Example Report on PH2255 Course: Introduction to Statistical Methods

Lev Levitin, RHUL Physics

15 January 2021

Abstract

This is an example report on Statistical Methods prepared in LaTeX and Python. It describes the straight line fit example and illustrates the use of `fill_between` Matplotlib function to present the error in a fit function.

The Statistical Methods course [1] starts with an exercise on fitting a dataset (x_i, y_i, σ_i), Table. 1, with polynomials using the least-square method. This report covers the introductory step of fitting a straight line

$$f(x) = \theta_0 + \theta_1 x \tag{1}$$

through the data. This function is defined in Python with the following code

```
def func(x, *theta):
    theta0, theta1 = theta
    return theta0 + theta1*x
```

Here *theta allows the parameter vector $\boldsymbol{\theta} = (\theta_0, \theta_1)$ to have arbitrary length, useful for coding higher order polynomials.

The essence of the least square method is to find the value $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}$ that minimises

$$\chi^2(\hat{\boldsymbol{\theta}}) = \min \chi^2(\boldsymbol{\theta}) = \sum_i \frac{(y_i - f(x_i; \boldsymbol{\theta}))^2}{\sigma_i^2}. \tag{2}$$

Table 1: Dataset provided by the course script [1]. The values of x are assumed to be known precisely, the standard deviation σ of y is supplied.

x	y	σ
1.0	2.7	0.3
2.0	3.9	0.5
3.0	5.5	0.7
4.0	5.8	0.6
5.0	6.5	0.4
6.0	6.3	0.3
7.0	7.7	0.7
8.0	8.5	0.8
9.0	8.7	0.5

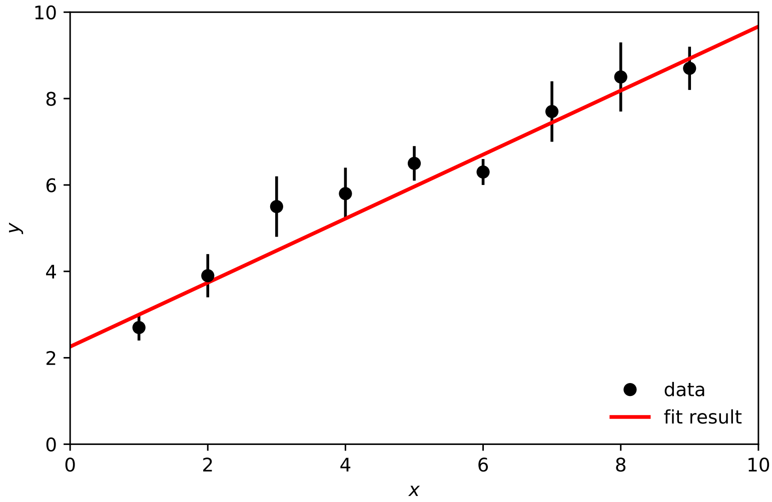

Figure 1: Straight line fit through data from Table. 1.

We perform this minimisation using curve_fit function provided by `scipy.optimise` Python module:

```
p0 = np.array([1.0, 1.0])
thetaHat, cov = curve_fit(func, x, y, p0, sig, absolute_sigma=True)
```

In addition to the fit function func and the dataset x,y,sig the vector of initial values of the fit parameters p0 is supplied to curve_fit. The function returns the estimator vector $\hat{\boldsymbol{\theta}}$ and the covariance matrix $\text{cov}[\hat{\theta}_i, \hat{\theta}_j]$ that encodes the uncertainties of $\hat{\theta}_{0,1}$ and correlations between them, that are important for the error propagation in later exercises. The fit is illustrated in Fig. 1. The estimator vector is

$$\theta_0 = 2.3 \pm 0.3, \quad \theta_1 = 0.74 \pm 0.06.$$

Finally Fig. 2 illustrates the use of Matplotlib function fill_between to plot a shaded band, representing the error in the fitted function $f(x) \pm \sigma_f(x)$. In this example an *arbitrary* function is chosen

$$\sigma_f(x) = 1 + \frac{1}{2}\cos x \tag{3}$$

purely to illustrate the Python functionality. We recognise that Eq. (3) is unrelated to the statistical methods. The relevant Python code is

```
fitSigma = 1 + 0.5*np.cos(xPlot)
ax.fill_between(xPlot, fit-fitSigma, fit+fitSigma, color='lightblue
    ', zorder=-1000)
```

Python code was included in this LaTeX document using lstlisting environment and \listinline command provided by `listings` LaTeX package [2].

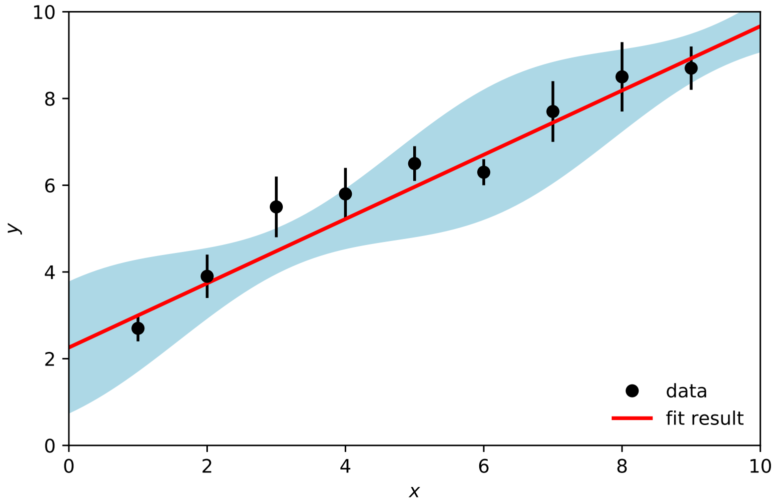

Figure 2: Example of Matplotlib `fill_between` function with *arbitrary* choice of the error band.

References

[1] G. Cowan, *Introduction to Statistic Methods*, RHUL PH2255 Moodle Page.

[2] `listings` LaTeXpackage: `http://texdoc.net/texmf-dist/doc/latex/listings/listings.pdf`

A Python Code

The complete code used in this example is presented below:

```
# #%matplotlib inline
# this line is required for the plots to appear in the Jupyter
    cells, rather than launching the matplotlib GUI
# %matplotlib widget
#this allows interactive view but you need to be in classic rather
    than CoCalc Jupyter notebook for this to work
from __future__ import division,print_function
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import curve_fit

# Consider the following set of $(x, y, \sigma)$ data points:
x   = np.array([1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0])
y   = np.array([2.7, 3.9, 5.5, 5.8, 6.5, 6.3, 7.7, 8.5, 8.7])
sig = np.array([0.3, 0.5, 0.7, 0.6, 0.4, 0.3, 0.7, 0.8, 0.5])

# define fit function
def func(x, *theta):
    theta0, theta1 = theta
    return theta0 + theta1*x

```

```
# set default parameter values and do the fit
p0 = np.array([1.0, 1.0])
thetaHat, cov = curve_fit(func, x, y, p0, sig, absolute_sigma=True)

# Having obtained $\hat{\underline\theta}$, we can plot the data and the fitted straight line
fig = plt.figure()
ax = fig.add_subplot(1,1,1)
ax.errorbar(x, y, yerr=sig, color='black', fmt='o')
# add an empty dataset to the axes to provide legend
ax.plot([], [], 'o', color='black', label='data')
ax.set_xlabel(r'$x$')
ax.set_ylabel(r'$y$')

# manually choose the x and y ranges for the plot
xMin = 0
xMax = 10
yMin = 0
yMax = 10
ax.set_xlim(xMin, xMax)
ax.set_ylim(yMin, yMax)

# generate the array of x for plotting a smooth fitted curve
xPlot = np.linspace(xMin, xMax, 100)
# calculate the fitted function for the above x
fit = func(xPlot, *thetaHat)
ax.plot(xPlot, fit, color='red', linewidth=2, label='fit result')

ax.legend(loc='lower right', frameon=False)

# Make and store plot
plt.tight_layout()
plt.show()
fig.savefig("simpleFit.pdf", format='pdf')

# The following code formats the '(x,y,sig)' dataset for $\LaTeX$ 'tabular' environment:
for i in range(len(x)):
    print('%.1f & %.1f & %.1f \\\\' % (x[i], y[i], sig[i]))

# The following code demonstrates the use of `fill_between` Matplotlib function.
# Large nergative 'zorder' ensures produced band is behind the dataset and fit line.
# `fitSigma` array represents the error in the fitted function.
# In this example it is **arbitrarily** assigned with a function of `xPlot`.
fitSigma = 1 + 0.5*np.cos(xPlot)
ax.fill_between(xPlot, fit-fitSigma, fit+fitSigma, color='lightblue', zorder=-1000)
plt.tight_layout()
plt.show()
fig.savefig("simpleFit_wrong_band.pdf", format='pdf')
```

python_code.py

In order to produce the above listing the Jupyter notebook was saved as `python_code.py` Python script, manually stripped of auto-generated comments, and then imported into LaTeX using the `lstinputlisting` command provided by the `listings` package [2].