Cousas básicas de python para deixar de usar excel

paula.antelo@usc.es

26 de marzo do 2024

1. Librerías básicas de python

Librerías de python básicas que necesitaredes para análise de datos (pode que teñades que instalar algunha, podedes facelo con pip ou conda):

- **numpy**: para operacións numéricas en xeral.

  Documentación: `https://numpy.org/doc/stable/user/quickstart.html`

- **matplotlib**: para gráficos.

  Documentación: `https://matplotlib.org/stable/users/explain/quick_start.html`

- **pandas**: para manexo de datos.

  Documentación: `https://pandas.pydata.org/docs/user_guide/index.html#user-guide`

- **scipy**: para estatística.

  Documentación: `https://docs.scipy.org/doc/scipy/tutorial/index.html#`

Consello: é un moi bo hábito ler a documentación das librerías que usedes (non todo, o que vos vaia facendo falta). Por exemplo, na documentación de funcións ides ver un apartado que explica os parámetros que lle tedes que pasar á función (o que vai entre paréntesis) e o que vos devolve (returns). Ademais, ao final adoita haber exemplos útiles.

Exemplo de como importar as librerías nun programa (con un alias: np, plt, sc, pd):

```
import numpy as np
import matplotlib.pyplot as plt
import scipy as sc
import pandas as pd
```

2. Ler os datos tomados no laboratorio

- Copialos directamente nunha lista ou array de numpy. Exemplo[1]:

[1] Se facedes esto traspoñede (pegado especial - traspoñer) as columnas a filas nun excel para poder copiar as filas despois. Ademais, podedes usar buscar e reemplazar en spyder para cambiar comas por puntos.

```
x = np.array([1,2,4,5,7,9,10,13])
y = np.array([1.2,2.3,4.3,5.5,7.1,9.2,9.8,12.5])
```

- Lelos dende un arquivo excel no que teñades en columnas os datos con nomes de columnas x e y (por exemplo). Para iso usamos unha función de pandas (documentación: `https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_excel.html`).

```
datos = pd.read_excel('tecnicasI.xlsx')
# datos é un DataFrame de pandas, por pantalla apareceranos como
# unha táboa. Podemos acceder ás súas columnas usando [] co nome
# da columna. Os DataFrame permiten facer operacións con eles e
# moitas outras cousas, pero se non vos queredes complicar
# podedes convertilo a un array de numpy facendo np.array(datos).

x = np.array(datos['x']) # convirto a columna 'x' a un array de numpy
y = np.array(datos['y'])
```

	A	B	C
1	x	y	
2	1	1.2	
3	2	2.1	
4	3	2.9	
5	4	4.1	
6	5	5.05	
7	6	6.2	
8	7	6.6	
9	8	8.15	
10	9	9.2	
11	10	10.3	
12			
13			

Figura 1: Exemplo dun excel do que ler os datos.

3. Axustes lineais

Queremos axustar os datos a unha recta tipo $y = bx + a$. Temos que obter: b, a, s(b), s(a), r^2, e s. As opcións que temos son as seguintes:

- Escribir as **ecuacións directamente** (bastante máis factible ca escribilas en excel[2]).
- Usar a función de **scipy linregress**. Esta só serve **para axustes lineais e sen ponderar** (documentación: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.linregress.html)[3]:

```
axuste = sc.stats.linregress(x, y)
b,  a  = axuste.slope,  axuste.intercept
sb, sa = axuste.stderr, axuste.intercept_stderr
r2     = axuste.rvalue**2   # esta función tamén devolve o r2 do axuste
```

[2] De verdade que si.

[3] Comentade o código sempre, aínda que só sexan un par de liñas, axudan moito a entender o código cada pasa un tempo que non traballades nel.

- Usar a función de **scipy curve fit**. Esta tamén serve **para axustes non lineais e para axustes ponderados**. Se ledes a documentación atoparedes en que parámetro tedes que pasarlle a incerteza no eixo y para facer axustes ponderados: `https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html`

```
def recta(x, b, a):   # función á que queremos axustar os datos
    y = x * b + a
return y

popt, pcov = sc.optimize.curve_fit(f = recta, xdata = x, ydata = y)
b,  a  = popt
sb, sa = np.sqrt(np.diag(pcov))  # raíz cadrada dos elementos da diagonal
                                 # da matriz de covarianza

# curve_fit non devolve valores para r2 e s, pero podemos usar outra función
# serve para axustes sen ponderar !!
r2     = sc.stats.pearsonr(x, y).statistic**2
```

Para calcular a desviación típica do axuste podedes buscar unha función dalgunha librería que vola calcule ou senón introducir a expresión en python vós directamente (é sinxela unha vez temos a e b). O mesmo para o r^2 dos axustes ponderados.

4. Graficar os datos

Exemplo de como graficar un axuste lineal:

```
fig = plt.figure(); ax = fig.add_subplot()
ax.set_xlabel('x'); ax.set_ylabel('y')
x_plot = np.linspace(np.min(x), np.max(x), 30)   # xero valores para o eixo x

ax.scatter(x, y, color = 'b')
ax.plot(x_plot, x_plot * b + a, linestyle = '-', marker = '', color = 'r')

fig.savefig('axuste.png')
```

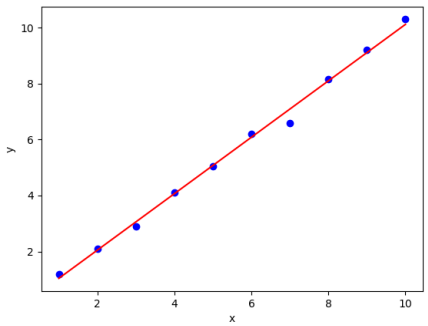

Figura 2: Datos experimentais e axuste lineal.

5. Gardar os datos nun excel de novo

A librería pandas permite manexar DataFrames, que son dalgún xeito táboas de datos. Podemos crear un DataFrame a partir dun array de numpy, poñerlle nomes ás columnas, e gardalo nun arquivo excel de novo[4] (documentación: `https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.to_excel.html#pandas.DataFrame.to_excel`).

```
datos_gardar = pd.DataFrame(np.array([x,y,y**2]).T, columns = ['x', 'y', 'y2'])

datos_gardar.to_excel('datos_gardar.xlsx')
```

```
In [43]: datos_gardar
Out[43]:
      x      y        y2
0   1.0   1.20    1.4400
1   2.0   2.10    4.4100
2   3.0   2.90    8.4100
3   4.0   4.10   16.8100
4   5.0   5.05   25.5025
5   6.0   6.20   38.4400
6   7.0   6.60   43.5600
7   8.0   8.15   66.4225
8   9.0   9.20   84.6400
9  10.0  10.30  106.0900
```

Figura 3: Saída dun DataFrame de pandas.

Para escribir a memoria

Altamente recomendable empregar Latex! Unha vez que o aprendades a manexar, veredes que para isto é moito máis rápido e eficiente ca Word.

- Ecuacións: Podedes atopar como escribilas no enlace `https://manualdelatex.com/tutoriales/ecuaciones`. Ademais, `https://mathpix.com` é unha aplicación á que lle podedes pasar unha imaxe da ecuación que queredes escribir e devólvevola en código latex (útil para ecuacións moi longas ou se non sabedes como escribir algunha).
- Imaxes: Na versión tex desde documento tedes exemplos de como engadir imaxes.
- Táboas: O máis sinxelo é usar o xerador de táboas `https://www.tablesgenerator.com`[5].
- Texto: Na versión tex deste documento tedes exemplos de como crear seccións, facer enumeracións, notas ao pé, subliñar ou poñer en negriña e incluso escribir código de python.
- Para escribir en Latex recoméndovos empregar Overleaf (`https://www.overleaf.com/project`). Non necesitades instalar nada, soamente conexión a internet. Podedes subir directamente un arquivo comprimido zip cun documento tex e as imaxes ou outros arquivos asociados (por exemplo o que vos subín xunto con este documento). Ademais, podedes compartir os proxectos que teñades con outras persoas (moi útil cando teñades que facer memorias en parellas)[6].

[4] En verdade recoméndovos gardar os datos en formato csv, tamén existe para iso unha función en pandas.

[5] Tamén existe unha maneira de obter unha táboa de datos en formato Latex directamente dende python, empregando a función tabulate da librería tabulate co argumento tablefmt = 'latex'.

[6] Todos os anos organízase un curso de Latex que vos recomendo moito, ademais danvos 1 crédito