Proyecto 1

Python para Finanzas
Universidad Adolfo Ibáñez

Francisco Ibáñez

Julio 5, 2021

El código escrito para resolver los problemas planteados en este documento deben seguir el estilo de programación en Python **PEP 8**[1], especialmente en el caso de los comentarios. Al comienzo de cada archivo deben crear un *block comment* explicando lo que hace el programa. El resto del programa debe ser correctamente explicado utilizando suficientes (pero no excesivos) *inline comments*. Las funciones creadas deben contener un *documentation string* (a.k.a. docstring) detallando qué hace la función, cuáles son los inputs esperados, el data type de éstos, y el output esperado de la función. En términos del formato de entrega, problemas 1 y 2 deben estar dentro de un archivo llamado **funciones.py** y el problema 3 dentro de **analisis.py**.

Paquetes permitidos: numpy, csv.

1 Importación de datos

Escriba una función llamada **importar_serie()** que construya un NumPy array utilizando las series de tiempo guardadas en un archivo csv. La función debe aceptar un solo parametro **str** que indique la ubicación del archivo .csv.

La importación se puede llevar a cabo utilizando el paquete **csv** y la cláusula **with** de la siguiente forma:

```
import csv

with open('archivo.csv', 'r') as f:
    data = list(csv.reader(f, delimiter=","))
```

El resultado de éste será una lista de tamanño igual al numero de filas en el csv, donde cada elemento es una lista de tamaño igual al número de columnas (lista de listas). Esto requerirá formateo adicional.

[1] Disponible en https://www.python.org/dev/peps/pep-0008/

Una serie de tiempo se caracteriza por asociar observaciones con un índice temporal de fechas. Dado que NumPy no permite hacer esta asociación, la función deberá entregar tres outputs:

1. **data**: NumPy array de dimensión $t \times n$ que contiene las obervaciones de las n series de tiempo en el archivo. Cada una de las observaciones debe ser un **float**.

2. **indice**: NumPy array de dimensión t que contiene las fechas asociadas a las obervaciones el archivo. Cada una de las fechas debe tener data type **numpy.datetime64**[2].

3. **etiquetas**: Lista de **str** con los nombres de las columnas de las variables (no-fechas).

La función debe asegurarse que tanto las fechas como las observaciones estén ordenadas de forma ascendente (más reciente al final).

2 ¿Stata quién?

Escriba una función llamada **regresa()** que reciba los siguientes parametros:

1. **dependiente**: NumPy array de dimensión $t \times 1$ que contiene las observaciones de la variable que queremos explicar.

2. **independiente**: NumPy array de dimensión $t \times k$ que contiene las obervaciones de las k variables explicativas.

3. **etiquetas**: lista de **str** que detalla el nombre de las variables.

4. **const**: boolean que define si la regresión debe ser corrida con o sin constante. Valor por defecto True.

5. **confianza**: float, nivel de confianza nutilizado para la estimación del intervalo de confianza y el p-value. Valor por defecto 0.05.

6. **mudo**: boolean que activa el modo silencioso. Si es True, la función no imprimirá los resultados de la regresión en la terminal. Valor por defecto False.

La función debe ser capaz de revisar que el data type de los 4 inputs correspondan a los indicados[3]. Si data types equivocados son ingresados, o las dimensiones de las matrices no coinciden, la función debe arrojar un error[4] y detenerse[5].

La función debe entregar los siguientes outputs:

1. **coeff**: NumPy array de dimensión k o $k+1$ que contiene los coeficientes de la regresión (incluyendo constante si así fue definido).

[2]Ver https://numpy.org/doc/stable/reference/arrays.datetime.html
[3]isinstance(variable, datatype)
[4]raise ValueError('Mensaje de error')
[5]return

2. **r2**: float detallando el R^2 de la regresión.

3. **p-value**: NumPy array de dimensión k o $k+1$ que contiene el p-value correspondiente a cada parámetro.

Finalmente, la función debe imprimir en la consola una tabla similar a la que se ve abajo (asumiendo que **mudo=False**):

```
Resumen de la regresion
-----------------------

Numero de observaciones:          XX
Grados de libertad:               XX
R-cuadrado:                       XX

nombre  |  coef. |  error std  |  p-value  |  int. de confianza XX%  |
=========================================================================
var 1   |  XX    |  XX         |  XX       |  [XX  ,  XX]            |
var 2   |  XX    |  XX         |  XX       |  [XX  ,  XX]            |
...     |  ...   |  ...        |  ...      |  ...                    |
var n   |  XX    |  XX         |  XX       |  [XX  ,  XX]            |
```

hint: tenga en mente que la tabla es ilustrativa, por lo que podemos limitar el número de decimales y caracteres para mantener el formato constante y así facilitar su automatización. También recuerde usar las etiquetas para llenar el campo 'nombre' en la tabla.

3 A la práctica!

En un nuevo archivo .py importe las funciones creadas para resolver los dos problemas anteriores y úselas para cargar la data contenida en los archivos **goog.csv** y **fama_french.csv**. El primer archivo contiene los retornos diarios de Google, mientras que el segundo contiene los retornos diarios de los 3 factores de Fama y French, más el retorno del instrumento libre de riesgo. Cualquier modificación a la data debe hecha en el *script* mismo y no fuera de este.

1. Regresione el exceso de retorno de Google contra el premio por riesgo de mercado (Mkt-RF)

2. Regresione el exceso de retorno de Google contra los retornos de los tres factores

3. Analice las diferencias en los resultados

Todo analisis/conclusión debe ser hecho por medio de bloques de comentarios dentro del código.