Subject: Re: CURVEFIT.PRO standard deviations?
Posted by Andrew Noymer on Mon, 13 May 2002 06:13:06 GMT
View Forum Message <> Reply to Message

> You can see the discrepancy in the one-sigma lines: can someone tell
> me what's up with the sigma returned from CURVEFIT, and how I can
> make them conform?

I'm not sure exactly what's going on, but I take an interest in this because I use these sorts of procedures.

I fed your code into IDL and modified it so the data were also written-out. I then fed the points into Stata (www.stata.com).

Here is what I found:

```
LINFIT parameters, sigma, and chi-square :
    -13.7844      2.91336
     1.32243    0.0944590
     266.783
CURVEFIT parameters, sigma, and chi-square :
    -13.7839      2.91333
    0.388221    0.0277362
     11.5993
```

So I cfm. that the parameters are the same (for all intents and purposes) between the two procedures, but the ch-sq and sigma is different. Here's what Stata gives me:

```
      Source |       SS       df       MS              Number of obs =      25
-------------+------------------------------           F(  1,    23) =  951.27
       Model |  11034.0003     1  11034.0003           Prob > F      =  0.0000
    Residual |  266.782989    23  11.5992604           R-squared     =  0.9764
-------------+------------------------------           Adj R-squared =  0.9754
       Total |  11300.7833    24   470.86597           Root MSE      =  3.4058

------------------------------------------------------------------------------
          v1 |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        var2 |   2.913364    .094459    30.84   0.000     2.717961    3.108768
       _cons |  -13.78436   1.322426   -10.42   0.000      -16.52   -11.04871
------------------------------------------------------------------------------
```

Stata's coefficient's are (of course) the same. What CURVEFIT calls chi-sq, Stata calls Residual MSE (mean sq. error). And it looks like LINFIT gives the Std. Errors of the coefficients that I would use if

I were you. The LINFIT ch-sq is what Stata calls the Residual SSE (sum sq. error).

I'm not sure how the CURVEFIT sigma values are calculated but I would not use them if I were you, without knowing exactly where they come from.

HTH.

Andrew

Subject: Re: CURVEFIT.PRO standard deviations?
Posted by Ivan Valtchanov on Mon, 13 May 2002 08:32:00 GMT
View Forum Message <> Reply to Message

Hi there,

Well, this is a known issue because in "What's new in IDL 5.4" there are some lines about LMFIT with correct estimation of sigmas:

..."In IDL versions prior to 5.4 the correction [sqrt(chisq/(n-m)), where n is the number of points and m is the number of coefficients] was not being applied."

So you have to multiply the sigmas by this correction factor and you'll get them right.

I suppose this is the case for CURVEFIT too.

Hope this helps?

Bye

--

 == ============
Ivan Valtchanov   e-mail:ivaltchanov@cea.fr
DSM/DAPNIA/Service d'Astrophysique
CEA/Saclay   tel: +33(0)1.69.08.34.92
Orme des Merisiers, bat.709   fax: +33(0)1.69.08.65.77
91191 Gif-sur-Yvette, France
 == ============

Subject: Re: CURVEFIT.PRO standard deviations?
Posted by Ralf Flicker on Mon, 13 May 2002 09:09:13 GMT
View Forum Message <> Reply to Message

Andrew Noymer wrote:
>
>> You can see the discrepancy in the one-sigma lines: can someone tell
>> me what's up with the sigma returned from CURVEFIT, and how I can
>> make them conform?
>
> I'm not sure exactly what's going on, but I take an interest in this because
> I use these sorts of procedures.
>
> I fed your code into IDL and modified it so the data were also written-out.
> I then fed the points into Stata (www.stata.com).
>
> Here is what I found:
>
> LINFIT parameters, sigma, and chi-square :
> -13.7844 2.91336
> 1.32243 0.0944590
> 266.783
> CURVEFIT parameters, sigma, and chi-square :
> -13.7839 2.91333
> 0.388221 0.0277362
> 11.5993
>
> So I cfm. that the parameters are the same (for all intents and
> purposes) between the two procedures, but the ch-sq and sigma
> is different. Here's what Stata gives me:
>
> Source | SS df MS Number of obs = 25
> ----------+---------- F(1, 23) = 951.27
> Model | 11034.0003 1 11034.0003 Prob > F = 0.0000
> Residual | 266.782989 23 11.5992604 R-squared = 0.9764
> ----------+---------- Adj R-squared = 0.9754
> Total | 11300.7833 24 470.86597 Root MSE = 3.4058
>
> --
> v1 | Coef. Std. Err. t P>|t| [95% Conf. Interval]
> ----------+---
> var2 | 2.913364 .094459 30.84 0.000 2.717961 3.108768
> _cons | -13.78436 1.322426 -10.42 0.000 -16.52 -11.04871
> --
>
> Stata's coefficient's are (of course) the same. What CURVEFIT calls
> chi-sq, Stata calls Residual MSE (mean sq. error). And it looks like
> LINFIT gives the Std. Errors of the coefficients that I would use if
> I were you. The LINFIT ch-sq is what Stata calls the Residual SSE (sum
> sq. error).
>
> I'm not sure how the CURVEFIT sigma values are calculated but I would not

> use them if I were you, without knowing exactly where they come from.

Hi Andrew

No, you're right I can't use these CURVEFIT sigma values as they come, but I think I've found the culprit. Just for the sake of it, I also included LMFIT in the test above, which returned exactly the same numbers as CURVEFIT. Now, I'm very inclined to think that I just don't understand the numbers rather than that the procedures are doing something spurious.

A closer inspection reveals something patently absurd with the CURVEFIT and LMFIT sigmas: they don't vary with the noise in the data, i.e., they are always the same. This means that they must have been normalized to the current chi-square (whoever would come up with such an idea ought to be flogged in public). Futhermore, they sometimes are and sometimes are not divided by the number of degrees of freedom (as, indeed, the final value should be). So comparing the three procedures LINFIT, CURVEFIT and LMFIT, I surmise the following transformations that need to be applied to have them return the same numbers (which hopefully are the "right" ones too):

LINFIT : chisq = chisq_0/nfree

CURVEFIT : sigma = sigma_0*sqrt(chisq_0)

LMFIT : chisq = chisq_0/nfree
sigma = sigma_0*sqrt(chisq)

where chisq_0 and sigma_0 denote the values returned by the procedure, and nfree is the number of degrees of freedom, nfree = n_elements(data)-n_elements(parameters).

One would think that they could bloody well inform you of these seemingly arbitrary variations in normalization in the preamble of the online help descriptions..

cheers
ralf

Subject: Re: CURVEFIT.PRO standard deviations?
Posted by Ralf Flicker on Mon, 13 May 2002 09:12:25 GMT
View Forum Message <> Reply to Message

Ivan Valtchanov wrote:
>
> Hi there,

>
> Well, this is a known issue because in "What's new in IDL 5.4" there are
> some lines about LMFIT with correct estimation of sigmas:
>
> ..."In IDL versions prior to 5.4 the correction [sqrt(chisq/(n-m)),
> where n is the number of points and m is the number of coefficients] was
> not being applied."
>
> So you have to multiply the sigmas by this correction factor and you'll
> get them right.
>
> I suppose this is the case for CURVEFIT too.
>
> Hope this helps?

Yes, you're absolutely right - I was too engrossed to see your
response until after I'd posted my own..

thanks,
ralf

Subject: Re: CURVEFIT.PRO standard deviations?
Posted by Craig Markwardt on Mon, 13 May 2002 16:19:30 GMT
View Forum Message <> Reply to Message

Ralf Flicker <rflicker@gemini.edu> writes:

> Andrew Noymer wrote:
>> LINFIT parameters, sigma, and chi-square :
>> -13.7844 2.91336
>> 1.32243 0.0944590
>> 266.783
>> CURVEFIT parameters, sigma, and chi-square :
>> -13.7839 2.91333
>> 0.388221 0.0277362
>> 11.5993
>>

Here is what MPFITFUN / MPCURVEFIT produces:

MPCURVEFIT parameters, sigma, and chi-square :
-14.5118 2.95626
0.388305 0.0277388
200.777

> A closer inspection reveals something patently absurd with the
> CURVEFIT and LMFIT sigmas: they don't vary with the noise in the

> data, i.e., they are always the same. This means that they must have
> been normalized to the current chi-square (whoever would come up
> with such an idea ought to be flogged in public). Futhermore, they
> sometimes are and sometimes are not divided by the number of degrees
> of freedom (as, indeed, the final value should be). So comparing the
> three procedures LINFIT, CURVEFIT and LMFIT, I surmise the following
> transformations that need to be applied to have them return the same
> numbers (which hopefully are the "right" ones too):

Ralph, these are not necessarily patently absurd. The "sigma" values of a fitting program typically mean to find the confidence limits on a parameter based on a change in the chi-squared of +1.

Since you left your fit unweighted, i.e., WEIGHTS=1, the resulting chi-squared value is too high. You have weighted your data so highly that the fit parameters are hypersensitive to any fluctuations in the data. Hence the confidence region is artificially too small. What you need to do is decrease the weights, corresponding to increasing the individual uncertainties, until you achieve a reduced chi-squared of order unity. Then the parameter errors reported by CURVEFIT or MPFITFUN / MPCURVEFIT will be appropriate. This is more or less described in the MPFIT.PRO documentation for the PERROR keyword.

Good luck,
Craig

MPFIT family of fitting functions found at:
http://cow.physics.wisc.edu/~craigm/idl/idl.html (under fitting)

Subject: Re: CURVEFIT.PRO standard deviations?
Posted by thompson on Mon, 13 May 2002 17:46:30 GMT

View Forum Message <> Reply to Message

I tested this with one of my own programs, under IDL/v5.4, and got the following:

LINFIT parameters, sigma, and chi-square :
-7.67345 2.36052
11.8136 0.843827
21290.1
CURVEFIT parameters, sigma, and chi-square :
-7.67969 2.36077
0.388235 0.0277367
925.657
LSTSQR parameters, sigma, and chi-square (w/o):
-7.6734443 2.3605204
11.813374 0.84381246

925.65655
LSTSQR parameters, sigma, and chi-square (with):
-7.6734443 2.3605204
0.38828358 0.027734542
925.65655

My chi-squared values agree with those from CURVEFIT. I can replicate either the LINFIT or CURVEFIT errors depending on how I define the weights to be applied to the data. The two cases are described below

Case 1: No errors passed

In the LINFIT program, as in my own, the errors are passed in through an optional keyword. (This is my "w/o" case.) For LINFIT, this is done through the keyword MEASURE_ERRORS. When this keyword is omitted, the LINFIT program tries to estimate the errors in the data based on the reduce chi-squared values. This behavior in LINFIT is clearly discussed in the documentation:

; Note: if MEASURE_ERRORS is omitted, then you are assuming that the
; linear fit is the correct model. In this case,
; SIGMA is multiplied by SQRT(CHISQ/(N-M)), where N is the
; number of points in X. See section 15.2 of Numerical Recipes
; in C (Second Edition) for details.

Apparently it manages to do this correctly, even though it reports a completely bogus chi-squared. I believe that someone mentioned that the chi-squared problem is fixed in v5.5?

Case 2: Error bars passed.

In the CURVEFIT program, the error bars are passed in through the WEIGHTS parameter in the calling sequence. There doesn't seem to be any way to make this optional. When you put in a WEIGHTS array of all ones, you're saying that the error in each point is +/- 1.0. If you put in more realistic weights, you'd get a more realistic error and chi-squared. With realistic weights, you should get a chi-squared of about one.

I was able to replicate your CURVEFIT results in my own program by passing in an array of error bars of unity. The same happens with LINFIT (except for chi-squared), when the MEASURE_ERROR is passed with all ones.

In short, LINFIT returned the correct SIGMAA values, but the wrong CHISQR. CURVEFIT would have returned the correct SIGMAA and CHISQR values if an appropriate WEIGHTS array had been passed.

William Thompson

P.S. Did you really intend to define your errors the way you
did? Your errors vary with position, and go to exactly 0 at X=5.
I would have expected something like

```
 f0 = (findgen(25)*3.-15) + randomn(seed,25)*noise
```

instead of

```
 f0 = (findgen(25)*3.-15)*(1.+randomn(seed,25)*noise)
```

Ralf Flicker <rflicker@gemini.edu> writes:

> Folks, I'm at wits' end here, don't know what's going on.

> I need to do a chi-square minimization curve fitting of a nonlinear
> parametrized function to noisy data. Using the routine CURVEFIT.PRO
> works admirably for the fitting itself, and the chi-square comes out
> ok, but the values returned for the standard deviations (sigma
> optional argument) on the fitted coefficients just don't make sense
> to me.

> Looking at the source code (idl 5.3) and comparing with the
> Levenberg-Marquardt algorithm in Numerical Recipes (ch 15.5), I am
> still unable to pinpoint the error (whether in my code or in my
> admittedly lacking understanding of chi-square statistics..).

> So here's a code I wrote (FTEST.PRO) for testing CURVEFIT versus a
> routine I know works, LINFIT. It generates a noisy straight line,
> fits a line using both LINFIT and CURVEFIT, and plots both fits
> together with the fits offset by one sigma on the coefficients.

> You can see the discrepancy in the one-sigma lines: can someone tell
> me what's up with the sigma returned from CURVEFIT, and how I can
> make them conform?

> ralf

```
> ;============================================
> ; straight line
> pro fline,x,a,f,pder
```

```
> f = a(0)+x*a(1)
> end

> ;=============================================
> ; Test of CURVEFIT vs. LINFIT
> pro ftest,noise

> ; sample calling sequence : ftest,0.4

>  ; generate noisy line
> f0 = (findgen(25)*3.-15)*(1.+randomn(seed,25)*noise)
> x = findgen(25)
> loadct,4
> plot,x,f0,xstyle=2,psym=-1

>  ; straight-line chi-square fitting with LINFIT.PRO
> a = linfit(x,f0,chisq=csq,sigma=sig)
> fline,x,a,f & oplot,x,f,color=80
> fline,x,[a+sig],f & oplot,x,f,linestyle=2,color=80
> fline,x,[a-sig],f & oplot,x,f,linestyle=2,color=80
> print,'LINFIT parameters, sigma, and chi-square : '
> print,a,sig,csq

>  ; Levenberg-Marquardt chi-square fitting
>  ;of straight line with CURVEFIT.PRO
> a = [0.,1.]
> weights = fltarr(n_elements(x))+1.
>  f=curvefit(x,f0,weights,a,sig,function_name='fline',chisq=cs q,/noderivative)
> oplot,x,f,color=200
> oplot,x,a(0)+sig(0)+x*(a(1)+sig(1)),linestyle=2,color=200
> oplot,x,a(0)-sig(0)+x*(a(1)-sig(1)),linestyle=2,color=200
> print,'CURVEFIT parameters, sigma, and chi-square : '
> print,a,sig,csq

> end
> ;=============================================

> (recombine as necessary)
```

Subject: Re: CURVEFIT.PRO standard deviations?
Posted by Ralf Flicker on Mon, 13 May 2002 22:12:00 GMT
View Forum Message <> Reply to Message

William Thompson wrote:
>
[....]
> In the CURVEFIT program, the error bars are passed in through the

> WEIGHTS parameter in the calling sequence. There doesn't seem to be any
> way to make this optional. When you put in a WEIGHTS array of all ones,
> you're saying that the error in each point is +/- 1.0. If you put in
> more realistic weights, you'd get a more realistic error and
> chi-squared. With realistic weights, you should get a chi-squared of
> about one.

Yes, as Craig pointed out, I had overlooked the importance of the
weights and failed to set them to ~1/variance(data). Unfortunately,
in trying to find out what was wrong, I was confused further by
three fitting routines giving different answers for the same input.

Thanks y'all for your input, I'm just no statistics whiz kid :)

ralf
