Statistical Data Analysis
Discussion notes – week 9

- Problem sheet 6
- Some info on Problem Sheet 8 regarding iminuit
- Comment about *s*-sigma covariance ellipse
- Another fit example (Snell's Law)

Problem sheet 6

Exercise 1: A number n of observed events is modeled as a Poisson distributed variable with mean $s + b$, where s and b are the expected numbers of events from the signal and background processes, respectively. Suppose $b = 3.7$ and $n_{\text{obs}} = 15$ are observed.

1(a) [4 marks] Compute the p-value of the hypothesis that $s = 0$. To sum Poisson probabilities, you can use the relation

$$\sum_{n=0}^{m} P(n;\nu) \;=\; 1 - F_{\chi^2}(2\nu; n_{\text{dof}})\,,$$

where $P(n;\nu)$ is the Poisson probability for n given a mean value ν, and F_{χ^2} is the cumulative χ^2 distribution for $n_{\text{dof}} = 2(m+1)$ degrees of freedom. This can be computed in python using `scipy.stats.chi2.cdf` or with the ROOT routine `TMath::Prob`.

1(b) [1 mark] Find the corresponding equivalent Gaussian significance Z and evaluate numerically (see lecture notes).

Problem sheet 6 / 1(a)

a) $p_0 = P(n \geq n_{obs} \mid s = 0,\ b = 3.7)$

$$= \sum_{n=n_{obs}}^{\infty} \frac{b^n}{n!} e^{-b}$$

$$= 1 - \sum_{n=0}^{n_{obs}-1} \frac{b^n}{n!} e^{-b}$$

upper limit of sum

$$= F_{\chi^2}(2b;\ 2n_{obs}) \qquad \left[n_{d.f} = 2(m+1) = 2n_{obs} \right]$$

$$= \text{scipy.stats.chi2.cdf}(2*3.7,\ 2*15)$$

$$= 1 - \text{TMath::Prob}(2*3.7,\ 2*15)$$

$$= 8.2 \times 10^{-6}$$

Problem sheet 6 / 1(b)

b) $Z_0 = \Phi^{-1}(1 - p_0)$

$= \text{scipy.stats.norm.ppf}(1 - p_0)$

$= \text{TMath::NormQuantile}(1 - p_0)$

$= 4.3$

Problem sheet 6

Exercise 2: Suppose x follows a Gaussian distribution with mean μ and variance σ^2 and consider an independently and identically distributed sample of N values $x_1, \ldots, x_N$.

2(a) [1 mark] Write down the log-likelihood function for the mean μ and variance σ^2.

2) $x \sim \text{Gauss}(\mu, \sigma^2)$

i.i.d. sample $x_1, \ldots, x_N$

2a) $$L(\mu, \sigma^2) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x_i-\mu)^2/2\sigma^2}$$

$$\Rightarrow \ln L(\mu, \sigma^2) = -\frac{1}{2} \sum_{i=1}^{N} \frac{(x_i-\mu)^2}{\sigma^2} - \frac{N}{2} \ln\sigma^2 + C$$

Problem sheet 6

2(b) [4 marks] Show that the maximum-likelihood estimators for μ and σ^2 are

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i \,, \qquad \widehat{\sigma^2} = \frac{1}{N}\sum (x_i - \hat{\mu})^2 \,.$$

2b) $$\frac{\partial \ln L}{\partial \mu} = \sum_{i=1}^{N} \frac{x_i - \mu}{\sigma^2} \overset{\text{set}}{=} 0 \qquad (1)$$

$$\frac{\partial \ln L}{\partial \sigma^2} = \frac{1}{2}\sum_{i=1}^{N} \frac{(x_i - \mu)^2}{(\sigma^2)^2} - \frac{N}{2\sigma^2} \overset{\text{set}}{=} 0 \qquad (2)$$

From (1) $$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

From (2) $$\hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N} (x_i - \hat{\mu})^2$$

Problem sheet 6

2(c) [5 marks] Find the Fisher information matrix for μ and σ^2 (use $\theta_1 = \mu$, $\theta_2 = \sigma^2$).

2c) Fisher information matrix

$$I_{ij} = -E\left[\frac{\partial^2 \ln L}{\partial\theta_i\,\partial\theta_j}\right] \qquad \theta_1 = \mu, \quad \theta_2 = \sigma^2$$

$$\frac{\partial^2 \ln L}{\partial \mu^2} = -\frac{N}{\sigma^2}$$

$$\frac{\partial^2 \ln L}{\partial (\sigma^2)^2} = \frac{N}{2\sigma^4} - \sum_{i=1}^{N} \frac{(x_i - \mu)^2}{\sigma^6}$$

$$\frac{\partial^2 \ln L}{\partial \mu\,\partial \sigma^2} = -\frac{1}{\sigma^4} \sum_{i=1}^{N} (x_i - \mu)$$

Problem sheet 6

$$E\left[\frac{\partial^2 \ln L}{\partial \mu^2}\right] = -\frac{N}{\sigma^2}$$

$$E\left[\frac{\partial^2 \ln L}{\partial (\sigma^2)^2}\right] = \frac{N}{2\sigma^4} - \sum_{i=1}^{N} \frac{\overbrace{E[(x_i-\mu)^2]}^{\sigma^2}}{\sigma^6} = -\frac{N}{2\sigma^4}$$

$$E\left[\frac{\partial^2 \ln L}{\partial \mu\, \partial \sigma^2}\right] = -\frac{1}{\sigma^4} \sum_{i=1}^{N} \underbrace{E[x_i - \mu]}_{\mu - \mu = 0} = 0$$

$$\Rightarrow \quad I = \begin{pmatrix} \frac{N}{\sigma^2} & 0 \\ 0 & \frac{N}{2\sigma^4} \end{pmatrix}$$

Problem sheet 6

2(d) [3 marks] By using this matrix derive the following expressions for the variances of the estimators and their covariance:

$$V[\hat{\mu}] = \frac{\sigma^2}{N}, \qquad V[\widehat{\sigma^2}] = \frac{2\sigma^4}{N}, \qquad \mathrm{cov}[\hat{\mu}, \widehat{\sigma^2}] = 0.$$

2d) Use $V^{-1} = I$

assumes zero bias, info inequality → equality

Since I diagonal, inverse by inspection

$$V = \begin{pmatrix} \frac{\sigma^2}{N} & 0 \\ 0 & \frac{2\sigma^4}{N} \end{pmatrix} = \begin{pmatrix} V[\hat{\mu}] & \mathrm{cov}[\hat{\mu}, \widehat{\sigma^2}] \\ \mathrm{cov}[\hat{\mu}, \widehat{\sigma^2}] & V[\widehat{\sigma^2}] \end{pmatrix}$$

Problem sheet 6 / 2(d) comment

Suppose we don't want the variance of the estimator of the variance but rather the standard deviation of the estimator of the standard deviation ("error of the error").

Std. dev. is square root of the variance: $\sigma = (\sigma^2)^{1/2}$

MLE of a a function of a parameter: $\hat{\sigma} = \left(\widehat{\sigma^2}\right)^{1/2}$

We found $V[\widehat{\sigma^2}] = \dfrac{2\sigma^4}{N}$

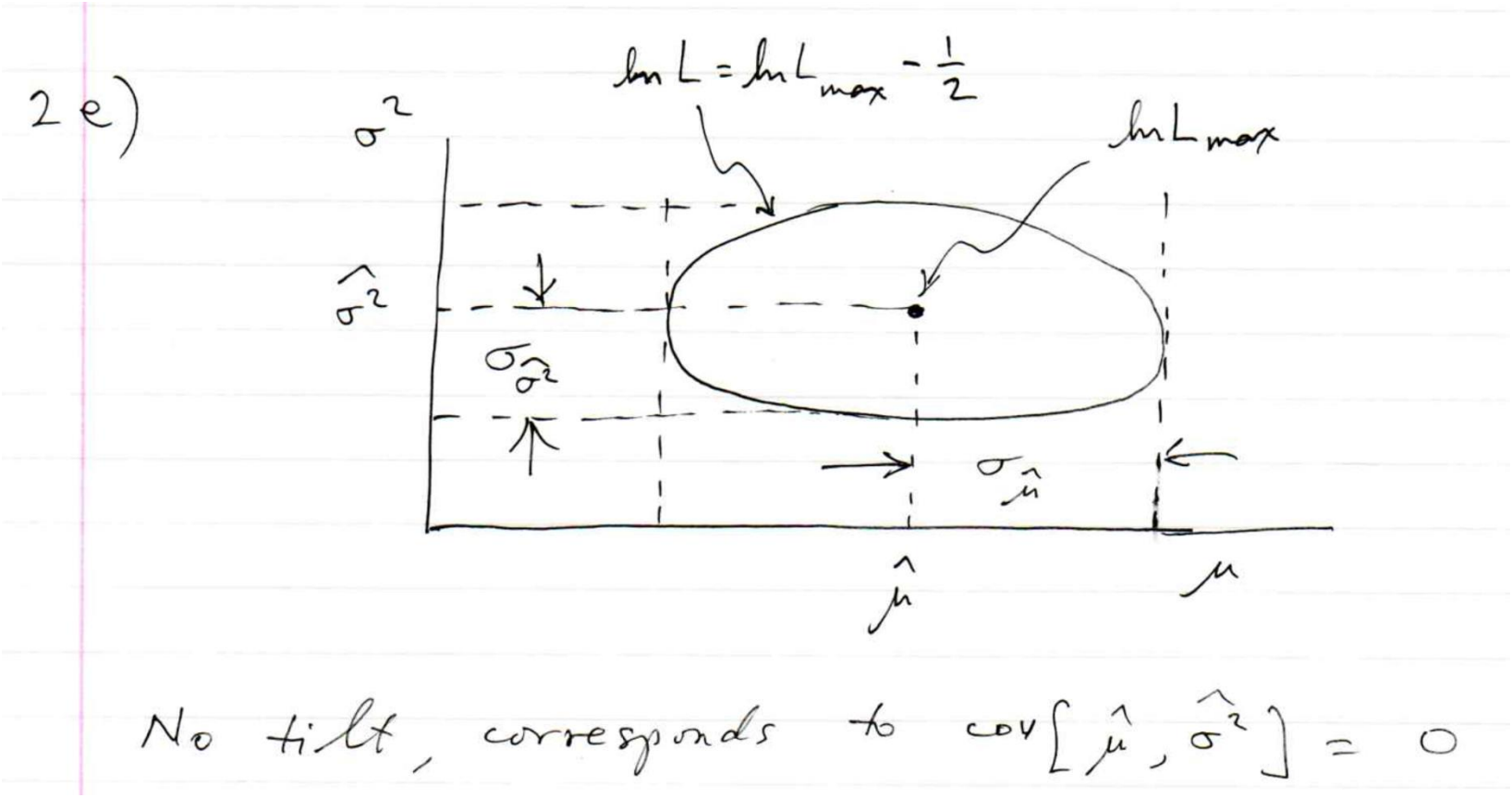
Use error propagation:

$$V[\hat{\sigma}] = \left(\frac{\partial\hat{\sigma}}{\partial\widehat{\sigma^2}}\right)^2\Bigg|_{\widehat{\sigma^2}=\sigma^2} V[\widehat{\sigma^2}] = \left(\frac{1}{2}(\sigma^2)^{-1/2}\right)^2 \frac{2\sigma^4}{N} = \frac{\sigma^2}{2N}$$

$$\longrightarrow \quad \sigma_{\hat{\sigma}} = \frac{\sigma}{\sqrt{2N}} \qquad \text{(compare to } \sigma_{\hat{\mu}} = \frac{\sigma}{\sqrt{N}} \text{ .)}$$

Problem sheet 6

2(e) [2 marks] Show with a sketch how one can estimate the standard deviations of $\hat{\mu}$ and $\widehat{\sigma^2}$ using the log-likelihood function. State how the sketch reflects in this case the appropriate value of the covariance of the two estimators.



Some info on Problem Sheet 8 (iminuit)

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

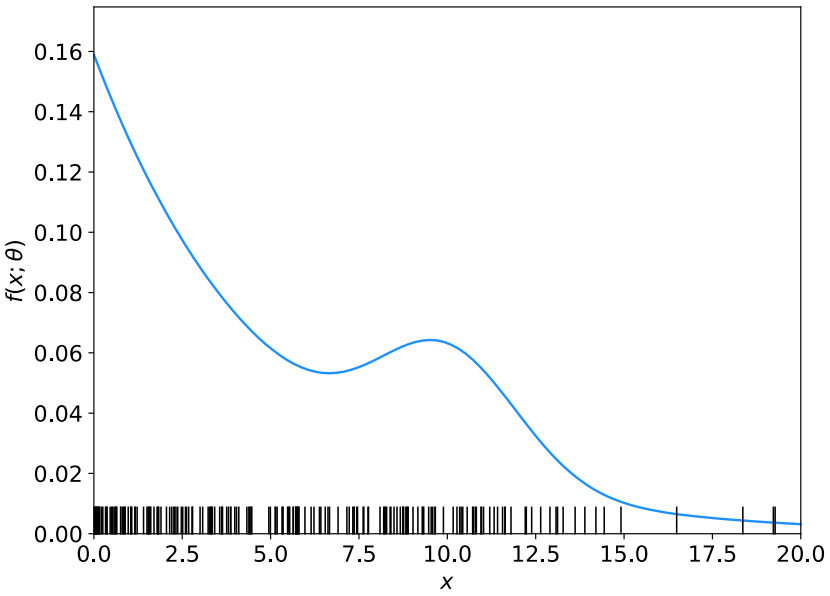
$$f(x;\theta,\xi) = \theta \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2} + (1-\theta)\frac{1}{\xi} e^{-x/\xi}$$

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,..., x_n.

Estimate θ (and other params.)



A quick look at mlFit.py

```
# Example of maximum-likelihood fit with iminuit version 2.
# pdf is a mixture of Gaussian (signal) and exponential (background),
# truncated in [xMin,xMax].
# G. Cowan / RHUL Physics / December 2022


import numpy as np
import scipy.stats as stats
from scipy.stats import truncexpon
from scipy.stats import truncnorm
from scipy.stats import chi2
import iminuit
from iminuit import Minuit
import matplotlib.pyplot as plt
from matplotlib import container
plt.rcParams["font.size"] = 14
print("iminuit version:", iminuit.__version__)  # need 2.x


# define pdf and generate data
np.random.seed(seed=1234567)       # fix random seed
theta = 0.2                        # fraction of signal
mu = 10.                           # mean of Gaussian
sigma = 2.                         # std. dev. of Gaussian
xi = 5.                          # mean of exponential
xMin = 0.
xMax = 20.
```

Define the fit function

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

Generate the data

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

Set up the fit

```
# Function to be minimized is negative log-likelihood
def negLogL(par):
    pdf = f(xData, par)
    return -np.sum(np.log(pdf))


# Initialize Minuit and set up fit:
parin   = np.array([theta, mu, sigma, xi]) # initial values (here = true values)
parname = ['theta', 'mu', 'sigma', 'xi']
parname_latex = [r'$\theta$', r'$\mu$', r'$\sigma$', r'$\xi$']
parstep = np.array([0.1, 1., 1., 1.])      # initial setp sizes
parfix  = [False, True, True, False]       # change these to fix/free params
parlim  = [(0.,1), (None, None), (0., None), (0., None)]    # set limits
m = Minuit(negLogL, parin, name=parname)
m.errors = parstep
m.fixed = parfix
m.limits = parlim
m.errordef = 0.5                    # errors from lnL = lnLmax - 0.5
```

Do the fit, get errors, extract results

```
# Do the fit, get errors, extract results
m.migrad()                          # minimize -logL
MLE = m.values                      # max-likelihood estimates
sigmaMLE = m.errors                 # standard deviations
cov = m.covariance                  # covariance matrix
rho = m.covariance.correlation()    # correlation coeffs.

print(r"par index, name, estimate, standard deviation:")
for i in range(m.npar):
    if not m.fixed[i]:
        print("{:4d}".format(i), "{:<10s}".format(m.parameters[i]), " = ",
         "{:.6f}".format(MLE[i]), " +/- ", "{:.6f}".format(sigmaMLE[i]))

print()
print(r"free par indices, covariance, correlation coeff.:")
for i in range(m.npar):
    if not(m.fixed[i]):
        for j in range(m.npar):
            if not(m.fixed[j]):
                print(i, j, "{:.6f}".format(cov[i,j]),
            "{:.6f}".format(rho[i,j]))
```

Make some plots...

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 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 – α (the "confidence level"). This uses the contour

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

N = 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.,

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

Comment on "*s*-sigma" covariance ellipse

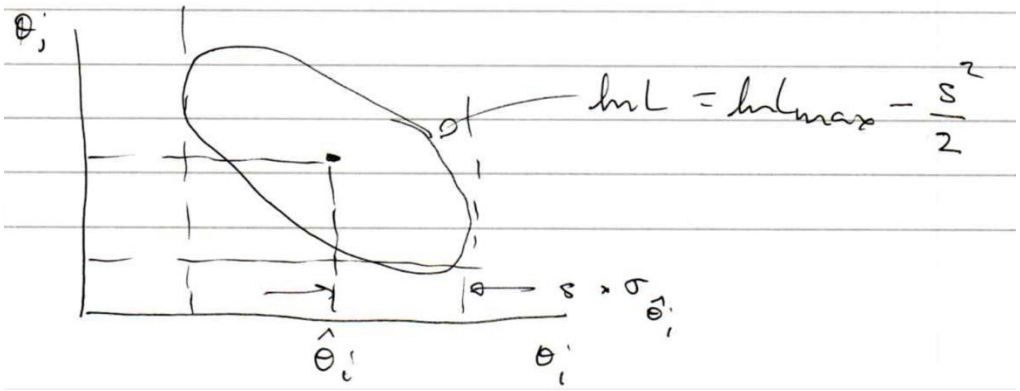
We have used the formula for the confidence region:

$$\ln L(\vec{\theta}) = \ln L_{\max} - \frac{1}{2} \underbrace{F^{-1}_{\chi^2_n}(1-\alpha)}_{Q_\alpha}$$

Similar to the ellipse whose tangents give std. deviations:

$$\ln L(\vec{\theta}) = \ln L_{\max} - \frac{1}{2} \underbrace{(\vec{\theta} - \hat{\vec{\theta}})^T V^{-1} (\vec{\theta} - \hat{\vec{\theta}})}_{s^2}$$

tangents at *s* std. devs.



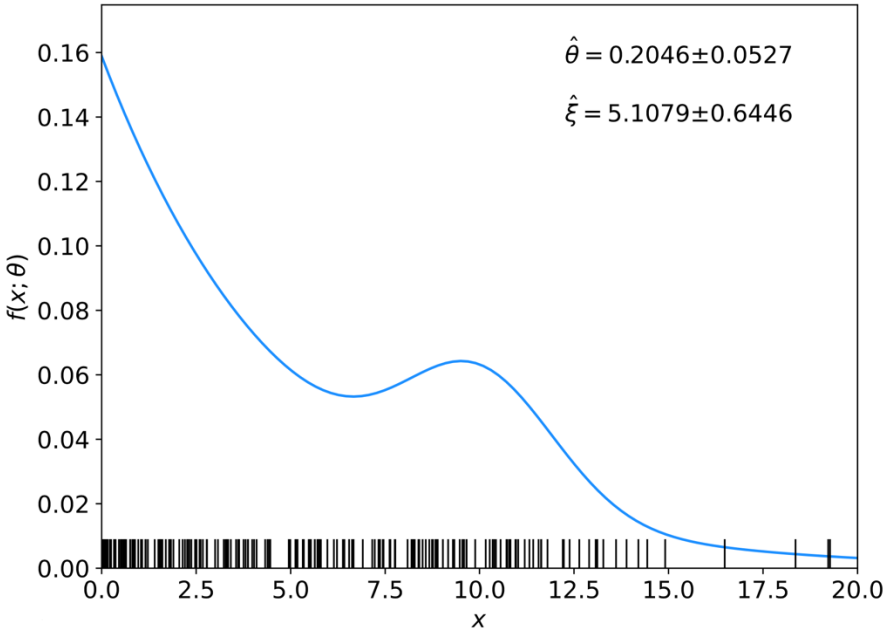
So *s*-sigma ellipse corresponds to

$$Q_\alpha = F^{-1}_{\chi^2_N}(1-\alpha) = s^2$$

$$\mathrm{CL} = 1 - \alpha = F_{\chi^2_N}(s^2)$$

Comments on using iminuit

In our iminuit example mlFit.py, the only argument of the log-likelihood function is the parameter array, and the data array xData enters as global (usually not a good idea):



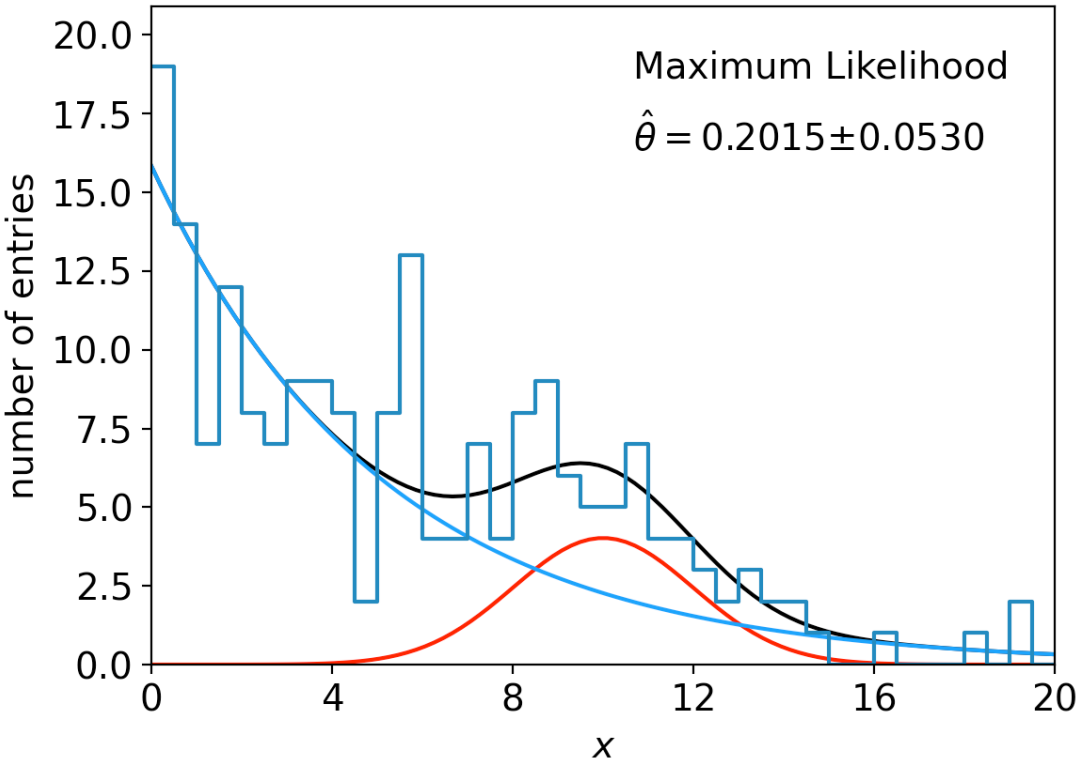

```
def negLogL(par):
    pdf = f(xData, par)
    return -np.sum(np.log(pdf))

    ⋮

m = Minuit(negLogL, par, name=parname)
```

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, which does the same fit as in mlFit.py but with a histogram of the data:



Commentary on histFit.py

The global data can be avoided if we make the objective function a method of a class:

```
class ChiSquared:                    # function to be minimized

    def __init__(self, xHist, bin_edges, fitType):
        self.setData(xHist, bin_edges)
        self.fitType = fitType

    def setData(self, xHist, bin_edges):
        numVal = np.sum(xHist)
        numBins = len(xHist)
        binSize = bin_edges[1] - bin_edges[0]
        self.data = xHist, bin_edges, numVal, numBins, binSize

    def chi2LS(self, par):            # least squares
        xHist, bin_edges, numVal, numBins, binSize = self.data
        xMid = bin_edges[:numBins] + 0.5*binSize
        binProb = f(xMid, par)*binSize
        nu = numVal*binProb
        sigma = np.sqrt(nu)
        z = (xHist - nu)/sigma
        return np.sum(z**2)
```

← called when object created (= “constructor”)

class ChiSquared (continued)

```
def chi2M(self, par):                # multinomial maximum likelihood
    xHist, bin_edges, numVal, numBins, binSize = self.data
    xMid = bin_edges[:numBins] + 0.5*binSize
    binProb = f(xMid, par)*binSize
    nu = numVal*binProb
    lnL = 0.
    for i in range(len(xHist)):
        if xHist[i] > 0.:
            lnL += xHist[i]*np.log(nu[i]/xHist[i])
    return -2.*lnL

def __call__(self, par):
    if self.fitType == 'LS':
        return self.chi2LS(par)
    elif self.fitType == 'M':
        return self.chi2M(par)
    else:
        print("fitType not defined")
        return -1
```

← makes object itself behave like a callable function

Using the ChiSquared class

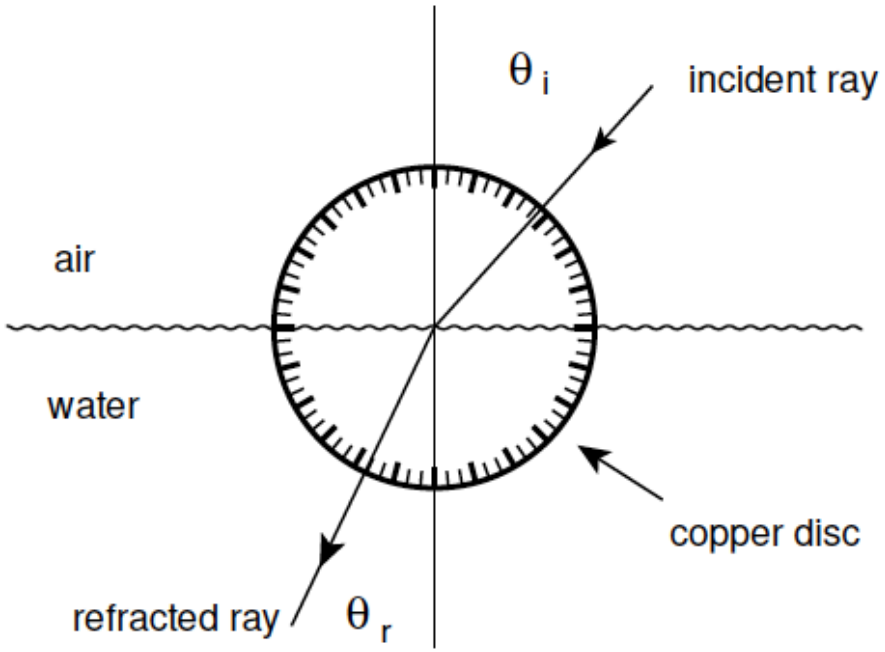
```
# Put data values into a histogram
numBins=40
xHist, bin_edges = np.histogram(xData, bins=numBins, range=(xMin, xMax))
binSize = bin_edges[1] - bin_edges[0]

# Initialize Minuit and set up fit:
parin   = np.array([theta, mu, sigma, xi])      # initial values (here = true)
parname = ['theta', 'mu', 'sigma', 'xi']
parstep = np.array([0.1, 1., 1., 1.])           # initial setp sizes
parfix  = [False, True, True, False]            # change to fix/free param.
parlim  = [(0.,1), (None, None), (0., None), (0., None)]
chisq = ChiSquared(xHist, bin_edges, fitType)
m = Minuit(chisq, parin, name=parname)
m.errors = parstep
m.fixed = parfix
m.limits = parlim
m.errordef = 1.0                        # errors from chi2 = chi2min + 1
```

For full program see
https://www.pp.rhul.ac.uk/~cowan/stat/exercises/fitting/python/

LS example: refraction data from Ptolemy

Astronomer Claudius Ptolemy obtained data on refraction of light by water in around 140 A.D.:



Angles of incidence and refraction (degrees)

θ_i	θ_r
10	8
20	$15\frac{1}{2}$
30	$22\frac{1}{2}$
40	29
50	35
60	$40\frac{1}{2}$
70	$45\frac{1}{2}$
80	50

Suppose the angle of incidence is set with negligible error, and the measured angle of refraction has a standard deviation of ½° .

Laws of refraction

A commonly used law of refraction was

$$\theta_{\rm r} = \alpha\theta_{\rm i} \ ,$$

although it is reported that Ptolemy preferred

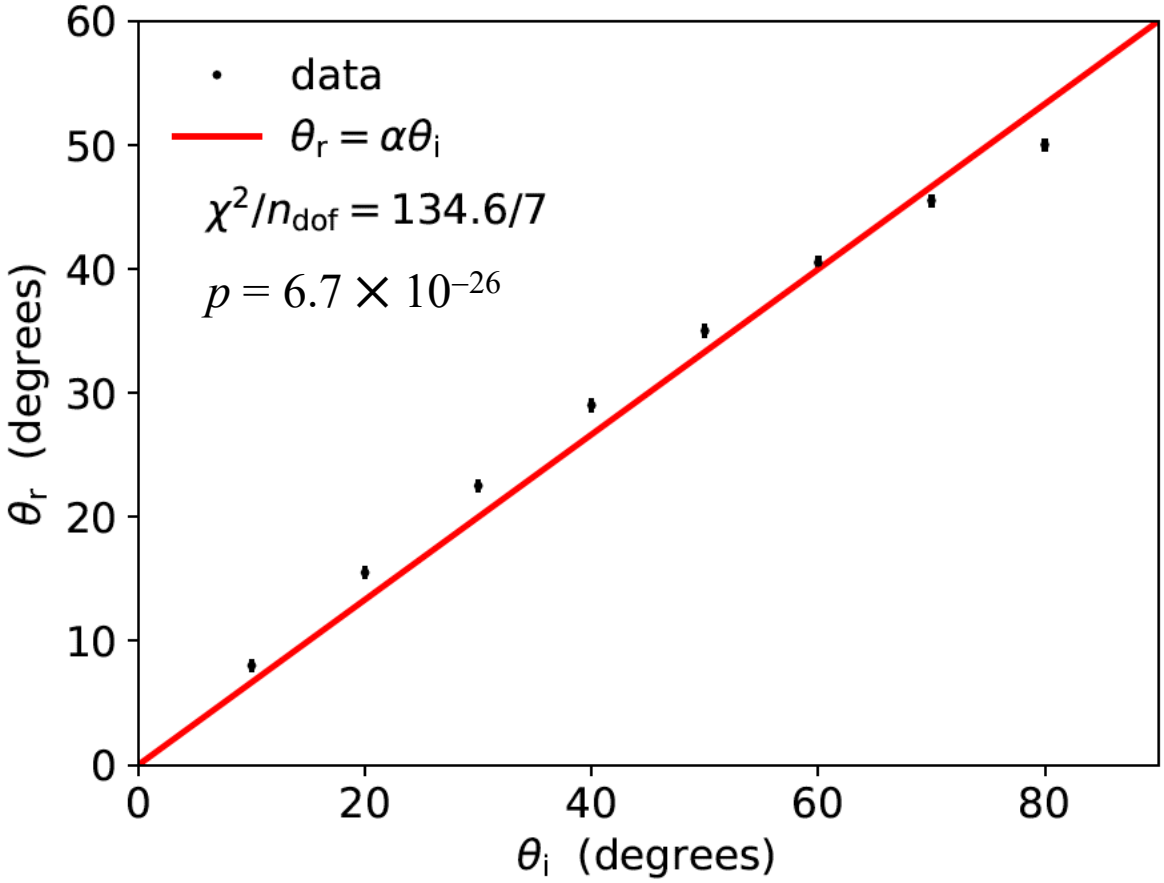
$$\theta_{\rm r} = \alpha\theta_{\rm i} - \beta\theta_{\rm i}^2 \ .$$

The law of refraction discovered by Ibn Sahl in 984 (and rediscovered by Snell in 1621) is

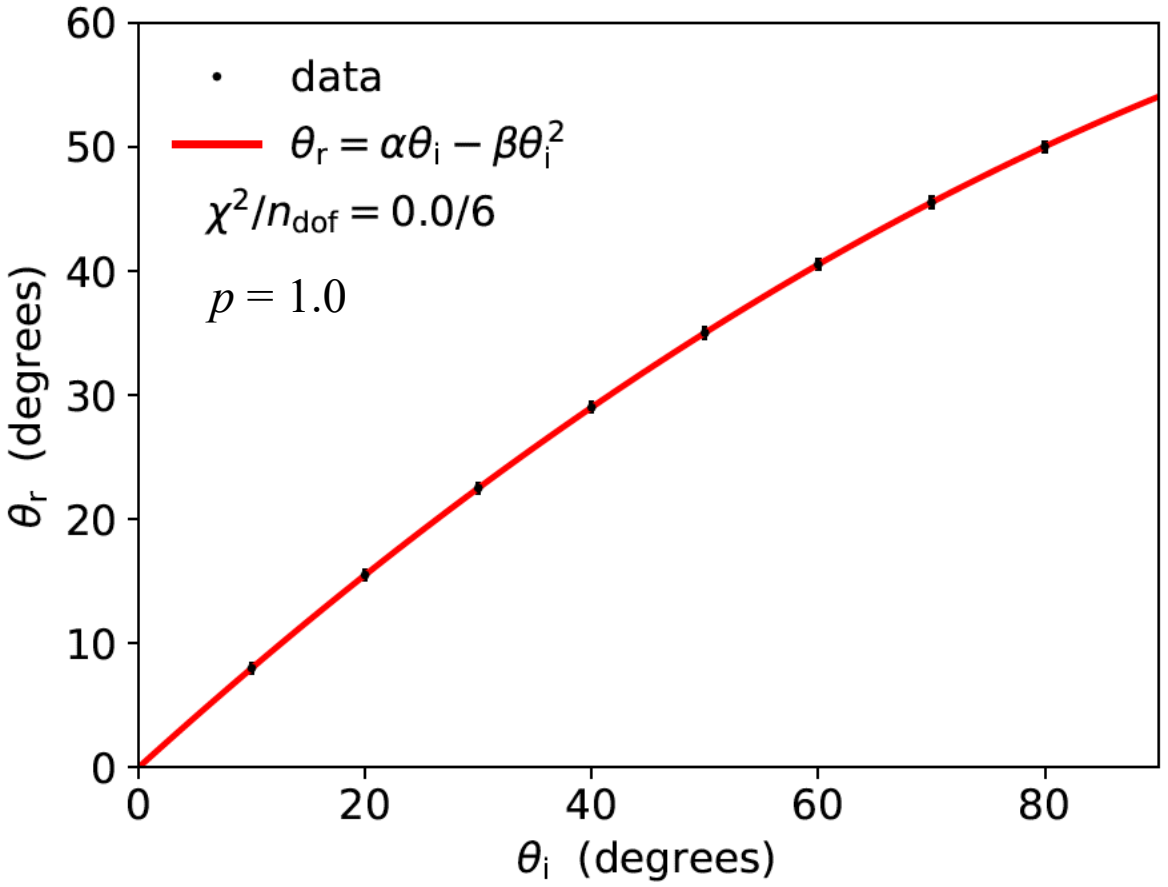
$$\theta_{\rm r} = \sin^{-1}\left(\frac{\sin\theta_{\rm i}}{r}\right) .$$

where $r = n_{\rm r}/n_{\rm i}$ is the ratio of indices of refraction of the two media.

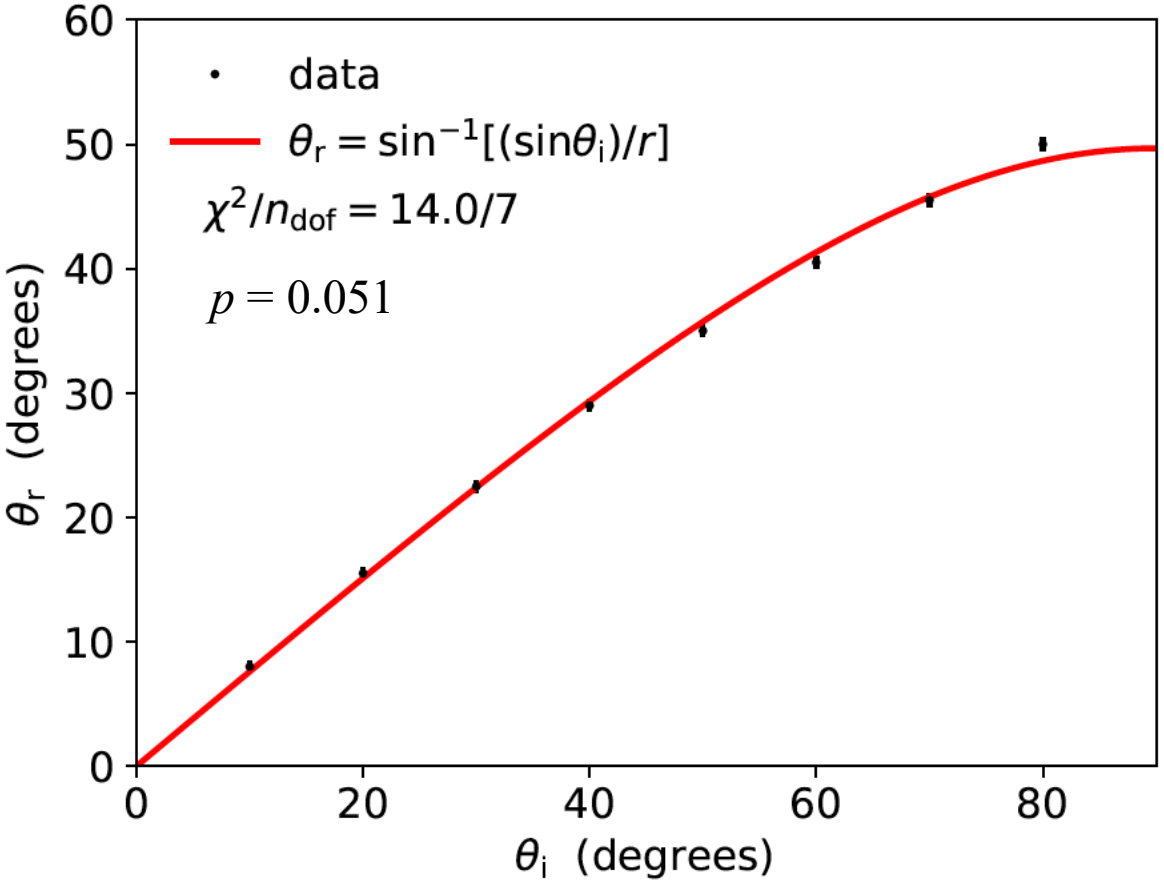
LS fit: $\theta_r = \alpha\theta_i$



LS fit: $\theta_r = \alpha\theta_i - \beta\theta_i^2$



LS fit: Snell's Law



Fitted index of refraction of water r = 1.3116 ± 0.0056 found not quite compatible with currently known value 1.330.