Parameter Estimation
Hands-on Session

INFN School of Statistics
Paestum, 15-20 May 2022

https://agenda.infn.it/event/28039/

Glen Cowan
Physics Department
Royal Holloway, University of London
g.cowan@rhul.ac.uk
www.pp.rhul.ac.uk/~cowan

Introduction and materials

The exercises for parameter estimation are at (linked also to indico)

https://www.pp.rhul.ac.uk/~cowan/stat/paestum/exercises

The exercise and are described in the file fitting_exercises.pdf.

There are both python and ROOT/C++ versions.

For python, you need python 3 and install iminuit from https://pypi.org/project/iminuit/ with pip install iminuit

For ROOT you should have version 6 and C++ installed with a "cern-like" (e.g., lxplus) setup.

Comment on the $\ln L = \ln L_{\max} - ½$ contour

In the lectures, we saw that the standard deviations of fitted parameters are found from the tanget lines (planes) to the contour

$$\ln L = \ln L_{\max} - \frac{1}{2}$$

A similar procedure can be used to find a "confidence region" in the parameter space that will cover the true parameter with probability CL = 1 – α (the "confidence level). This uses the contour

$$\ln L = \ln L_{\max} - \frac{1}{2} F^{-1}_{\chi^2}(1-\alpha; N) \,, \qquad N = \text{number of parameters}$$

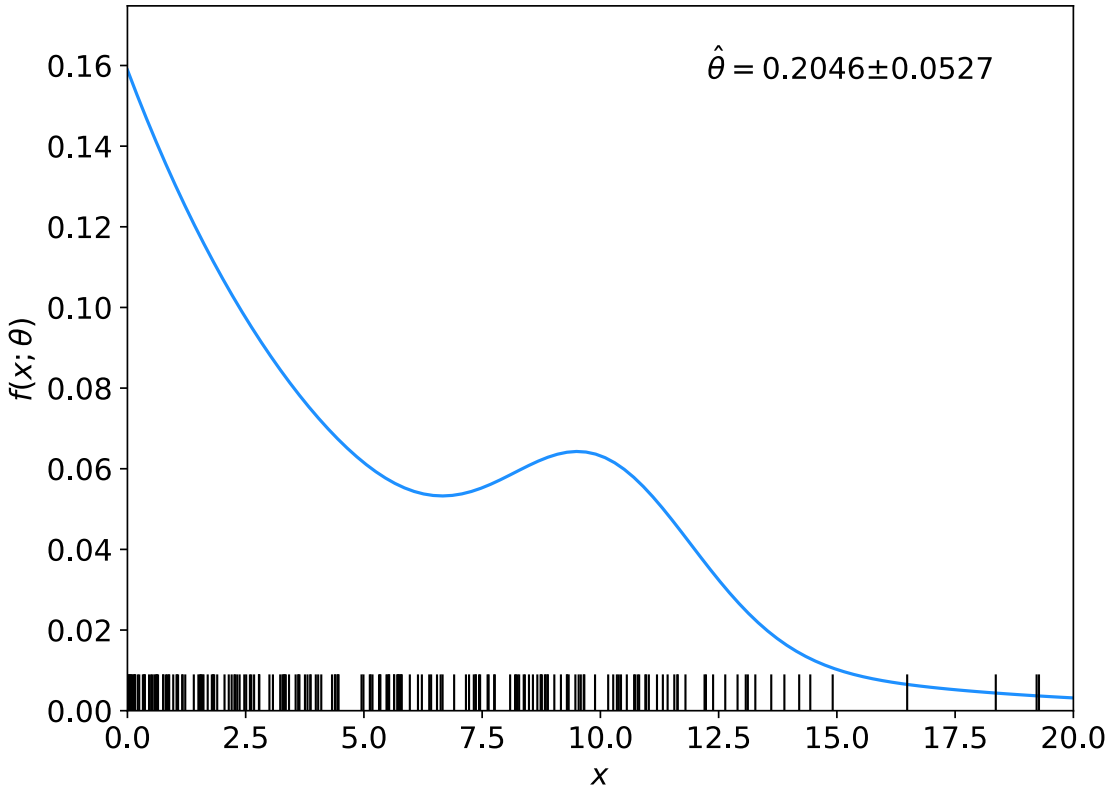
If you want the contour $\ln L = \ln L_{\max} - ½$ in iminuit, you need to choose CL (= 1 – α) such that $F\chi^{2-1}(1-\alpha, N) = 1$, i.e.,

$$\mathrm{CL} = F_{\chi^2}(1; N) = \texttt{stats.chi2.cdf(1.,N)}$$

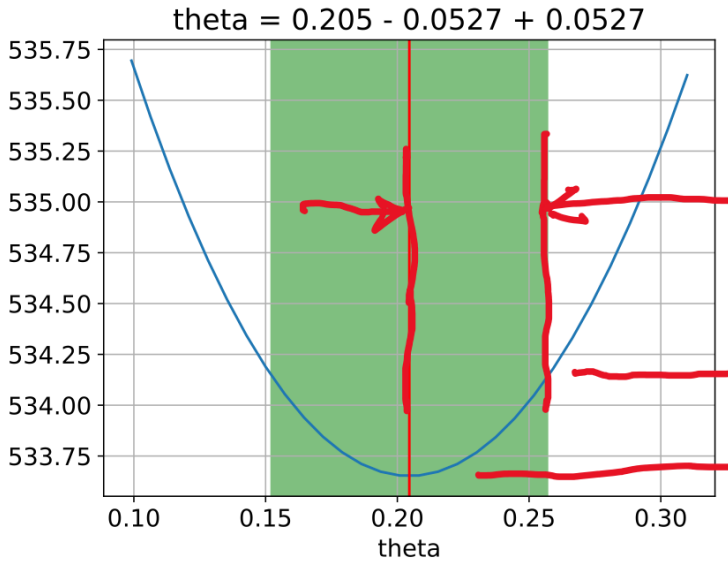
Solutions

1a) Running the program mlFit.py produces the following plots:

A fit of the pdf:



A scan of -lnL versus theta:



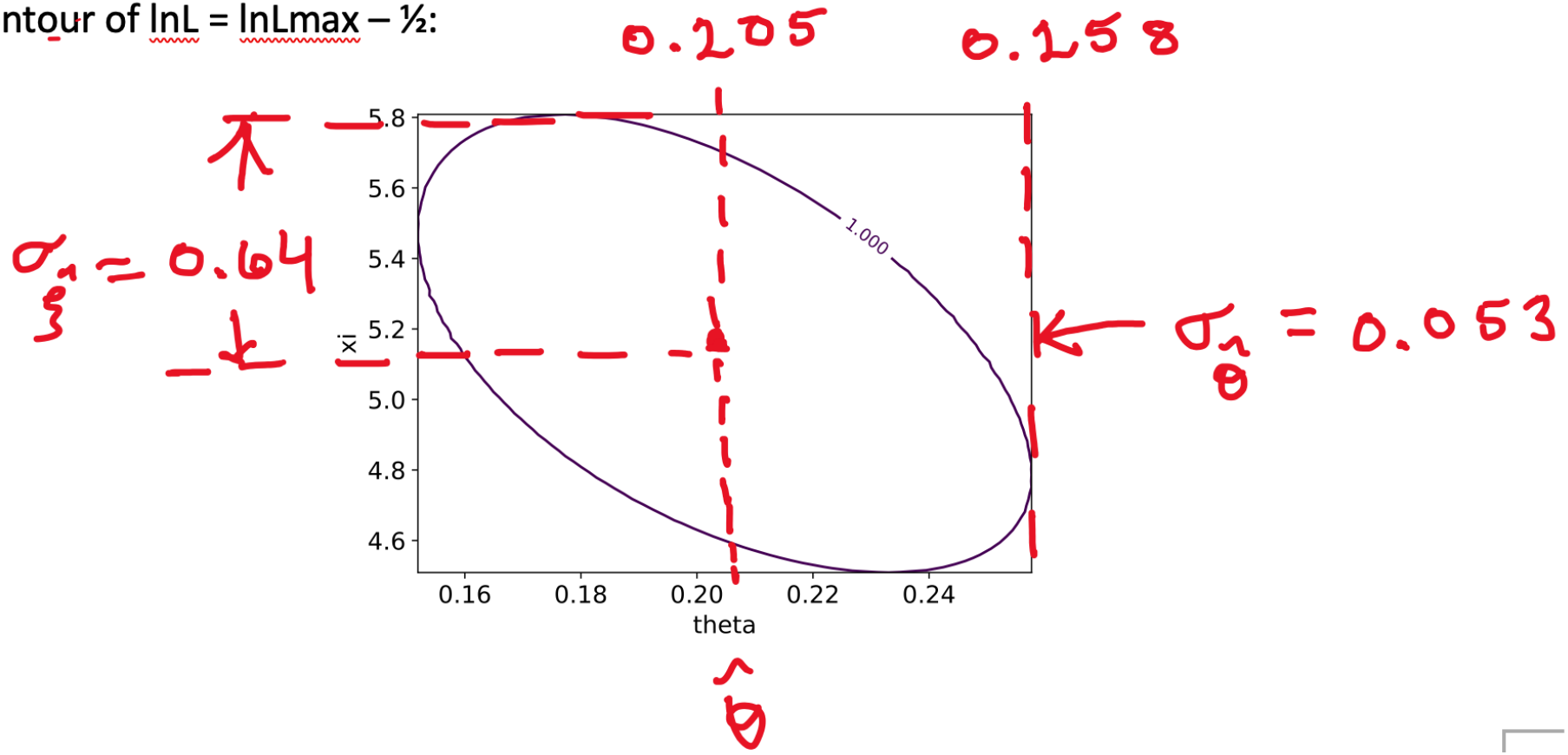
$-\ln L$

$\sigma_\theta = 0.053$

$\ln L_{max} - \frac{1}{2} = 534.15$

$\ln L_{max} = 533.65$

A contour of lnL = lnLmax – ½:



1b) Assume i.i.d. data sample, so $L(\boldsymbol{\theta}) = P(\mathbf{x}|\boldsymbol{\theta}) = \prod_{k=1}^{n} P(x_k|\boldsymbol{\theta})$

Assume inverse covariance from Fisher Information (large sample):

$$V_{ij}^{-1} = -E\left[\frac{\partial^2 \ln L}{\partial\theta_i \partial\theta_j}\right] = -\int \frac{\partial^2 \ln L}{\partial\theta_i \partial\theta_j} P(\mathbf{x}|\boldsymbol{\theta})\, d\mathbf{x}$$

Since $\ln L(\boldsymbol{\theta}) = \sum_{k=1}^{n} \ln P(x_k|\boldsymbol{\theta})$ we find

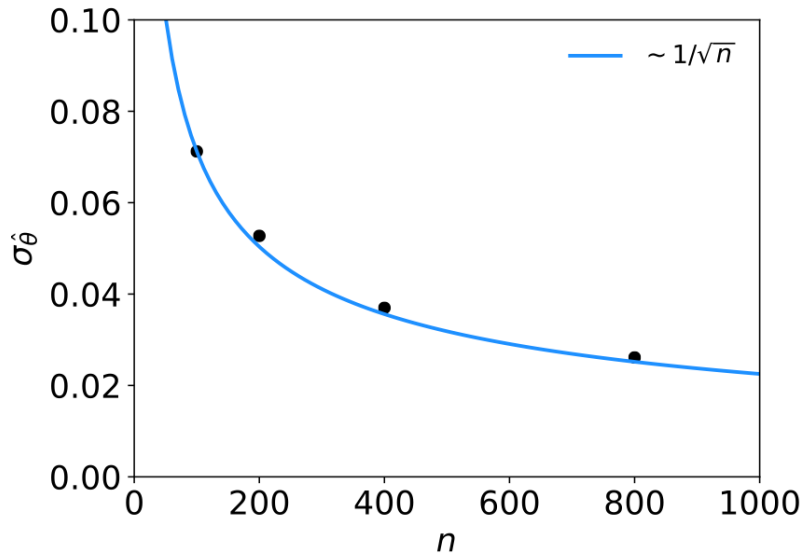
$$V_{ij}^{-1} = -\sum_{k=1}^{n} \int \frac{\partial^2 \ln P(x_k|\boldsymbol{\theta})}{\partial\theta_i \partial\theta_j} P(x_k|\boldsymbol{\theta})\, dx_k = -n \int \frac{\partial^2 \ln P(x|\boldsymbol{\theta})}{\partial\theta_i \partial\theta_j} P(x|\boldsymbol{\theta})\, dx$$

But $V^{-1}V = I$ so if $V^{-1} \propto n$, then $V \propto 1/n$, and so from the square roots of the diagonal elements $\sigma_{\hat{\theta}_i} \propto 1/\sqrt{n}$

1(c) Running mlFit.py with different numbers of events gave:

numVal	thetaHat	sigma_thetaHat
100	0.197218	0.071219
200	0.204551	0.052736
400	0.160808	0.036985
800	0.198224	0.026129

A plot of sigma_thetaHat versus numVal is shown below. The standard deviation of the estimator is seen to decrease as $1/\sqrt{n}$, as expected.



1(d) The results of the fit with different combinations of parameters adjustable are:

Free	Fixed	sigma_thetaHat
theta	mu, sigma, xi	0.044535
theta, xi	mu, sigma	0.052736
theta, xi, sigma	mu	0.064456
theta, xi, sigma, mu	--	0.085786

As can be seen, the standard deviation of the estimator of theta increases when it is fitted simultaneously with an increasing number of other adjustable parameters.