

25 DE SEPTIEMBRE DE 2021

ANÁLISIS DE DATOS DE TIPOS DE CAMBIO

PABLO RAMOS DEL BUSTO
NEREIDA GONZÁLEZ LÓPEZ

ANÁLISIS DE DATOS DE TIPOS DE CAMBIO

1. INTRODUCCIÓN

Es importante entender que este documento solo intentará explicar el código mostrado en el [artículo](#). Este documento tan solo será orientativo y no enseñará nada de economía.

Este documento trata de explicar de manera sencilla y breve como llevar a cabo el tratamiento de datos de mercados obtenidos a través de una API de la [siguiente página](#), para su análisis automático a través de Python.

Para entender la parte de los datos que se ven hizo falta la ayuda de [Nereida González López](#).

2. TIPOS DE CAMBIO EN PYTHON

Primero se va a explicar cómo obtener datos a través de la API *financialmodelingprep*, que ofrece una gran variedad de datos financieros de forma gratuita, y graficar tipos de cambio con Python. Se construirá un código para recuperar el precio histórico de los tipos de cambio de varias divisas y se graficarán usando la librería *matplotlib*.

2.1. PAQUETES

Antes de empezar aconsejo crear un *workspace* propio en Python y un entorno virtual en donde instalar los paquetes necesarios, así como realizar los cambios de configuración que se crean convenientes. Para ello aconsejo utilizar el siguiente [documento como apoyo](#).

Los paquetes que se van a descargar serán del repositorio [PyPi](#):

- [Pandas](#)
- [Requests](#)
- [Matplotlib](#)

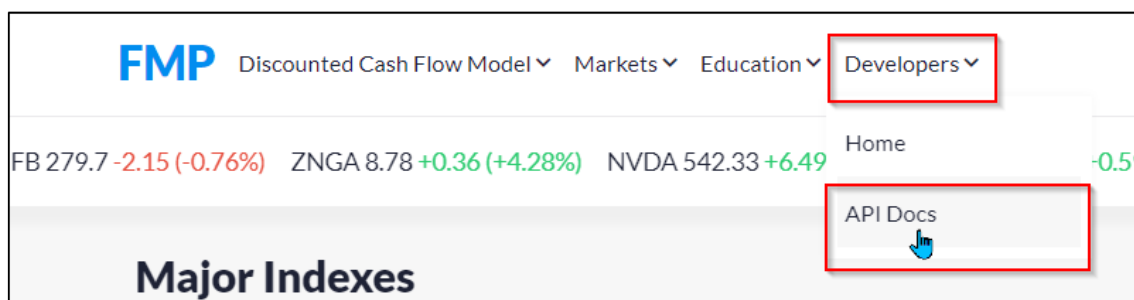
```
#Importamos los paquetes que vamos a necesitar
import requests
import pandas as pd
import matplotlib.pyplot as plt
import json
```

2.2. LLAMADA A LA API

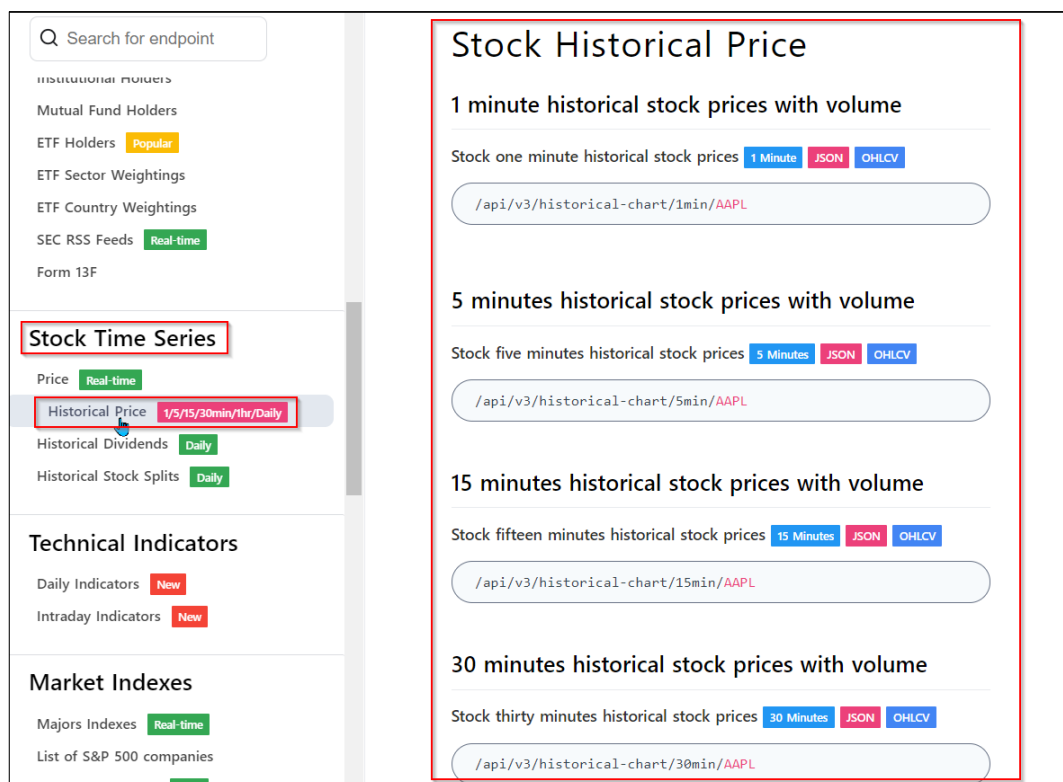
Después de importar los paquetes se realizará una llamada a la API para recuperar el histórico de los últimos años para algunos tipos de cambio. Para ello se programará un bucle donde, en cada iteración, se obtendrá el precio histórico de una divisa.

2.2.1. CONFIGURACIÓN DE LA API

Llegados a este punto, se debe comprender el funcionamiento de la API de la página web. Para acceder a los documentos API, donde se indica como obtener los datos, se hará clic en el menú para *developers*.



Este caso nos ocupa los precios históricos, por lo que buscamos en la lista del margen izquierdo la sección correspondiente. Una vez ahí, se explica cómo obtener diferentes medidas para esta función. En el caso que trata este documento serán los **Daily historical Price** dentro de **stock time series**.



Antes de empezar a extraer los valores es necesario poseer un **API Key** que permita extraer los valores que se desean. Para realizar esto, se debe ingresar una cuenta en la siguiente [dirección](#).

Financial Modeling Prep API Documentation

Getting Started

We are the most accurate **financial data API** out there.

To make it simple we are a **free stock API** | **historical data API** | **financial statements API**

We update our financial statements in real time, every statements is audited, standardized, and up to date.

We cover NYSE, NASDAQ, AMEX, EURONEX, TSX, INDEXES, ETFs, MUTUAL FUNDS, FOREX and CRYPTO.

We have **real time stock price**, we cover the fundamental data part of the stocks via providing income statement, balance sheet statement and cashflow statement **quarterly and annually**.

We have **1 minute, 15 minutes, 30 minutes, 1 hour** and daily historical stock prices.

Get your Free API Key Today!



Get my API KEY here

Our goal is to provide the best service that we can. Thank you for all the support and suggestions - it really helps us maintain and improve the API.

Posteriormente, en la variable **forexJSON**, se recorre por cada divisa todo el histórico (**historical**). Se almacenan dos variables: el precio de cierre con su fecha correspondiente. Esto se almacenará en el **dict** `Exchange_rates_Python[currency]`, donde **currency** equivale a la divisa. Todo este proceso se puede ver en la imagen inferior.

```
#Almacenamos en el array los valores deseados
exchange_rates_Python[currency] = {}

for item in forexJSON['historical']:
    adj_close = item['adjClose']
    trade_date = item['date']
    exchange_rates_Python[currency][trade_date] = adj_close
```

Si se desean realizar algunas pruebas tan solo añadiendo al final del **for** una de las divisas que está en la lista **currencies** y una de las fechas que se pueden ver en el JSON se puede ver el formato.

```
print(exchange_rates_Python[divisa][fecha])
```

```
21 #Almacenamos en un dict los valores deseados
22 exchange_rates_Python[currency] = {}
23
24 for item in forexJSON['historical']:
25     adj_close = item['adjClose']
26     trade_date = item['date']
27     exchange_rates_Python[currency][trade_date] = adj_close
28
29 print(exchange_rates_Python['EURUSD']['2020-12-03'])
30
```

2.5. ORDENAR DATOS

Posteriormente lo que se realiza es la conversión de los datos al tipo **DataFrame** usando el paquete Pandas. Como los datos están en forma **dictionary** se debe utilizar la función **from_dict**.

	EURUSD	CHFUSD=X	AUDUSD	GBPUSD
2020-12-08	1.211974	1.122536	0.742710	1.335952
2020-12-07	1.212136	1.121403	0.743157	1.341742
2020-12-04	1.214565	1.122624	0.743898	1.345497
2020-12-03	1.211460	1.117805	0.741801	1.337077
2020-12-02	1.207438	1.111716	0.738089	1.342264
...
2015-12-14	1.098394	1.016467	0.716486	1.519896
2015-12-11	1.094811	1.012146	0.727220	1.516093
2015-12-10	1.101795	1.016984	0.723537	1.518395
2015-12-09	1.089396	1.007252	0.722507	1.501006
2015-12-08	1.083400	0.999600	0.725900	1.505027

Por ultimo y como se ve en la imagen superior se convierten los valores índices, es decir, los valores “cabecera” de las filas en formato tiempo por medio de la función `to_datetime`.

```
#Salimos del bucle inicial de currencies y convertimos los datos en DataFrame
currenciesDataFrame = pd.DataFrame.from_dict(exchange_rates_Python)

#Convertimos los valores "cabeceros" de las filas a formato tiempo
currenciesDataFrame.index = pd.to_datetime(currenciesDataFrame.index)
```

2.6. REPRESENTACIÓN DE DATOS

Para acabar y representar por medio de figuras los valores, se utilizará el paquete **matplotlib**, importado al inicio con el resto de los paquetes.

En la imagen representada inferiormente se seleccionarán con la función `iloc` los valores de los últimos 30 días. Esto se debe a que se pone el valor después de los ‘:’ (desde : hasta) de esta manera indicamos hasta donde. Por el contrario, si se llega a poner el valor previo a los ‘:’ se indicaría desde donde.

Se definen el número de gráficos que se representarán en la misma figura por medio de la función `subplot`. Posteriormente, se van creando las figuras por cada tipo de divisa y situando está en la posición deseada, dentro de la figura creada con la función `subplot`.

```
#Representamos las figuras
fig, axes = plt.subplots(nrows=2, ncols=2)

currenciesDataFrame.iloc[-30:].plot(ax=axes[0,0])
axes[0,0].set_title(currencies[0])

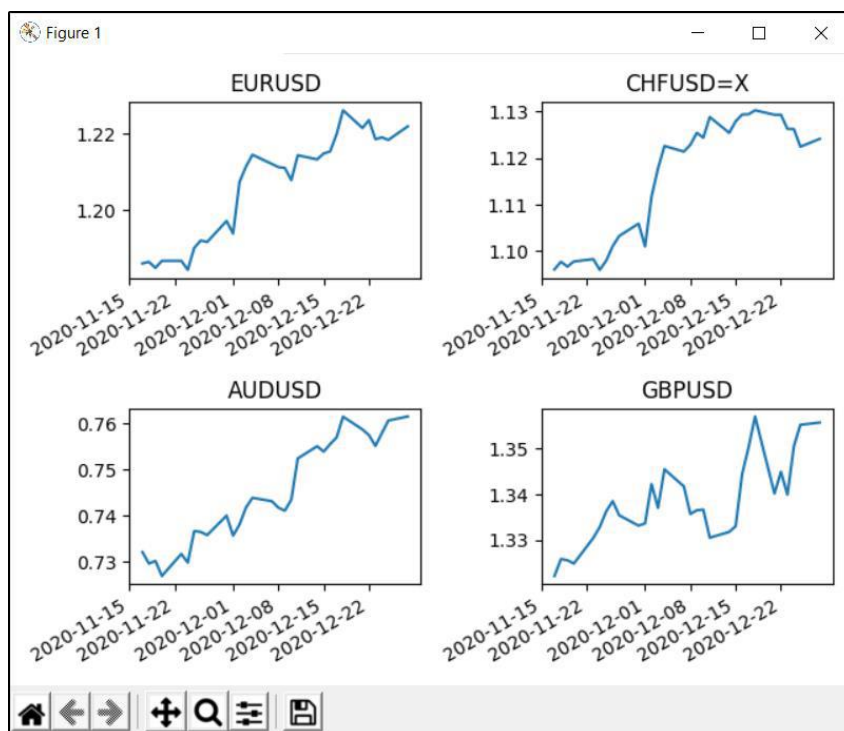
currenciesDataFrame.iloc[-30:].plot(ax=axes[0,1])
axes[0,1].set_title(currencies[1])

currenciesDataFrame.iloc[-30:].plot(ax=axes[1,0])
axes[1,0].set_title(currencies[2])

currenciesDataFrame.iloc[-30:].plot(ax=axes[1,1])
axes[1,1].set_title(currencies[3])

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

Una vez escrito adecuadamente todo el código con la función **show** se representarán las figuras en la forma definida. Serán una figura con cuatro gráficos, uno por cada divisa.



3. RECURSOS

Para la realización de este documento se utilizó como recursos las siguientes páginas web:

- [Impact of exchange rates in companies | by Jose Manu \(CodingFun\) | Towards Data Science](#)
- [Financial Modeling Prep – Home](#)
- [PyPI · The Python Package Index](#)
- [Python Tutorial \(w3schools.com\)](#)
- [Pandas documentation — pandas 1.2.0 documentation \(pydata.org\)](#)

Además, si se desea tener acceso completo al código se podrá consultar [aquí](#).