

Error Estimation

Source: Richter "Estimating Errors in Least Square Fitting"

http://ipnpr.jpl.nasa.gov/progress_report/42-122/122E.pdf

The problem:

Data of the form $(x_i, y_i), i \in (1, \dots, N)$ is fitted by the function $y(x; a_1, \dots, a_M) \equiv y(x; \mathbf{a})$.

To determine the coefficients a_j , **it is sought to minimize**

$$\chi^2(\mathbf{a}) = \sum_{i=1}^N \frac{(y_i - y(x_i; \mathbf{a}))^2}{\sigma_i^2}$$

where σ_i is the standard deviation of the random errors of y_i (which are assumed to be normally distributed).

\mathbf{a}_0 is a coefficient vector that minimizes χ^2 .

The variances of the elements a_j are given by $\sigma_{a_j}^2 = C_{jj}$: the diagonal elements of the covariance matrix \mathbf{C} which is itself the inverse of the curvature or Hessian matrix \mathbf{H} .

The *weighted mean value of the variance of the fit* is given by: $\frac{1}{N} \sum_{i=1}^N \frac{\sigma_y^2(x_i)}{\sigma_i^2} = \frac{M}{N}$ so that for constant data errors, **the mean standard error of the fit is:**

$$\overline{\sigma_y^2} = \frac{1}{N} \sum_{i=1}^N \sigma_y^2(x_i) = \frac{M}{N} \sigma^2$$

The **error in the value of the fitted function** is a function of x even when the σ_i are all the same and independent of x . Hence variance of the value of the fitted function (due to random data errors) is:

$$\sigma_y^2(x) = \sum_{j=1}^M \sum_{k=1}^M C_{jk} d_j(x) d_k(x) = \mathbf{d}(x)^T \mathbf{C} \mathbf{d}(x)$$

Where $d_j(x) = \left(\frac{\partial y(x; \mathbf{a})}{\partial a_j} \right)_{\mathbf{a}_0}$

For *linear* fitting where $y(x; \mathbf{a}) = \sum_{j=1}^M a_j X_j(x)$, $d_j(x) = X_j(x)$ and $\sigma_y^2(x) = \mathbf{x}(x)^T \mathbf{C} \mathbf{x}(x)$ where \mathbf{x} is a column vector with elements $X_j(x)$.

Standard Error of Fit:

What is $\sigma_y(x)$, the standard error of the fit?

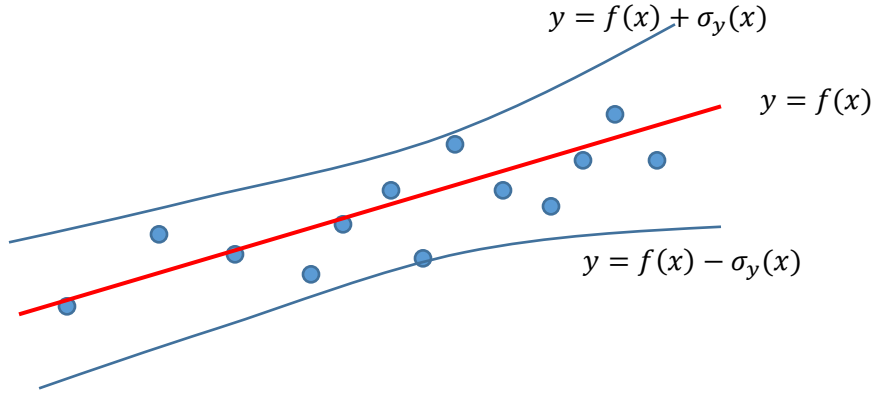


Figure 1: Illustration of standard error of fit $\sigma_y(x)$.

As is apparent from Figure 1, the standard error of fit is a very useful term! It is minimum at the centroid of the data points and maximum at the edges.

Least Squares Fitting:

$$\chi^2(\mathbf{a}) = \sum_{i=1}^N \frac{(y_i - y(x_i; \mathbf{a}))^2}{\sigma_i^2}$$

σ_i can be obtained from knowledge of experimental errors or from analysis of the data itself. It is assumed that the errors are normally distributed.

Linear Least Squares Fitting:

$$y(x; \mathbf{a}) = \sum_{j=1}^M a_j X_j(x)$$

$X_j(x)$ are arbitrary basis functions for the independent variable x .

In terms of linear algebra, the fitting function $y(x_i; \mathbf{a})$ is an N -element column vector, the coefficients a_j are elements of an M element column vector and the basis functions $X_j(x_i)$ are elements of an $N \times M$ matrix X_{ij} . Hence:

$$y(x_i; \mathbf{a}) = \sum_{j=1}^M a_j X_{ij}$$

i.e.

$$\mathbf{y} = \mathbf{X}\mathbf{a}$$

Lets now define the column vector $b_i = \frac{y_i}{\sigma_i}$ and the matrix $A_{ij} = \frac{X_{ij}}{\sigma_i}$. Hence:

$$\chi^2(\mathbf{a}) = (\mathbf{b} - \mathbf{Aa})^T (\mathbf{b} - \mathbf{Aa})$$

Extremum condition is: $\frac{\partial \chi^2}{\partial a_j} = 0$ i.e.¹

$$(\mathbf{A}^T \mathbf{A}) \mathbf{a} = \mathbf{A}^T \mathbf{b}$$

Hence:

$$\mathbf{a} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b} = \mathbf{C} \mathbf{A}^T \mathbf{b}$$

Here $\mathbf{H} = \mathbf{A}^T \mathbf{A}$ and $\mathbf{C} = \mathbf{H}^{-1}$ is the covariance matrix: a symmetric $M \times M$ matrix.

Hence:

$$a_j = a_j(y_1, y_2, \dots, y_N) = \sum_{k=1}^N C_{jk} \sum_{i=1}^N y_i \frac{X_k(x_i)}{\sigma_i^2}$$

Suitability of Basis Functions:

The value of χ^2 should be of the order $N - M \equiv \nu$: the number of **degrees of freedom** of this system. Hence

$$\chi_\nu^2 = \frac{\chi^2}{\nu} \approx 1$$

This is the condition for the fit to be meaningful. If $\nu \gg 1$, χ^2 is normally distributed with $\frac{\chi^2}{\nu}$ having mean 1 and standard deviation of $\sqrt{\frac{2}{\nu}}$.

¹ $\frac{\partial \chi^2}{\partial a_j} = (\mathbf{b} - \mathbf{Aa})^T \frac{\partial (\mathbf{b} - \mathbf{Aa})}{\partial a_j} + \frac{\partial (\mathbf{b} - \mathbf{Aa})^T}{\partial a_j} (\mathbf{b} - \mathbf{Aa}) = -(\mathbf{b} - \mathbf{Aa})^T \mathbf{A} \frac{\partial \mathbf{a}}{\partial a_j} - \frac{\partial \mathbf{a}^T}{\partial a_j} \mathbf{A}^T (\mathbf{b} - \mathbf{Aa}) = -(\mathbf{b} - \mathbf{Aa})^T \mathbf{A} \frac{\partial \mathbf{a}}{\partial a_j} - (\mathbf{b} - \mathbf{Aa})^T \left(\frac{\partial \mathbf{a}^T}{\partial a_j} \mathbf{A}^T \right)^T = -(\mathbf{b} - \mathbf{Aa})^T \mathbf{A} \frac{\partial \mathbf{a}}{\partial a_j} - (\mathbf{b} - \mathbf{Aa})^T \mathbf{A} \frac{\partial \mathbf{a}}{\partial a_j} = -2(\mathbf{b} - \mathbf{Aa})^T \mathbf{A} \frac{\partial \mathbf{a}}{\partial a_j}$. Hence $\frac{\partial \chi^2}{\partial \mathbf{a}} = -2(\mathbf{b} - \mathbf{Aa})^T \mathbf{A} = -2\mathbf{b}^T \mathbf{A} + 2(\mathbf{Aa})^T \mathbf{A} = -2\mathbf{A}^T \mathbf{b} + 2\mathbf{A}^T \mathbf{Aa}$

Standard Errors

$$\delta a_j = \sum_{i=1}^N \frac{\partial a_j}{\partial y_i} \delta y_i = \sum_{k=1}^N c_{jk} \sum_{i=1}^N \frac{X_k(x_i)}{\sigma_i^2} \delta y_i$$

In matrix form:

$$\delta \mathbf{a} = \mathbf{C} \mathbf{A}^T \delta \mathbf{b}$$

The covariance of a_j and a_k is given by:

$$\sigma_a^2 = \langle \delta \mathbf{a} \delta \mathbf{a}^T \rangle = \mathbf{C} \mathbf{A}^T \langle \delta \mathbf{b} \delta \mathbf{b}^T \rangle \mathbf{A} \mathbf{C}^T$$

Since $\langle \delta y_i \delta y_k \rangle = \delta_{ik}$ (errors are uncorrelated), hence: $\langle \delta \mathbf{b} \delta \mathbf{b}^T \rangle = \mathbf{I}$. Hence²,

$$\sigma_a^2 = \mathbf{C} (\mathbf{A}^T \mathbf{A}) \mathbf{C}^T = \mathbf{C}$$

Hence, the **errors in the coefficients are correlated!**

The variance of the coefficients are given by the **diagonal** elements:

$$\sigma_{a_j}^2 = C_{jj}$$

Hence:

$$\delta y(x; \mathbf{a}) = \sum_{j=1}^M \delta a_j X_j(x)$$

Hence the covariance:

$$\sigma_y^2(x, x') = \sum_{j=1}^M \sum_{k=1}^M \langle \delta a_j \delta a_k \rangle X_j(x) X_k(x') = \sum_{j=1}^M \sum_{k=1}^M C_{jk} X_j(x) X_k(x')$$

This is independent of parameter values a_j . For the special case where $x = x'$:

$$\sigma_y^2(x) = \sum_{j=1}^M \sum_{k=1}^M C_{jk} X_j(x) X_k(x) = \mathbf{X}^T \mathbf{C} \mathbf{X}$$

Here \mathbf{X} is a column vector whose elements are $X_j(x)$.

Hence, the **errors in $y(x; \mathbf{a})$ at two different values of x are correlated!**

The **weighted mean value of $\sigma_y^2(x)$ over all x** is given by:

$$\overline{\sigma_y^2} = \frac{1}{N} \sum_{i=1}^N \frac{\sigma_y^2(x_i)}{\sigma_i^2} = \sum_{j=1}^M \sum_{k=1}^M C_{jk} \frac{1}{N} \sum_{i=1}^N \frac{X_j(x_i)}{\sigma_i} \frac{X_k(x_i)}{\sigma_i}$$

² $\mathbf{C} = (\mathbf{A}^T \mathbf{A})^{-1}$ and $\mathbf{C}^T = \mathbf{C}$

But $\frac{x_{ij}}{\sigma_i} = A_{ij}$. Hence:

$$\overline{\sigma_y^2} = \frac{1}{N} \sum_{i=1}^N \frac{\sigma_y^2(x_i)}{\sigma_i^2} = \sum_{j=1}^M \sum_{k=1}^M C_{jk} \frac{1}{N} \sum_{i=1}^N A_{ij} A_{ik} = \sum_{j=1}^M \sum_{k=1}^M C_{jk} \frac{1}{N} \sum_{i=1}^N A_{ji}^T A_{ik}$$

Hence:

$$\overline{\sigma_y^2} = \sum_{j=1}^M \sum_{k=1}^M C_{jk} \frac{1}{N} (A^T A)_{jk} = \frac{1}{N} \sum_{j=1}^M \sum_{k=1}^M C_{jk} (C^{-1})_{jk} = \frac{1}{N} \sum_{j=1}^M I_{jj} = \frac{M}{N}$$

Hence if $\sigma_i = \sigma$,

$$\overline{\sigma_y^2} = \frac{1}{N} \sum_{i=1}^N \sigma_y^2(x_i) = \frac{M}{N} \sigma^2$$

This is analogous to the variance of the mean for the case of one value of x where $M=1$. Hence, as a result of fitting to the function $y(x)$, **the variance of y is reduced by $\frac{M}{N}$** . Hence the higher the value of M required to fit the data to reduce χ_v^2 to a value close to 1, the larger will be the errors in the values of the fitted function.

Very Important: For the fit to be meaningful:

1. $\chi_v^2 \approx 1$
2. Data errors (σ_i) have to be correctly chosen otherwise fit is not meaningful even with $\chi_v^2 \approx 1$.

Only if the fit is meaningful can one estimate the fitting errors meaningfully.

Non-Linear Least Squares Problem

No explicit solution for \mathbf{a} exists and an iterative procedure must be followed.

$$y(x; \mathbf{a}) = y(x; \mathbf{a}_0) + \sum_{j=1}^M \frac{\partial y(x; \mathbf{a})}{\partial a_j} \bigg|_{\mathbf{a}_0} (a_j - a_{0j}) = y(x; \mathbf{a}_0) + (\mathbf{a} - \mathbf{a}_0)^T \mathbf{d}(x; \mathbf{a}_0)$$

Where $d_j(x; \mathbf{a}_0) = \frac{\partial y(x; \mathbf{a})}{\partial a_j} \bigg|_{\mathbf{a}_0}$

And similarly:

$$\chi^2(\mathbf{a}) = \chi_{min}^2 + \frac{1}{2} \sum_{j=1}^M \sum_{k=1}^M \frac{\partial^2 \chi^2(\mathbf{a})}{\partial a_j \partial a_k} \bigg|_{\mathbf{a}_0} (a_j - a_{0j})(a_k - a_{0k}) = \chi_{min}^2 + (\mathbf{a} - \mathbf{a}_0)^T \mathbf{H}(\mathbf{a}_0)(\mathbf{a} - \mathbf{a}_0)$$

Here:

$$h_{jk}(\mathbf{a}_0) = \frac{1}{2} \frac{\partial^2 \chi^2(\mathbf{a})}{\partial a_j \partial a_k} \bigg|_{\mathbf{a}_0} = \sum_{i=1}^N \frac{1}{\sigma_i^2} \frac{\partial y(x; \mathbf{a})}{\partial a_j} \bigg|_{\mathbf{a}_0} \frac{\partial y(x; \mathbf{a})}{\partial a_k} \bigg|_{\mathbf{a}_0}$$

If we define:

$$A_{ij}(\mathbf{a}_0) = \frac{1}{\sigma_i} \frac{\partial y(x_i; \mathbf{a})}{\partial a_j} \Big|_{\mathbf{a}_0} = \frac{d_{ij}(\mathbf{a}_0)}{\sigma_i}$$

Then: $\mathbf{H} = \mathbf{A}^T \mathbf{A}$, just like for linear least squares. The derivation then³, on similar lines, leads to:

$$\sigma_a^2(\mathbf{a}_0) = \mathbf{C}(\mathbf{a}_0) = \mathbf{H}^{-1}(\mathbf{a}_0)$$

Hence also,

$$\sigma_y^2(x; \mathbf{a}_0) = \sum_{j=1}^M \sum_{k=1}^M C_{jk}(\mathbf{a}_0) d_j(x; \mathbf{a}_0) d_k(x; \mathbf{a}_0) = \mathbf{d}^T \mathbf{C} \mathbf{d}$$

i.e.

$$\overline{\sigma_y^2} = \frac{M}{N} \sigma^2$$

The standard error of regression is: $\sigma_R = \sqrt{\frac{\chi^2}{M-N}} = \sqrt{\chi_v^2}$

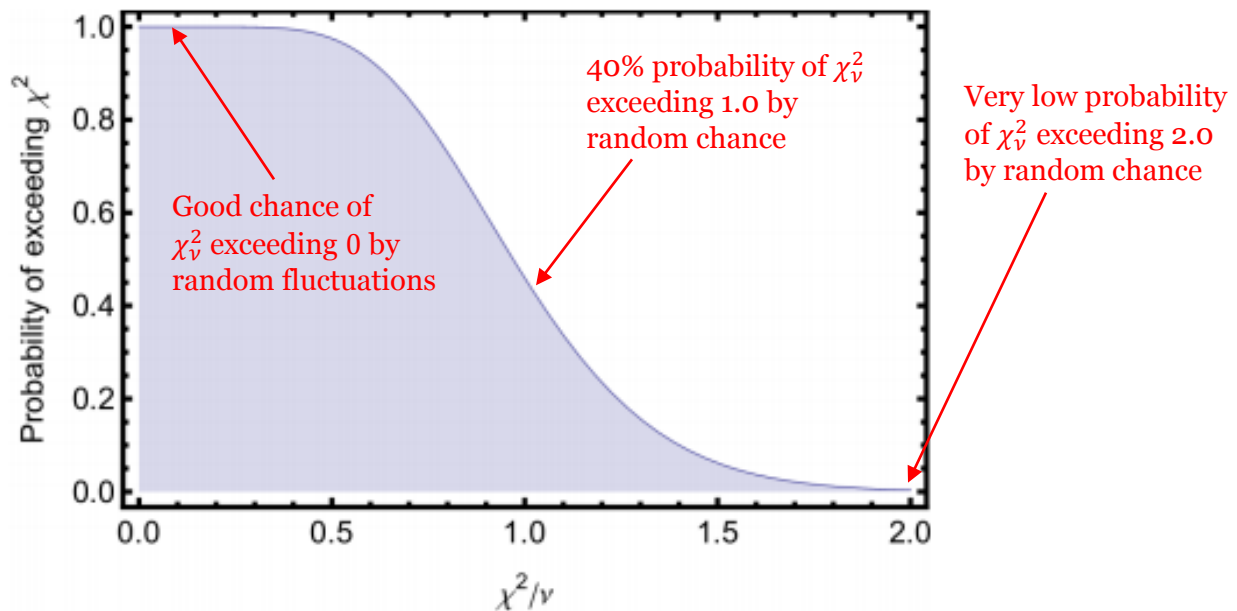
³ Here d_{ij} replaces X_{ij} of linear least squares.

$\pm\sigma_x$ is the 68.3% confidence limit

$\pm2\sigma_x$ is the 95.4% confidence limit

$\pm3\sigma_x$ is the 99.73% confidence limit

Note: Notice that σ_a and $\sigma_y(x)$ are not functions of $\chi^2(a_0)$. This is because the former are only meaningful if $\chi^2_v \approx 1$ i.e. the χ^2 dependence is already baked in.



Null Hypothesis: The fit explains the data.

$\chi^2_v \gg 1$ indicates a poor model fit: probability that the value of χ^2_v occurred by random chance is very low.

$\chi^2_v > 1$ indicates that fit has not fully captured the data but null hypothesis cannot be rejected.

$\chi^2_v = 1$ indicates that the match between observation and estimate is in accord with the error variance. Null hypothesis cannot be rejected.

$\chi^2_v < 1$ indicates *over-fitting* of data i.e. model is (improperly) fitting noise or the error variance has been over-estimated. Null hypothesis cannot be rejected.

Shapiro-Wilk test can be used to see if the residuals are normally distributed. If they are, then the student-t distribution can be used to verify that the mean is statistically zero.

Getting p-value for coefficient $\neq 0$.

The t –statistic to test if a given coefficient a_j is different from zero is:

$t_j = \frac{a_j}{\sigma_{a_j}}$. Get the value of $p = 1.0 - t.cdf(t_j, M - N)$ and if this is less the 0.05, then the value of a_j is significantly different from zero.

User's Guide to Error Analysis

<http://www.colorado.edu/physics/EducationIssues/zwickl/Resources/Error%20Analysis%20Activity%202.pdf>

What is least squares fitting:

$$\chi^2 = \sum_{i=1}^N \frac{1}{\sigma_i^2} (y - y(x_i, \mathbf{a}))^2$$

Where there are N data points (x_i, y_i) and the fit function is given by $y(x, \mathbf{a})$ where \mathbf{a} is a vector of the fit parameters.

Assumptions:

1. **Gaussian Distribution** of the random fluctuations in each y_i .
2. **Uncorrelated:** the random fluctuations in any one data point are uncorrelated with another data point.

If these two assumptions hold, minimizing χ^2 gives the *most likely* function that reproduces the observed data.

Uncertainty in the data σ_y^2 is calculated using the **residuals of the best fit**:

$$\sigma_y^2 = \frac{1}{N - n} \sum_{i=1}^N (y_i - y(x_i; \mathbf{a}))^2$$

This is analogous to finding the variance of a repeated measurement.

Note: This was not calculated in Richter's analysis.

The residuals must be randomly distributed about zero (no systematic variation).

Note: It is **not necessary** for a good fit line to pass through each set of error bars: though it should pass through most. If we have shown the error bars for standard error, then 68% of data points are expected to lie within their error bar.

Applying the chi-squared test:

The graph of the probability of $\chi^2_\nu = \frac{\chi^2}{\nu}$ exceeding the observed value for ν degrees of freedom is available (cumulative density function CDF of the chi-squared distribution). This gives us the likelihood that the observed value of χ^2 occurred by chance.

The **null-hypothesis** for a chi-squared test is that the data are **independent** and **normally-distributed**. The **chi-squared** test for goodness-of-fit is used to *reject* the null hypothesis that the data are independent.

Note: It is possible to get a good looking χ^2_ν by overestimating experimental errors (σ_i).

Interpreting Sum of Squares

<http://facweb.cs.depaul.edu/sjost/csc423/documents/f-test-reg.htm>

[http://reliawiki.org/index.php/Simple Linear Regression Analysis](http://reliawiki.org/index.php/Simple_Linear_Regression_Analysis)

For M observations and N regression parameters:

(1) Corrected Sum of Squares also called Sum of Squares of Regression :

$$SSM \text{ (or } SSR) = \sum_{i=1}^M \frac{(y(x_i; \mathbf{a}) - \bar{y})^2}{\sigma_i^2}$$
$$\bar{y} = \frac{1}{M} \sum_{i=1}^M y_i$$

This is a measure of **explained variation**

(2) Sum of Squares for Error:

$$SSE = \sum_{i=1}^M \frac{(y_i - y(x_i; \mathbf{a}))^2}{\sigma_i^2}$$

This is a measure of **unexplained variation**

(3) Corrected Sum of Squares (Total):

$$SST = \sum_{i=1}^M \frac{(y_i - \bar{y})^2}{\sigma_i^2}$$

(4) For multiple regression models:

$$SST = SSM + SSE$$
$$R^2 = \frac{SSM \text{ (or } SSR)}{SST}$$

(5) Corrected Degrees of Freedom for Model:

$$DFM = N - 1$$

(6) Degrees of Freedom for Error:

$$DFE = M - N$$

(7) Corrected Degrees of Freedom Total:

$$DFT = DFM + DFE = M - 1$$

(8) Mean of Squares of Model:

$$MSM = \frac{SSM}{DFM}$$

(9) Mean of Squares for Error:

$$MSE = \frac{SSE}{DFE}$$

(10) Mean of Squares Total:

$$MST = \frac{SST}{DFT}$$

It is desirable to have MSM be large wrt MSE.

F-test

We want to test the following null hypothesis:

$$H_0: \beta_1 = \beta_2 = \dots = \beta_{N-1} = 0$$

$$H_1: \beta_j \neq 0 \text{ for at least one value of } j$$

Steps of the F-test.

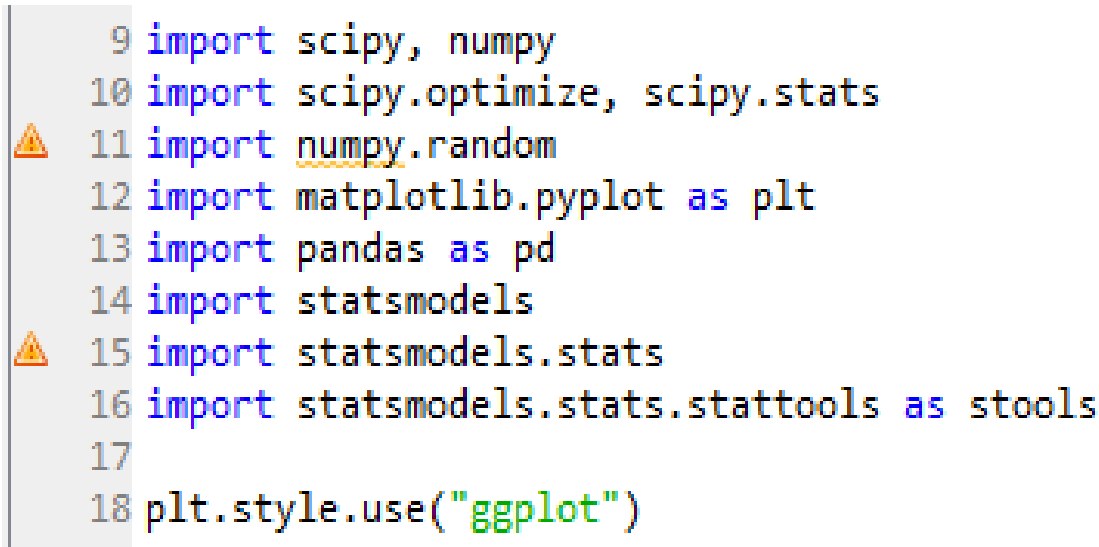
1. State the null and alternative hypothesis.
2. Compute test statistics assuming null-hypothesis is true:

$$F = \frac{MSM}{MSE} = \frac{\text{explained variance}}{\text{unexplained variance}}$$
3. Find a $(1-\alpha)*100\%$ confidence interval I for (DFM, DFE) degrees of freedom using an F-table or statistical software.
4. Accept the null-hypothesis if $\subset I$; else reject.
5. Use statistical software to determine the p value.

Implementation in Python

To implement this in Python, let's open a new file called *errorestimation.py*.

We need to import several things. See Figure 2



```
9 import scipy, numpy
10 import scipy.optimize, scipy.stats
11 import numpy.random
12 import matplotlib.pyplot as plt
13 import pandas as pd
14 import statsmodels
15 import statsmodels.stats
16 import statsmodels.stats.stattools as stools
17
18 plt.style.use("ggplot")
```

Figure 2: Various libraries that we need to import for analysis of a curve-fitting.

We define a function *fitdata* as shown in Figure 3. The inputs of the function are explained in the docstring (the green text in the triple quotes).

We will use the Levenberg-Marquadt algorithm to optimize the **p** (see discussion at the beginning of this chapter). This is a *non-linear* least square fitting.

The code corresponding to this is *within* the *fitdata* function's block. It appears in Figure 4. Sums of squares of various deviations appear in Figure 5 and analysis of goodness of fit appears in Figure 6.

The code for the plotting appears in Figure 7. The calculation of the standard error of the fit appears in Figure 8 and the code for formatting the axes in Figure 9.

See Figure 10, Figure 11 and Figure 12 for an illustration of importing, preparing and fitting data from an Excel sheet.

```

80 def fitdata(f, Xdata, Ydata, Errdata, pguess, dict_data, ax = False, ax2 = False):
81     ...
82     fitdata(f, Xdata, Ydata, Errdata, pguess, dict_data)
83     f = function f(X, p, dict_data)
84     Xdata = array like object (k, M) shaped array for data with k predictors
85     e.g. if X = (x1, x2, x3) then X = (X1, X2, X3) where X1 is a vector of x1 etc
86     Ydata = array like object of length M
87     Errdata = array like object of length M: error estimate of ydata.
88     pguess = array like object of length N (vector of guess of parameters)
89     dict_data = dictionary containing other data necessary for f
90     Returns:
91     popt = vector of length N of the optimized parameters
92     pcov = Covariance matrix of the fit
93     perr = vector of length N of the std-dev of the optimized parameters
94     p95 = half width of the 95% confidence interval for each parameter i.e. popt-p95 and popt+p95
95     p_p = vector of length N of the p-value for the parameters being zero
96     (if p<0.05, null hypothesis rejected and parameter is non-zero)
97     chisquared = (chisquared, chisquared_red, degfreedom, p)
98     chisquared = chisquared value for the fit: sum of squares of weighted residuals
99     chisquared_red = chisquared/degfreedom. Value should be approx. 1 for a good fit.
100    degfreedom = M - N the degrees of freedom of the fitting
101    chisquare = (p, chisquared, chisquared_red, degfreedom)
102    p = Probability of finding a chisquared value at least as extreme as the one shown
103    purely by random chance (should be high for good fit)
104    chisquared = chisquared value for the fit: sum of squares of weighted residuals
105    chisquared_red = chisquared/degfreedom. Value should be approx. 1 for a good fit.
106    degfreedom = M - N the degrees of freedom of the fitting
107    R2 = correlation coefficient or proportion of explained variance
108    R2_adj = adjusted R2 taking into account number of predictors
109    resanal = (p, w, mean, stddev) Analysis of residuals
110    p = Probability of finding a w at least as extreme as the one observed (should be high for good fit)
111    w = Shapiro-Wilk test criterion
112    mean = mean of residuals
113    p_res = probability that the mean value obtained is different from zero merely by chance
114    (should be low for good fit)
115    The mean must be within 1 stddev of zero for highly significant fitting.
116    F = F-statistic for the fit MSM/MSE.
117    Null hypothesis is that there is NO Difference between the two variances.
118    p_F = probability that this value of F can arise by chance alone.
119    p_F < 0.05 to reject null hypothesis and prove that the fit is good.
120    dw = Durbin_Watson statistic (value between 0 and 4).
121    2 = no-autocorrelation. 0 = +ve autocorrelation, 4 = -ve autocorrelation.
122    ...

```

Figure 3: Function `fitdata` and its docstring explaining its inputs and outputs.

```

...
def error(p, Xdata, Ydata, Errdata, dict_data):
    Y = f(Xdata, p, dict_data)
    residuals = (Y - Ydata)/Errdata
    return residuals
res = scipy.optimize.leastsq(error, pguess, args=(Xdata, Ydata, Errdata, dict_data), full_output=1)
(popt, pcov, infodict, errmsg, ier) = res
perr = scipy.sqrt(scipy.diag(pcov))

```

Figure 4: Code to optimize the parameters **p** using the `scipy.optimize.leastsq` module with `full_output = 1`. Also shown is **perr**: the vector of the standard-deviation for each of the **p** values.

```

132
133 M = len(Ydata)
134 N = len(popt)
135 #Residuals
136 Y = f(Xdata, popt, dict_data)
137 residuals = (Y - Ydata)/Errdata
138 meanY = scipy.mean(Ydata)
139 squares = (Y - meanY)/Errdata
140 squaresT = (Ydata - meanY)/Errdata
141
142 SSM = sum(squares**2) #Corrected Sum of Squares
143 SSE = sum(residuals**2) #Sum of Squares of Errors
144 SST = sum(squaresT**2) #Total corrected sum of squares
145
146 DFM = N - 1 #Degrees of freedom for model
147 DFE = M - N #Degrees of freedom for error
148 DFT = M - 1 #Degrees of freedom total
149
150 MSM = SSM/DFM #Mean squares for model (explained variance)
151 MSE = SSE/DFE #Mean squares for Error (should be small wrt MSM) Unexplained variance
152 MST = SST/DFT #Mean squares for total
153
154 R2 = SSM/SST #proportion of explained variance
155 R2_adj = 1 - (1 - R2)*(M - 1)/(M - N - 1) #Adjusted R2
156

```

Figure 5: Code to analyse the sum of squares of various deviations of the fit. See the part about the F-test above.

```

157 #t-test to see if parameters are different from zero
158 t_stat = popt/perr #t-statistic for popt different from zero
159 t_stat = t_stat.real
160 p_p = 1.0 - scipy.stats.t.cdf(t_stat, DFE) #should be Low for good fit.
161 z = scipy.stats.t(M-N).ppf(0.95)
162 p95 = perr*z
163 #Chisquared Analysis on Residuals
164 chisquared = sum(residuals**2)
165 degfreedom = M - N
166 chisquared_red = chisquared/degfreedom
167 p_chi2 = 1.0 - scipy.stats.chi2.cdf(chisquared, degfreedom)
168 stderr_reg = scipy.sqrt(chisquared_red)
169 chisquare = (p_chi2, chisquared, chisquared_red, degfreedom, R2, R2_adj)
170
171 #Analysis of residuals
172 w, p_shapiro = scipy.stats.shapiro(residuals)
173 mean_res = scipy.mean(residuals)
174 stddev_res = scipy.sqrt(scipy.var(residuals))
175 t_res = mean_res/stddev_res #t-statistic to test that mean_res is zero.
176 p_res = 1.0 - scipy.stats.t.cdf(t_res, M-1)
177 #if p_res < 0.05, null hypothesis rejected and mean is non-zero.
178 #Should be high for good fit.
179 #F-test on residuals
180 F = MSM/MSE #explained variance/unexplained . Should be Large
181 p_F = 1.0 - scipy.stats.f.cdf(F, DFM, DFE)
182 #if p_F < 0.05, null-hypothesis is rejected
183 #i.e. R^2 > 0 and at least one of the fitting parameters > 0.
184 dw = stats.durbin_watson(residuals)
185 resanal = (p_shapiro, w, mean_res, p_res, F, p_F, dw)
186

```

Figure 6: Test to see goodness of fit. p_{95} is the vector of the 95% confidence range of p i.e. $p = p \pm p_{95}$ with 95% confidence. Shapiro test is to check if the residuals are normally distributed and the Durbin-Watson test is to check if they are correlated. $p_shapiro$ should be > 0.05 and dw should be near 2 and away from 0 and 4.

```

189 if ax:
190     formataxis(ax)
191     ax.plot(Ydata, Y, 'ro')
192     ax.errorbar(Ydata, Y, yerr = Errdata, fmt='.')
193     Ymin, Ymax = min((min(Y),min(Ydata))), max((max(Y),max(Ydata)))
194     ax.plot([Ymin, Ymax],[Ymin, Ymax], 'b')
195
196     ax.xaxis.label.set_text('Data')
197     ax.yaxis.label.set_text('Fitted')
198
199     sigmay, avg_stddev_data = get_stderr_fit(f, Xdata, popt, pcov, dict_data)
200     Yplus = Y + sigmay
201     Yminus = Y - sigmay
202     ax.plot(Y, Yplus, 'c', alpha = 0.6, linestyle = '--', linewidth = 0.5)
203     ax.plot(Y, Yminus, 'c', alpha = 0.6, linestyle = '--', linewidth = 0.5)
204     ax.fill_between(Y, Yminus, Yplus, facecolor = 'cyan', alpha = 0.5)
205     titletext = 'Parity plot for fit.\n'
206     titletext += r'$r^2$ = %5.2f, $r^2_{adj}$ = %5.2f, '
207     titletext += '$\sigma_{exp}$ = %5.2f, $\chi^2_{\nu}$ = %5.2f, $p_{\chi^2}$ = %5.2f, '
208     titletext += '$\sigma_{err}^{reg}$ = %5.2f'
209
210     ax.title.set_text(titletext%(R2, R2_adj, avg_stddev_data, chisquared_red, p_chi2, stderr_reg))
211     ax.figure.canvas.draw()
212
213 if ax2: #Test for homoscedasticity
214     formataxis(ax2)
215     ax2.plot(Y, residuals, 'ro')
216
217     ax2.xaxis.label.set_text('Fitted Data')
218     ax2.yaxis.label.set_text('Residuals')
219
220     titletext = 'Analysis of Residuals\n'
221     titletext += r'mean = %5.2f, $p_{res}$ = %5.2f, $p_{shapiro}$ = %5.2f, $Durbin-Watson$=%2.1f'
222     titletext += '\n F = %5.2f, $p_F$ = %3.2e'
223     ax2.title.set_text(titletext%(mean_res, p_res, p_shapiro, dw, F, p_F))
224
225     ax2.figure.canvas.draw()
226
227
228 return popt, pcov, perr, p95, p_p, chisquare, resanal

```

Figure 7: Plotting and return

```

34
35 def get_stderr_fit(f, Xdata, popt, pcov, dict_data):
36     '''
37     popt, pcov, perr, chisquare, shapiro = get_stderr_fit(f, Xdata, popt, pcov, dict_data)
38
39     f = function f(X, p, dict_data)
40     Xdata = array like object (k, M) shaped array for data with k predictors
41     e.g. if X = (x1, x2, x3) then X =(X1, X2, X3) where X1 is a vector of x1 etc
42     Ydata = array like object of length M
43     popt = vector of length N of the optimized parameters
44     pcov = Covariance matrix of the fit
45     dict_data = dictionary containing other data necessary for f
46
47     returns
48     sigmay: array like object of length M. The standard deviations for error at a given value of X
49     avg_stddev_data: The constant standard deviation (experimental error) that would justify the given fit.
50     '''
51     Y = f(Xdata, popt, dict_data)
52     listdY = []
53     for i in xrange(len(popt)):
54         p = popt[i]
55         dp = abs(p)/1e6 + 1e-20
56         popt[i] += dp
57         Yi = f(Xdata, popt, dict_data)
58         dY = (Yi - Y)/dp
59         listdY.append(dY)
60         popt[i] -= dp
61     listdY = scipy.array(listdY)
62     #listdY is an array with N rows and M columns N = len(popt), M = len(Xdata[0])
63     #pcov is an array with N rows and N columns
64     left = scipy.dot(listdY.T, pcov)
65     #left is an array with M rows and N columns
66     right = scipy.dot(left, listdY)
67     #right is an M x M matrix. The diagonals of this are what we need.
68     sigma2y = right.diagonal()
69     #sigmay is the standard error of the fit and is a function of X
70     mean_sigma2y = scipy.mean(right.diagonal())
71     M = Xdata.shape[1]
72     N = len(popt)
73     avg_stddev_data = scipy.sqrt(M*mean_sigma2y/N)
74     #This is because if experimental error is constant at sig_dat, then mean_sigma2y = N/M*sig_dat**2
75     sigmay = scipy.sqrt(sigma2y)
76     return sigmay, avg_stddev_data
77

```

Figure 8: Code to calculate standard error of the fit.

```

20 def formataxis(ax):
21     ax.xaxis.label.set_fontname('Georgia')
22     ax.xaxis.label.set_fontsize(12)
23     ax.yaxis.label.set_fontname('Georgia')
24     ax.yaxis.label.set_fontsize(12)
25     ax.title.set_fontname('Georgia')
26     ax.title.set_fontsize(12)
27
28
29
30     for tick in ax.xaxis.get_major_ticks():
31         tick.label.set_fontsize(8)
32     for tick in ax.yaxis.get_major_ticks():
33         tick.label.set_fontsize(8)

```

Figure 9: Code to format axis.

```

229
230 def import_data(xlfile, sheetname):
231     df = pd.read_excel(xlfile, sheetname = sheetname)
232     return df
233
234 def prepare_data(df, Criterion, Predictors, Error = False):
235     Y = scipy.array(df[Criterion])
236     if Error:
237         Errdata = scipy.array(df[Error])
238     else:
239         Errdata = scipy.ones(len(Y))
240     Xdata = []
241     for X in Predictors:
242         X = list(df[X])
243         Xdata.append(X)
244     Xdata = scipy.array(Xdata)
245     return Xdata, Y, Errdata

```

Figure 10: Code for importing data from an excel file and preparing it for fitting. See Figure 11 for illustration.


```

246
247 if __name__ == "__main__":
248     fig = plt.figure()
249     ax = fig.add_subplot(111)
250     fig.show()
251
252     fig2 = plt.figure()
253     ax2 = fig2.add_subplot(111)
254     fig2.show()
255
256 # Make arbitrary function of three variables
257 def f(X, p, dict_data):
258     a = dict_data['a']
259     b = dict_data['b']
260     (x,y,z) = X
261     Y = p[0] + p[1]*x**2 + p[2]*y + p[3]*z
262     return Y
263
264
265 #Get data from excel file using pandas
266 df = import_data('SynthData.xlsx','Data')
267 Xdata, Ydata, Errdata = prepare_data(df, 'Ydata', ('x', 'y', 'z'), Error = 'err')
268
269 #Intital Guess
270 N = 4
271 pguess = N*[0.0]
272
273 popt, pcov, perr, p95, p_p, chisquare, resanal = fitdata(f, Xdata, Ydata, Errdata, pguess, dict_data, ax = ax, ax2 = ax2)
274

```

Figure 11: Illustration of use of importing, preparing and fitting data. The Excel file is named 'Synthdata.xlsx' and it contains a sheet called 'Data' formatted as shown in Figure 12

	A	B	C	D	E	F
	x	y	z	Ydata	err	
1	0	-1	-10	-26.312	10	
2	0.052632	-0.78947	-9.42105	-33.0895	10	
3	0.105263	-0.57895	-8.84211	-23.0898	10	
4	0.157895	-0.36842	-8.26316	-26.3104	10	
5	0.210526	-0.15789	-7.68421	-37.0378	10	
6	0.263158	0.052632	-7.10526	-37.9639	10	
7	0.315789	0.263158	-6.52632	-25.6197	10	
8	0.368421	0.473684	-5.94737	-3.45358	10	
9	0.421053	0.684211	-5.36842	-23.5125	10	
10	0.473684	0.894737	-4.78947	-36.7872	10	
11	0.526316	1.105263	-4.21053	-24.2072	10	
12	0.578947	1.315789	-3.63158	-11.0806	10	
13	0.631579	1.526316	-3.05263	12.43371	10	
14	0.684211	1.736842	-2.47368	1.646041	10	
15	0.736842	1.947368	-1.89474	22.93574	10	
16	0.789474	2.157895	-1.31579	21.03534	10	
17	0.842105	2.368421	-0.73684	19.66702	10	
18	0.894737	2.578947	-0.15789	20.93693	10	
19	0.947368	2.789474	0.421053	42.65305	10	
20	1	3	1	71.88136	10	
21						
22						
23						
24						

Figure 12: The Excel sheet used for the above problem. Note that the **first** row is only the headings and the headings are referenced in the code of Figure 11.