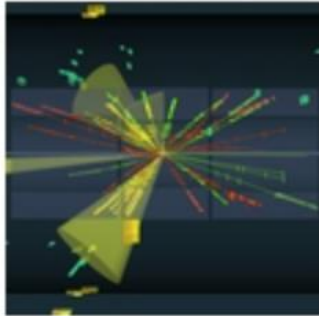


Parameter Estimation Hands-on Session



Taller de Altas Energías
Benasque, Spain (online)
16 September 2026

<http://benasque.org/2026tae/>

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

<https://www.pp.rhul.ac.uk/~cowan/stat/exercises/fitting>

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

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

`mlFit.py` ← use this for today's exercises

`histFit.py` ← binned version (uses python class)

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.

Prior to the exercises we will see how to obtain a confidence interval/region directly from contours of the log-likelihood.

Introduction to the exercises

Consider a pdf for continuous random variable x , (truncate and renormalize in $0 \leq x \leq x_{\max}$)

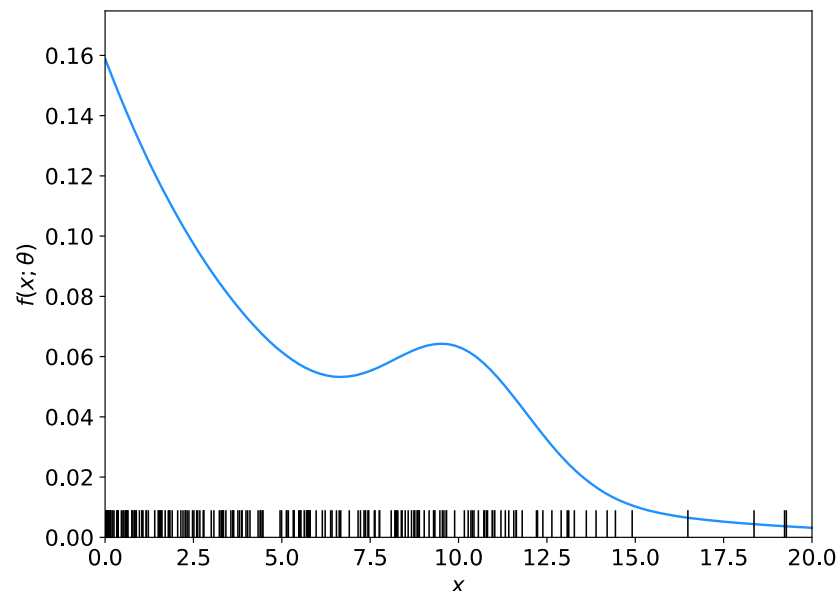
$$f(x; \theta, \xi) = \theta \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2} + (1 - \theta) \frac{1}{\xi} e^{-x/\xi}$$

θ = parameter of interest ,
gives signal rate.

Depending on context, take ξ, μ, σ
as nuisance parameters or fixed.

Generate i.i.d. sample $x_1, \dots, x_n$.

Estimate θ (and other params.)



Fitting with MINUIT (python or root/C++)

To use python, you will need to install the package iminuit (should just work with “pip install iminuit”). See:

<https://pypi.org/project/iminuit/>

Then download and run the program `mlFit.py` or the jupyter notebook `mlFit.ipynb` from

<http://www.pp.rhul.ac.uk/~cowan/stat/exercises/fitting/python>

To use C++/ROOT, download the files from

<http://www.pp.rhul.ac.uk/~cowan/stat/exercises/fitting/root>

to your work directory and build the executable program by typing `make` and run by typing `./mlFit`. This uses the class `TMinuit`, which is described here:

<https://root.cern.ch/doc/master/classTMinuit.html>

The instructions below refer to the python version; the corresponding steps for the C++/ROOT program are similar.

mlFit.py (also jupyter notebook mlFit.ipynb)

```
1  # Example of maximum-likelihood fit with iminuit version 2.
2  # pdf is a mixture of Gaussian (signal) and exponential (background),
3  # truncated in [xMin,xMax].
4  # G. Cowan / RHUL Physics / December 2021
5
6  import numpy as np
7  import scipy.stats as stats
8  from scipy.stats import truncexpon
9  from scipy.stats import truncnorm
10 from scipy.stats import chi2
11 import iminuit
12 from iminuit import Minuit
13 import matplotlib.pyplot as plt
14 from matplotlib import container
15 plt.rcParams["font.size"] = 14
16 print("iminuit version:", iminuit.__version__) # need 2.x
17
18 # define pdf and generate data
19 np.random.seed(seed=1234567) # fix random seed
20 theta = 0.2 # fraction of signal
21 mu = 10. # mean of Gaussian
22 sigma = 2. # std. dev. of Gaussian
23 xi = 5. # mean of exponential
24 xMin = 0.
25 xMax = 20.
```

Define the fit function

```
27 def f(x, par):
28     theta    = par[0]
29     mu       = par[1]
30     sigma    = par[2]
31     xi       = par[3]
32     fs = stats.truncnorm.pdf(x, a=(xMin-mu)/sigma, b=(xMax-mu)/sigma, loc=mu, scale=sigma)
33     fb = stats.truncexpon.pdf(x, b=(xMax-xMin)/xi, loc=xMin, scale=xi)
34     return theta*fs + (1-theta)*fb
```

Generate the data

```
36 numVal = 200
37 xData = np.empty([numVal])
38 for i in range (numVal):
39     r = np.random.uniform();
40     if r < theta:
41         xData[i] = stats.truncnorm.rvs(a=(xMin-mu)/sigma, b=(xMax-mu)/sigma, loc=mu,
42                                     scale=sigma)
43     else:
44         xData[i] = stats.truncexpon.rvs(b=(xMax-xMin)/xi, loc=xMin, scale=xi)
```

Set up the fit

```
45 # Function to be minimized is negative log-likelihood
46 def negLogL(par):
47     pdf = f(xData, par)
48     return -np.sum(np.log(pdf))
49
50 # Initialize Minuit and set up fit:
51 parin = np.array([theta, mu, sigma, xi]) # initial values (here = true values)
52 parname = ['theta', 'mu', 'sigma', 'xi']
53 parstep = np.array([0.1, 1., 1., 1.]) # initial setp sizes
54 parfix = [False, True, True, False] # change these to fix/free parameters
55 parlim = [(0., 1), (None, None), (0., None), (0., None)] # set limits
56 m = Minuit(negLogL, parin, name=parname)
57 m.errors = parstep
58 m.fixed = parfix
59 m.limits = parlim
60 m.errordef = 0.5 # errors from lnL = lnLmax - 0.5
```

Do the fit, get errors, extract results

```
62 # Do the fit, get errors, extract results
63 m.migrad()                                # minimize -logL
64 MLE = m.values                            # max-likelihood estimates
65 sigmaMLE = m.errors                      # standard deviations
66 cov = m.covariance                       # covariance matrix
67 rho = m.covariance.correlation()         # correlation coeffs.
68
69 print(r"par index, name, estimate, standard deviation:")
70 for i in range(m.npar):
71     if not m.fixed[i]:
72         print("{:4d}".format(i), "{:<10s}".format(m.parameters[i]), " = ",
73               "{:.6f}".format(MLE[i]), " +/- ", "{:.6f}".format(sigmaMLE[i]))
74
75 print()
76 print(r"free par indices, covariance, correlation coeff.:")
77 for i in range(m.npar):
78     if not(m.fixed[i]):
79         for j in range(m.npar):
80             if not(m.fixed[j]):
81                 print(i, j, "{:.6f}".format(cov[i,j]), "{:.6f}".format(rho[i,j]))
```

Make some plots...

Approximate confidence intervals/regions from the likelihood function

Suppose we test parameter value(s) $\theta = (\theta_1, \dots, \theta_n)$ using the ratio

$$\lambda(\theta) = \frac{L(\theta)}{L(\hat{\theta})} \quad 0 \leq \lambda(\theta) \leq 1$$

Lower $\lambda(\theta)$ means worse agreement between data and hypothesized θ . Equivalently, usually define

$$t_{\theta} = -2 \ln \lambda(\theta)$$

so higher t_{θ} means worse agreement between θ and the data.

p -value of θ therefore

$$p_{\theta} = \int_{t_{\theta, \text{obs}}}^{\infty} f(t_{\theta} | \theta) dt_{\theta}$$

need pdf

Confidence region from Wilks' theorem

Wilks' theorem says (in large-sample limit and provided certain conditions hold...)

$$f(t_{\theta}|\theta) \sim \chi_n^2$$

chi-square dist. with # d.o.f. =
of components in $\theta = (\theta_1, \dots, \theta_n)$.

Assuming this holds, the p -value is

$$p_{\theta} = 1 - F_{\chi_n^2}(t_{\theta}) \quad \leftarrow \text{set equal to } \alpha$$

To find boundary of confidence region set $p_{\theta} = \alpha$ and solve for t_{θ} :

$$t_{\theta} = F_{\chi_n^2}^{-1}(1 - \alpha)$$

Recall also

$$t_{\theta} = -2 \ln \frac{L(\theta)}{L(\hat{\theta})}$$

Confidence region from Wilks' theorem (cont.)

i.e., boundary of confidence region in θ space is where

$$\ln L(\theta) = \ln L(\hat{\theta}) - \frac{1}{2} F_{\chi_n^2}^{-1}(1 - \alpha)$$

For example, for $1 - \alpha = 68.3\%$ and $n = 1$ parameter,

$$F_{\chi_1^2}^{-1}(0.683) = 1$$

and so the 68.3% confidence level interval is determined by

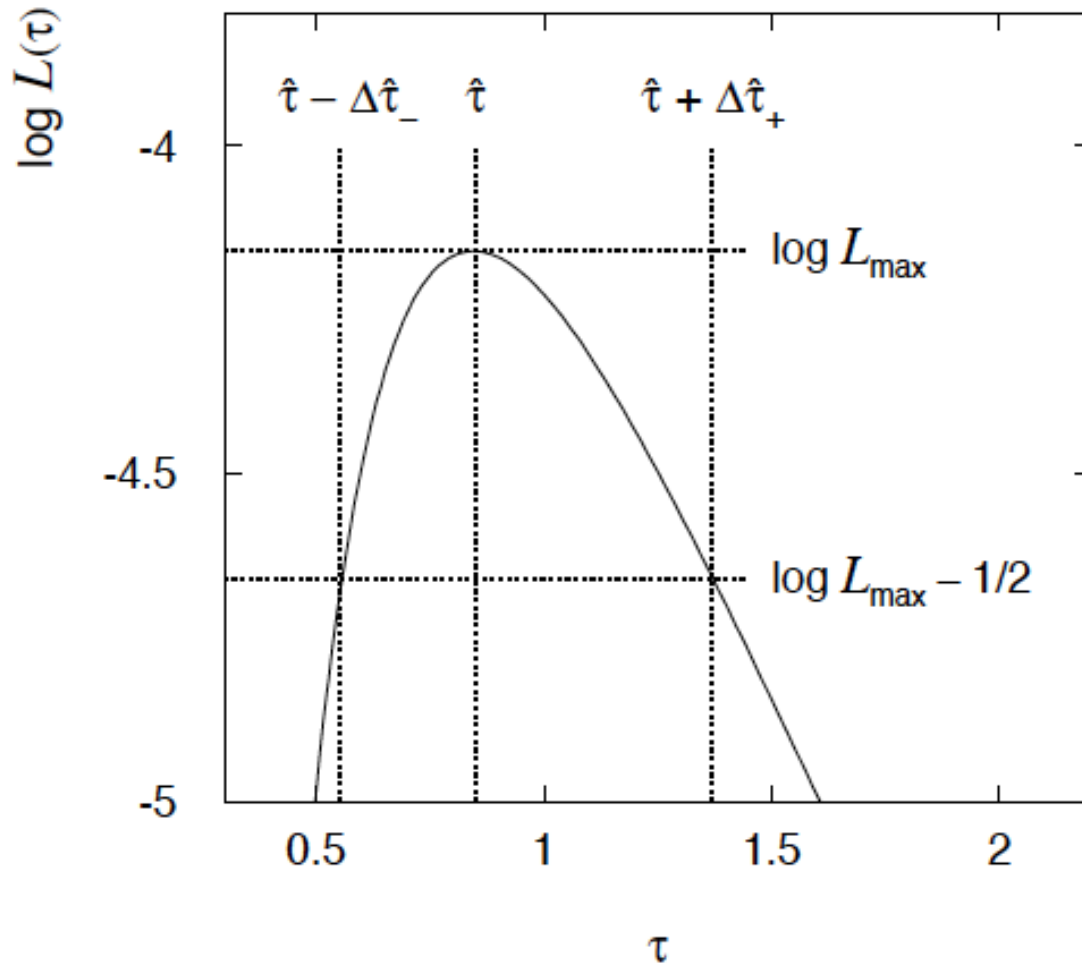
$$\ln L(\theta) = \ln L(\hat{\theta}) - \frac{1}{2}$$

Same as recipe for finding the estimator's standard deviation, i.e.,

$[\hat{\theta} - \sigma_{\hat{\theta}}, \hat{\theta} + \sigma_{\hat{\theta}}]$ is a 68.3% CL confidence interval.

Example of interval from $\ln L(\theta)$

For $n=1$ parameter, CL = 0.683, $Q_\alpha = 1$.



Our exponential example, now with only $n = 5$ events.

Can report ML estimate with approx. confidence interval from $\ln L_{\max} - 1/2$ as “asymmetric error bar”:

$$\hat{\tau} = 0.85^{+0.52}_{-0.30}$$

Multiparameter case

For increasing number of parameters, $CL = 1 - \alpha$ decreases for confidence region determined by a given

$$Q_\alpha = F_{\chi_n^2}^{-1}(1 - \alpha)$$

Q_α	$1 - \alpha$				
	$n = 1$	$n = 2$	$n = 3$	$n = 4$	$n = 5$
1.0	0.683	0.393	0.199	0.090	0.037
2.0	0.843	0.632	0.428	0.264	0.151
4.0	0.954	0.865	0.739	0.594	0.451
9.0	0.997	0.989	0.971	0.939	0.891

Multiparameter case (cont.)

Equivalently, Q_α increases with n for a given $\text{CL} = 1 - \alpha$.

$1 - \alpha$	$\tilde{Q}_\alpha$				
	$n = 1$	$n = 2$	$n = 3$	$n = 4$	$n = 5$
0.683	1.00	2.30	3.53	4.72	5.89
0.90	2.71	4.61	6.25	7.78	9.24
0.95	3.84	5.99	7.82	9.49	11.1
0.99	6.63	9.21	11.3	13.3	15.1

Comment on the $\ln L = \ln L_{\max} - \frac{1}{2}$ contour

In the lectures, we saw that the standard deviations of fitted parameters are found from the tangent 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 $CL = 1 - \alpha$ (the “confidence level”). This uses the contour

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

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

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

$\ln L$ in a class, binned data,...

Sometimes it is convenient to have the function being minimized as a method of a class. An example of this is shown in the program `histFit.py` (in same directory), which does the same fit as in `mlFit` but with a histogram of the data:

