Chi-square

Alejandro Salazar Lobos

January 2022

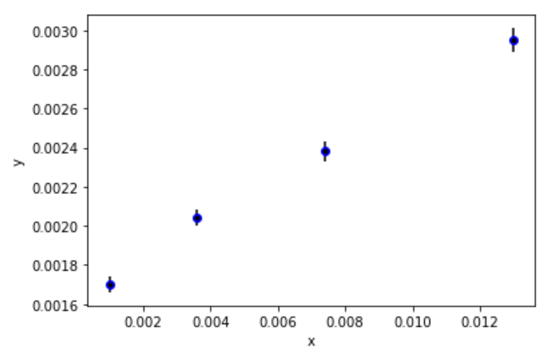
1 Introduction

The chi-square is a tool that helps you asses whether the mathematical model you are fitting to your data does indeed represent/describe the behaviour of this data.

2 Fitting a (weighted) curve to a set of data points

A set of observed data points:

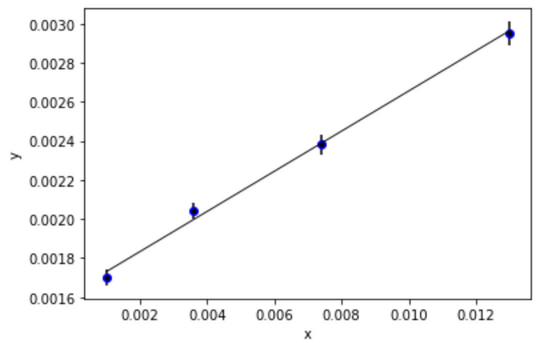
x_{obs}	y_{obs}	x_{unc}	y_{unc}
0.01300	0.00295	0.00002	0.00006
0.00740	0.00238	0.00002	0.00005
0.00360	0.00204	0.00001	0.00004
0.00102	0.00170	0.00006	0.00004



For this data you create a weighted fit (take into account the y-uncertainties). Make sure that your linearised equation leaves the bigger uncertainties in the y axis; this is because the weighted fit is based on the assumption that the uncertainties in the x variable are "negligible". Start with the assumption that this set of data points follows a straight line (according to your model, physical equation [this is whatever you manipulate from your lab manual to make it look like the equation of a line, for example]).

$$y = mx + b \tag{1}$$

In Python:
slope,intercept = np.polyfit(x_{obs}, y_{obs}, $w = 1/y_{unc}$, ...)



3 Testing the fit (weighted curve) with the chi-square

From the results obtained from np.polyfit, we can generate a set of expected y values in the following way:

$$y_{exp} = slope \times x_{obs} + intercept \tag{2}$$

We fill in a new column in the table above:

x_{obs}	y_{obs}	x_{unc}	y_{unc}	y_{exp}
0.01300	0.00295	0.00002	0.00006	0.00296
0.00740	0.00238	0.00002	0.00005	0.00239
0.00360	0.00204	0.00001	0.00004	0.00200
0.00102	0.00170	0.00006	0.00004	0.00173

The chi-square for a curve (Hughes&Hase, Eq. 6.1):

$$\chi^2 = \sum_i \left(\frac{y_{obs\,i} - y_{exp\,i}}{y_{unc\,i}} \right)^2 \tag{3}$$

... and for a probability distribution (also in Highes&Hase):

$$\chi^2 = \sum_i \frac{(O_i - E_i)^2}{E_i} \tag{4}$$

From which one calculates the *reduced* chi-square:

$$\chi^2_{reduced} = \frac{\chi^2}{\nu} \approx 1, \tag{5}$$

where $\nu = n - p$. n is the number of data points you observed (in this case $n = 4$) and p is the number of free parameters in your model (in this case, a line, two: the slope and the intercept, $p = 2$). By approximately 1, values of 0.5 and 1.5, for example are acceptable (but this depends on your value for ν; read more on Hughes&Hase, p. 107). Also, your value for $\chi^2_{reduced}$ can tell you whether you are overestimating (a really small reduced chi-square value) or underestimating your uncertainties (a really big reduced chi-square value) before you dive more into the analysis of your data.

For this example: $\chi^2 = 0.93$.

The next step is to test the chi-square cumulative distribution (Hughues&Hase, Eq. 8.3):

$$X(\chi^{2\prime};\nu) = \int_{\chi^{2\prime}=\chi^2}^{\chi^{2\prime}\to\infty} \frac{\chi^{2\prime\frac{\nu}{2}-1} exp(-\chi^{2\prime}/2)}{2^{\nu/2}\Gamma(\nu/2)} d\chi^{2\prime} \approx 0.5 \tag{6}$$

By approximately 0.5, a value of 0.3, for example, is acceptable. A value > 0.5 is questionable. Read more on Hughes&Hase, p. 106.

For this example, $X(\chi^{2\prime}, 2) = 0.39$.

4 Sample Python code

```python
# -*- coding: utf-8 -*-
"""
Created on Wed Jan 26 11:46:50 2022

@author: Alejandro Salazar Lobos
"""

import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import quad
import math

# Observed values and their uncertainties.
x = [0.01300, 0.00740, 0.00360, 0.00102]
y = [0.00295, 0.00238, 0.00204, 0.00170]
xunc = [0.00002, 0.00002, 0.00001, 0.00006]
yunc = [0.00006, 0.00005, 0.00004, 0.00004]

plt.plot(x, y, 'o', color='b')
plt.errorbar(x, y, yunc, xunc,'.', color='k')
plt.xlabel('x'); plt.ylabel('y')

# Weigthed fit.
slope,intercept = np.polyfit(x, y, deg=1, w=np.divide(1,yunc))

# Expected values (based on weighted fit).
x_exp = x
y_exp = np.multiply(slope,x) + intercept

#print(y_exp)

# Create a line based on weighted fit.
x_fit = np.linspace(min(x), max(x), 100)
y_fit = np.multiply(slope,x_fit) + intercept

plt.plot(x_fit, y_fit, color='k', linestyle='-', linewidth=0.9)

# Test the fit with the chi-square.
#--- Calculate the chi-square (Hughes&Hase, Eq. 6.1)
chi2_i = []
for i in range(0, len(y)):
    chi2_dummy = ((y[i]-y_exp[i])/yunc[i])**2
    chi2_i.append(chi2_dummy)
chi2 = sum(chi2_i)

#--- Calculate the reduced chi-square.
n = len(y) # number of data points.
p = 2 # number of free parameters (in this case the slope and the intercept).
v = n-p
chi2_reduced = chi2/(v)
print('Reduced chi squared =',chi2_reduced)

#--- Calculate the chi-square distribution.
def chi2dist(chi2, v): # Hughes&Hase (Eq. 8.3).
    return (chi2**(v/2 - 1))*np.exp(-chi2/2)/(2**(v/2) * math.gamma(v/2))

chi2prob = quad(chi2dist, chi2, np.infty, args=v)
```

```
print('Chi-square cumulative probability =', chi2prob[0])
```

5 More sources

From Caltech:
https://ned.ipac.caltech.edu/level5/Leo/Stats7_2.html. Retrieved: Jan. 26, 2022 at 1:00 PM (MST).

Hughes&Hase:
Hughes, I. G. & Hase, T. P. A. (2010). Measurements and Their Uncertainties: A Practical Guide to Modern Error Analysis. 1st. ed. Oxford University Press. Available here: https://login.ezproxy.library.ualberta.ca/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=e000xna&AN=330634&site=ehost-live&scope=site