Facultad de Matemática, Astronomía, Física y Computación, U.N.C.

Computación

Aula Virtual: https://famaf.aulavirtual.unc.edu.ar/course/view.php?id=747

Resguardo tutoriales: https://www.famaf.unc.edu.ar/∼moreschi/docencia/Computacion/

Tutorial Problemas 4 de la Guía N° 6

Problema 4: Determinación del parámetro de expansión y aceleración del universo:

El sistema cosmológico está gobernado por el fenómeno de expansión del universo. La ley de Hubble indica que en escalas grandes vale aproximadamente la regla

$$v = H_0 r;$$

donde v es la velocidad de recesión de las galaxias, r la distancia a ellas y H_0, que lo denominamos el parámetro de Hubble, describe la expansión. Para simplificar obviaremos las discusiones de las unidades en esta ocasión.

Aquí simularemos la problemática de extraer información de datos cosmológicos. Primeramente generaremos datos a partir de una ley exacta

$$y_{\text{ideal}} = H_{00} x;$$

con $H_{00} = 74.$; a la que agregaremos ruido gausiano que aumenta con la distancia.

Luego plantearemos varias funciones de ajuste, dadas por:

$$f_1(x) = ax,$$

$$f_2(x) = ax + bx^2,$$

$$f_3(x) = ax + bx^2 + cx^3$$

y

$$f_4(x) = ax + bx^2 + cx^3 + dx^4.$$

Finalmente usaremos herramientas de PYTHON para hacer ajustes de estas funciones por cuadrados mínimos, de los datos 'observacionales'.

a) En un archivo de comandos PYTHON defina las funciones descriptas anteriormente, pero donde las constantes aparecen como argumento de la función; por ejemplo:

```
def f4(x,a,b,c,d):
    return a*x + b*x**2 + c*x**3 + d*x**4
```

b) Genere los datos 'observacionales' con los siguientes comandos:

```
xmin = 0.
xmax = 1.4
anc = 10.

# elección de la semilla ----------------------------------------
np.random.seed(11)

# elección de Nrep ----------------------------------------------
```

```
Nx = 1000
x = np.linspace(xmin, xmax, Nx)
H00 = 74. # generamos datos con este valor ideal de H_0
ideal = H00*x
ancho = x * anc
y = ideal + np.random.normal(loc=x,scale=ancho)
```

donde estamos usando la función NORMAL, del paquete RANDOM de la librería NUMPY; a la que se le puede dar la localización del valor medio (loc) y su deviación estándar (scale).

c) Genere un gráfico de y vs x (eje vertical vs. eje horizontal) y guarde el mismo en el archivo: 'graficos/p4-observaciones-H0-'+str(Nx)+'.png'

d) Calcule los ajustes de los parámetros de las funciones con los comandos:

```
from scipy.optimize import curve_fit # preferentemente en el encabezamiento

popt1, pcov1 = curve_fit(f1, x, y)
print('popt1 =',popt1)
print('pcov1 =',pcov1)

popt2, pcov2 = curve_fit(f2, x, y)
print('popt2 =',popt2)
print('pcov2 =',pcov2)

popt3, pcov3 = curve_fit(f3, x, y)
print('popt3 =',popt3)
print('pcov3 =',pcov3)

popt4, pcov4 = curve_fit(f4, x, y)
print('popt4 =',popt4)
print('pcov4 =',pcov4)
```

e) Genere un gráfico con los datos y las cuatro funciones ajustadas, mostrando el valor de los coeficientes, con los comandos:

```
plt.plot(x , y,'o-', label='observaciones')
plt.plot(x, f1(x, *popt1), 'r-', linewidth=8.0,
         label='fit: $H_0$=%5.3f' % tuple(popt1))
plt.plot(x, f2(x, *popt2), 'y-', linewidth=4.0,
         label='fit: $H_0$=%5.3f, acel=%5.3f' % tuple(popt2))
plt.plot(x, f3(x, *popt3), 'c-',  linewidth=2.0,
         label='fit: $H_0$=%5.3f, acel=%5.3f, c=%5.3f' % tuple(popt3))
plt.plot(x, f4(x, *popt4), 'k-',  linewidth=1.0,
         label='fit: $H_0$=%5.3f, acel=%5.3f, c=%5.3f, d=%5.3f' % tuple(popt4
    ))
```

y guarde el gráfico en el archivo con nombre: 'graficos/p4-observaciones+ajuste-H0-'+str(Nx)+'.png'

f) ¿Hay diferencia cualitativa en las curvas graficadas?

g) ¿Es posible con este grado de ruido observacional recuperar el valor original de H_0? ¿Con qué error?

h) Según estos datos, ¿Se puede decir que está acelerado el universo? Eventualmente: ¿Qué signo tiene la aceleración?

Tutorial:

- Guarde en el archivo p4.py las siguientes instrucciones:

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit


def f1(x,a):
    return a*x

def f2(x,a,b):
    return a*x + b*x**2

def f3(x,a,b,c):
    return a*x + b*x**2 + c*x**3

def f4(x,a,b,c,d):
    return a*x + b*x**2 + c*x**3 + d*x**4

xmin = 0.
xmax = 1.4
anc = 10.

# elección de la semilla ----------------------------------------
np.random.seed(11)

# elección de Nrep ----------------------------------------------------
Nx = 1000
x = np.linspace(xmin, xmax, Nx)
H00 = 74. # generamos datos con este valor ideal de H_0
ideal = H00*x
ancho = x * anc
y = ideal + np.random.normal(loc=x,scale=ancho)

print('x =',x)
print('y =',y)
print('len(x) =',len(x))
print('len(y) =',len(y))
print('min(y) =',min(y))
print('max(y) =',max(y))
print('max(y)-min(y) =',(max(y)-min(y)) )
numbines = (max(y)-min(y))
print('      numbines =',numbines)

plt.figure( figsize=(10, 7.5) )
plt.title('Observaciones para $H_0$, '+str(Nx)+' puntos')
plt.xlabel('x')
plt.ylabel('y')
plt.grid()
plt.plot(x , y,'o-', label='observaciones')
plt.legend(loc="best")
plt.savefig('graficos/p4-observaciones-H0-'+str(Nx)+'.png', dpi=100)
plt.show()
```

```
52
53
54 popt1, pcov1 = curve_fit(f1, x, y)
55 print('popt1 =',popt1)
56 print('pcov1 =',pcov1)
57
58 popt2, pcov2 = curve_fit(f2, x, y)
59 print('popt2 =',popt2)
60 print('pcov2 =',pcov2)
61
62 popt3, pcov3 = curve_fit(f3, x, y)
63 print('popt3 =',popt3)
64 print('pcov3 =',pcov3)
65
66 popt4, pcov4 = curve_fit(f4, x, y)
67 print('popt4 =',popt4)
68 print('pcov4 =',pcov4)
69
70 plt.figure( figsize=(10, 7.5) )
71 plt.title('Observaciones + ajuste para $H_0$, '+str(Nx)+' puntos')
72 plt.xlabel('x')
73 plt.ylabel('y')
74 plt.grid()
75 plt.plot(x , y,'o-', label='observaciones')
76 plt.plot(x, f1(x, *popt1), 'r-', linewidth=8.0,
77          label='fit: $H_0$=%5.3f' % tuple(popt1))
78 plt.plot(x, f2(x, *popt2), 'y-', linewidth=4.0,
79          label='fit: $H_0$=%5.3f, acel=%5.3f' % tuple(popt2))
80 plt.plot(x, f3(x, *popt3), 'c-',  linewidth=2.0,
81          label='fit: $H_0$=%5.3f, acel=%5.3f, c=%5.3f' % tuple(popt3))
82 plt.plot(x, f4(x, *popt4), 'k-',  linewidth=1.0,
83          label='fit: $H_0$=%5.3f, acel=%5.3f, c=%5.3f, d=%5.3f'
84          % tuple(popt4))
85 plt.legend(loc="best")
86 plt.savefig('graficos/p4-observaciones+ajuste-H0-'+str(Nx)+'.png', dpi=100)
87 plt.show()
88
89
90 print('--------------------')
91
92 #quit()
```

- Desde la terminal ejecute:

  python3 p4.py

  e interprete el resultado.

  Alternativamente ejecute:

  python3

  y vaya agregando uno a uno los bloques del programa.

- Estudie cada paso del programa y agrege comentarios explicativos.

- Altere el programa para probar distintas cosas.

Se deberían generar los siguientes gráficos:

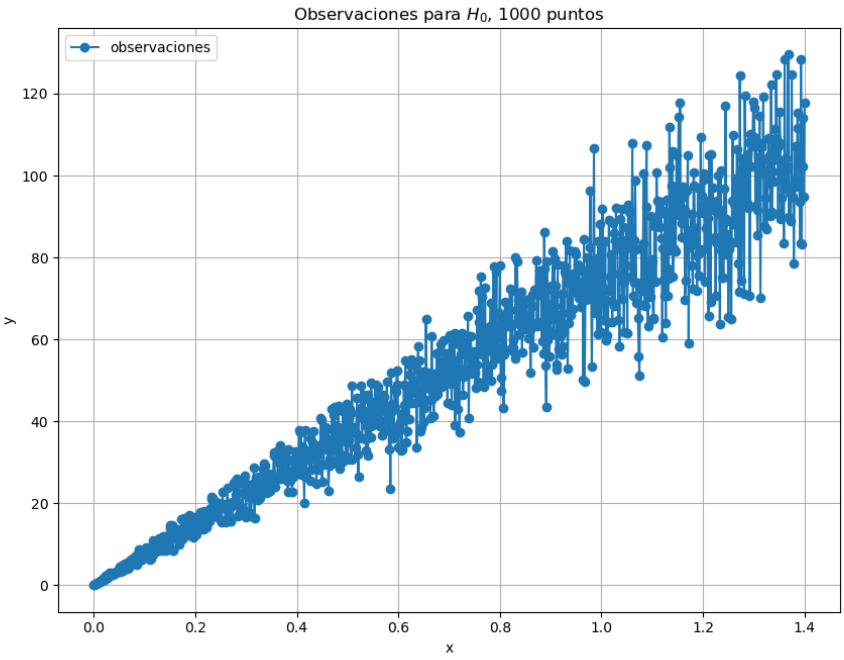

Figura 1: Observaciones para determinar H_0.

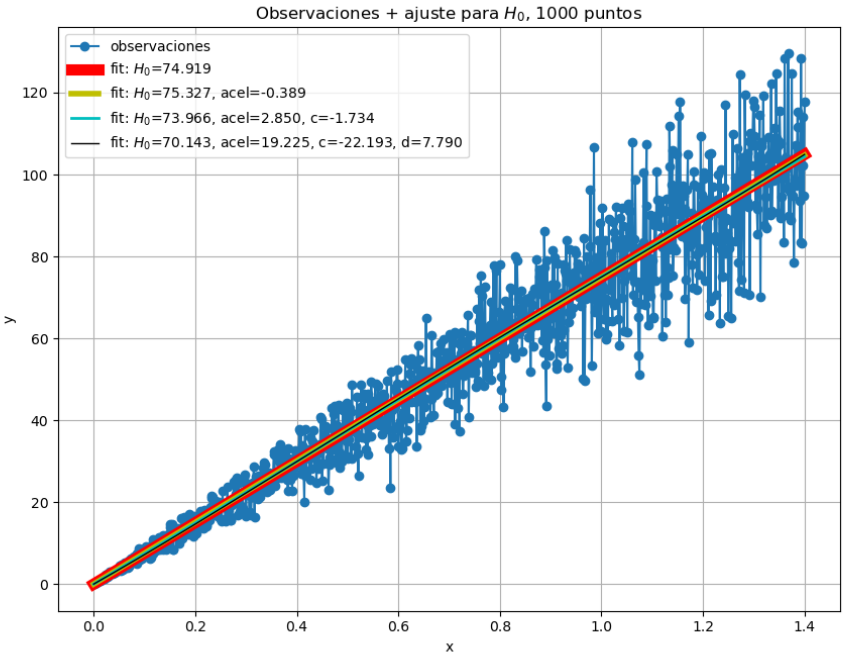

Figura 2: Observaciones y ajustes para determinar H_0.