GDPC1    Real Gross Domestic Product…
3  series id
NGDPRSAXDCGBQ    Real Gross Domestic…
4  series id
NGDPRSAXDCKRQ    Real Gross Domestic…

                                          Frequency
0  series id
NGDPRSAXDCAUQ    Quarterly
Name: fre…  \
1  series id
JPNRGDPEXP    Quarterly
Name: freque…
2  series id
GDPC1    Quarterly
Name: frequency, …
3  series id
NGDPRSAXDCGBQ    Quarterly
Name: fre…
4  series id
NGDPRSAXDCKRQ    Quarterly
Name: fre…

                                              Units
0  series id
NGDPRSAXDCAUQ    Millions of Domesti…  \
1  series id
JPNRGDPEXP    Billions of Chained 20…
2  series id
```

1.1.7 Export metadata

```
[ ]: #file_path_csv = '/content/drive/MyDrive/ALFREDO_A/Dataframes/metadata_example.
      ↪csv'
     #metadata_df.to_csv(file_path_csv, index=True)
```

1.2 Data Selection

1.2.1 MENSUALES

10-Year Treasury Constant Maturity Minus 2-Year Treasury Constant Maturity

- (T10Y2YM)
- Percent, Monthly, Not Seasonally Adjusted

```
[ ]: T10Y2YM = f.get_series('T10Y2YM', frequency = 'm')
     import matplotlib.pyplot as plt
     plot = T10Y2YM.hvplot(title='10-Year Treasury Constant Maturity Minus 2-Year␣
      ↪Treasury Constant Maturity', grid=True, line_color='#000080')
     plt.rcParams.update({'font.family': 'serif', 'font.size': 12})
     plot.opts(xlabel='Date', ylabel='Percent (%)')
     plot
```

```
[ ]: :Curve   [index]   (0)
```

Federal Funds Effective Rate

- (FEDFUNDS)

- Percent, Not Seasonally Adjusted

```
[ ]: FEDFUNDS = f.get_series('FEDFUNDS', frequency = 'm')
     import matplotlib.pyplot as plt
     plot = FEDFUNDS.hvplot(title='Federal Funds Effective Rate', grid=True,␣
      ↪line_color='#000080')
     plt.rcParams.update({'font.family': 'monospace', 'font.size': 12})
     plot.opts(xlabel='Date', ylabel='Percent (%)')
     plot
```

```
[ ]: :Curve   [index]   (0)
```

M2

- (M2SL)
- Billions of Dollars, Seasonally Adjusted

```
[ ]: M2SL = f.get_series('M2SL', frequency = 'm')
     import matplotlib.pyplot as plt
     plot = M2SL.hvplot(title='M2', grid=True, line_color='#000080')
     plt.rcParams.update({'font.family': 'monospace', 'font.size': 12})
```