Using FRED® API for Economic Indicators and Data (Example)

Abstract There are different application programming interfaces (API) for accessing data directly from the web and one of the most important for economic and financial analysis is the Federal Reserve of Saint Lois (FRED).

Keywords Data · Inflation · Growth · Deflator

For the application to indicators, first, the API (application programming interface) of the Federal Reserve of Saint Lois (FRED) will be used to retrieve information The FRED is an important database for macroeconomic concepts that are necessary for understanding the market. The example will be developed in *Google Colaboratory* to exemplify how to create a notebook that can be shared.

Installing the FRED® API

The FRED® API tool is easy to use and it leads to a faster analysis of the macroeconomic situation.

For installing the API, one should visit the webpage https://fred.stlouisfed.org/docs/api/fred/ or google search the term FRED® API to obtain the result. For the use of the FRED API, it is important to ask for a key, the key is useful to the FRED because it gives representation

M. Garita, *Applied Quantitative Finance*,
https://doi.org/10.1007/978-3-030-29141-9_3

to the person using the API. To obtain the key one should visit the API Keys page at https://research.stlouisfed.org/docs/api/api_key.html. The key can be requested by creating a user and filling out the information. Once the key is received (it is usually a quick process), then it can be used for analyzing the data.

Using the FRED® API to Retrieve Data

The first process for using the FRED ® API is to install it in the environment that will be used. In this example, the Google Collaboratory will be used given that it is a platform that can be used in similarity with the use of Anaconda. For installing the FRED® API the process is as follows:

First Step

```
pip install fredapi
```

The first process is using pip install[1] which allows to retrieve the information of the FRED. In the present example by using Google Collaboratory the process is extremely simple.

Second Step

```
from fredapi import Fred
fred = Fred(api_key='XXXXXXXX')
```

In this step, once the fredapi is installed then the Fred data library can be retrieved. For this it is necessary to have the API_key which will allow the retrieval of the information.

Third Step

To retrieve information from the FRED one should know how to use the FRED website and how to retrieve the information. For the following exercise will use the information from the FRED website which is as follows: https://research.stlouisfed.org/.

[1] For more information concerning pip install please visit: https://pip.pypa.io/en/stable/reference/pip_install/.

The search bar is useful for searching the information which is needed. In this example the term *economic growth* will be used, once entered the option of *search only FRED economic data* will be chosen. This filters the result in the data that can be used as an API.

The result page based on the search will return different economic indicators which can be filtered based on the concepts such as:

– Indexes
– Prices
– Price Index
– Consumer Price Index
– Employment

The information can also be filtered by geography types, geographies, frequencies, sources, releases and seasonal adjustments. For this example, the refine of the search won´t be used and the first result on the search will be the example.

When selected the result *Consumer Price Index: Total All Items for the United States* the data is illustrated by the FRED in a line graph. Next to the name of the *Consumer Price Index: Total All Items for the United States* there is a code in parenthesis `CPALTT01USM657N`. This code is useful for retrieving the information.

It is important to visit the indicator on the webpage to understand the frequency (in this case it is monthly), the last observation and last update as well as the units.

In the Google Collaboratory or Anaconda, with the code, it is possible to retrieve the information. To retrieve the data the process is as follows:

```
consumer_price_index = fred.get_series('CPALTT01USM657N')
```

The variable *consumer_price_index* has been created and now can be used for descriptive analysis or to develop an econometric model. The command tail() can be used to analyze the latest information as follows:

```
consumer_price_index.tail()
```

```
2020-05-01    0.001950
2020-06-01    0.547205
2020-07-01    0.505824
2020-08-01    0.315321
2020-09-01    0.139275
dtype: float64
```

There is a second method which is easier and works better with the methods that are going to be seen in this book.

The first step is to install de FRED API:

```
pip installfredapi
```

The command *pip*[2] is the program that installs the packages in Python. Once the *fredapi* is installed the following process is to install *pandas_datareader*. With *pandas_datareader* it is possible to access different information such as FRED, World Bank, OECD and NASDAQ. For more information concerning the different sources and how to use them please visit: https://pandas-datareader.readthedocs.io/en/latest/remote_data.html.

```
import pandas_datareader as pdr
```

After *pandas_datareader* is installed the next part of the process is to install *datetime*.[3] *Datetime* is useful for setting dates for retrieving specific data. Given that the data can be accessed daily, the process is as follows:

```
start = datetime.datetime (2019, 1, 1)
end = datetime.datetime (2020, 9, 1)
```

The *start* is the date from which the data begins, and the *end* is where the data ends. Setting the date is important because it allows the different analysis concerning the specific time. With these settings the data can be retrieved by utilizing the code `CPALTT01USM657N` that was used before.

```
consumer_price_index = pdr.DataReader('CPALTT01USM657N', 'fred', start,
end)
```

When the variable is created, the data can be analyzed. To check that the data is correct, the next process is suggested:

```
consumer_price_index.tail()
```

```
df_cpi  = consumer_price_index.rename(columns={'CPALTT01USM657N':
'CPI'})
```

[2] For more information visit: https://pypi.org/project/pip/.

[3] For more information visit: https://docs.python.org/3/library/datetime.html.

DATE	*CPALTT01USM657N*
2020–05-01	0.001950
2020–06-01	0.547205
2020–07-01	0.505824
2020–08-01	0.315321
2020–09-01	0.139275

What can be observed is that the last data in FRED is from the first of May 2020. Given that the code `CPALTT01USM657N` could be difficult for others to understand when sharing the Collaboratory, the title of the column must be changed:

```
df_cpi  = consumer_price_index.rename(columns={'CPALTT01USM657N':
'CPI'})
```

The Gross Domestic Product

When analyzing economic cycles, one of the most important variables is the Gross Domestic Product better known as GDP. The Gross Domestic Product can be measured through different approaches but the most useful is the expenditure approach. The expenditure approach is based on the following formula:

$$GDP = C + I + G + (X - M)$$

C = consumption
I = investment
G = Government expenditure
X = Exports
M = Imports

The approach behind the *GDP* equation is to understand how the economy is funded and from there understand how it works. The theory is based that the households offer funds to the banks as savings or to the companies as investments. Households also offer labor to the companies in returns of funds, salaries, that can be used for consuming the products created by the companies or they can be invested and/or saved. Also, from the salary, the government charges taxes which are used for public goods and services. Finally, exports and imports are based on the exposure of a country to other economies. In this sense, if a country didn´t

share economic activity with other countries there wouldn´t exist exports and imports, just consumption, investment and government expenditure.

In order to understand the economic cycle, the GDP will be analyzed. The Real GDP will be used because it is inflation adjusted and the nominal GDP is established on a base price. The code for the Real GDP is GDPC1). The data for the GDP is in billions of dollars, seasonally adjusted at an annual rate and with a quarterly frequency.

```
gdp = pdr.DataReader('GDPC1', 'fred', start, end)
```

Once the data is retrieved, a change can be made for a more useful analysis. For this change, the *pct.change()*[4] will be used to analyze the growth of the GDP. Using *pct.change()* is simple, when the parenthesis is left blank, the program assumes that there is one period, which is the process that will be used.

```
gdp_change = gdp.pct_change() * 100
```

DATE	*GDPC1*
2019–07-01	0.636915
2019–10-01	0.586232
2020–01-01	–1.262655
2020–04-01	–8.986117
2020–07-01	7.403492

The name of the column can be changed with the process used before.

```
gdp_change = gdp_change.rename(columns={'GDPC1': 'GDP %'})
```

DATE	*GDPC1*
2019–04-01	0.370716
2019–07-01	0.636915
2019–10-01	0.586232
2020–01-01	–1.262655
2020–04-01	–8.986117
2020–07-01	7.478741
2019–04-01	0.370716

[4] For more information please visit: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.pct_change.html.

The change in the GDP is important to understand the business cycle. One of the most usual question considering macroeconomics is if the nominal or real growth should be used when analyzing the GDP.

The nominal GDP reflects the growth without the inflation, meaning that it considers the output of production by the country and it analyzes its growth when compared to previous year. Usually, GDP measuring has a base GDP year which establishes the growth for the nominal GDP. The main difference is that when using the real GDP is that it reflects the growth adjusted, leading to the GDP Deflator.

The Gross Domestic Product Price Deflator

The GDP deflator, also known as implicit price deflator for GDP measures the percentual difference between the nominal GDP and the real GDP. It is a useful tool for understanding the change in price because it helps the investor understand if the growth has been motivated by production or by prices. The equation is as follows:

$$GDPDeflator = \frac{nominalGDP}{realGDP} \times 100$$

To calculate the *deflator* of the GDP the *sum* function will be used. The sum function sums all the data points in the list. The first step is to establish a base year, in this case the year will be 2017.

```
start = datetime.datetime (2017, 1, 1)
end = datetime.datetime (2017, 12, 31)
```

```
nominal_GDP = pdr.DataReader(['GDP'],'fred', start, end)
```

The comparison will be the year 2019.

```
start = datetime.datetime (2019, 1, 1)
end = datetime.datetim   2019, 12, 31)
real_GDP = pdr.DataReader( 'GDPC1'],'fred', start, end)
```

Once the real GDP and the nominal GDP had been set into a variable, the following process is to create the sum of the production in the year.

```
ngdp_sum = nominal_GDP.sum()
ngdp_sum = int (ngdp_sum)


rgdp_sum =   real_GDP.sum()
rgdp_sum = int (rgdp_sum)
```

Notice that in the process the variables have been converted to integers with the function *int*. The reason for this is that if the data is divided without converting it into an integer, then the problem is that Python interprets the data as two different series and cannot unite them for a process.

With the data converted into integers, then it can be divided.

```
deflator = ngdp_sum/rgdp_sum * 100

102.36361731660686
```

The deflator for 2019 in comparison with the nominal GDP in 2017, the prices have grown by 125%, which reflects that the growth has grown by boost of prices. Considering that investment it is important to have a growth higher than inflation, following the deflator is an important indicator.

Understanding the Process into the Basics

One of the most important process is the use of *sum* instead of the (+) sign seen in the understanding numbers in Python. Basic arithmetic functions can be interchanged with the commands such as *sum*() or *prod*(). These functions are extremely useful when analyzing the data and developing arithmetic operations in data frames.

When understanding the type of numbers Python supports and registers, a number such as the variable *deflator* can be identified with the command *type*. One example as follows:

```
type(deflator)
float
```

In the example above the deflator is a float given that it has decimal numbers. The number could be changed to an integer by using the *int()* function.

```
int(deflator)
102
```

The variables can be changed and therefore it is important to understand what each variable represents. For example, if the variables are developed by a data frame and there is a difficulty when adding them, the variables can be modified to an integer so that they can be summed. An example as follows:

```
deflator = (int (nominal_GDP.sum()))/ (int (real_GDP1.sum())) * 100
```

The process above changes the *sum* of the variable *nominal_GDP* into an integer so that it can be divided into the sum of the integer of the *real_GDP1* variable.

Considering data structures, given that the information will be retrieved from an API during this book the result when analyzing list is a DataFrame.

```
type(real_GDP)
pandas.core.frame.DataFrame
```

As mentioned before, a DataFrame is a library in *pandas* that allow us to create a more integral analysis. The API used during this book converts the data into a DataFrame automatically and there is not a necessity to apply the commands learned before.

As seen in the chapter before, this creates an easy access to the functions and on how to work with them. For a better example, let us use more variables.

Comparing GDP

When analyzing growth, it is important to compare how the growth of the country that is being analyzed compares to the other countries. In this process, a graph will be developed to understand how the different countries behaved.

Installing packages:

```
import pandas_datareader as pdr
import panda  as pd
import datetime
import matplotlib.pyplot as plt
```

Setting a date and retrieving information based on the FRED code:

```
start = datetime.datetime (2015, 1, 1)
end = datetime.datetime  2020, 12, 31)



gdp_comparison = pdr.DataReader(['BRAGDPNQDSMEI ,
'CHNGDPNQDSMEI','USAGDPNQDSMEI ],'fred', start, end)
```

Modifying the names on the country list:

```
gdp_comparison =
gdp_comparison.rename(columns= 'BRAGDPNQDSM   :'Brazil','CHNGDPNQDSM
:'China', 'USAGDPNQDSMEI': 'USA'})

gdp_comparison.head(
```

DATE	*Brazil*	*China*	*USA*
2015–01-01	1.488903e + 12	1.511379e + 13	4.500850e + 12
2015–04-01	1.485916e + 12	1.685497e + 13	4.555894e + 12
2015–07-01	1.504503e + 12	1.765977e + 13	4.586856e + 12
2015–10-01	1.516466e + 12	1.925729e + 13	4.594701e + 12
2016–01-01	1.532747e + 12	1.624100e + 13	4.617539e + 12

Changing the data to percentage change:

```
gdp_change = gdp_comparison.pct_change() * 100

gdp_change.head()
```

Dropping NA for creating a chart:

```
gdp_change = gdp_change.dropna()
gdp_change.head()
```

DATE	*Brazil*	*China*	*USA*
2015–04-01	–0.200587	11.520472	1.222980
2015–07-01	1.250849	4.774853	0.679603
2015–10-01	0.795178	9.046097	0.171021
2016–01-01	1.073628	–15.663107	0.497056
2016–04-01	2.034344	11.209285	1.007306

As seen before, this is a DataFrame in which the columns can be edited, the numbers can be dropped and the information changed. This is important when applying certain loops as well as when developing a portfolio, which will be seen in future chapters.