

1. Method:

Program uses Nelder Mead method for fitting a user defined model to a given set of data by minimizing the sum of the squared deviations (see [Nelder–Mead method - Wikipedia](#))

2. Parameters:

Input:

zs0: Initial step width for creating simplex, e.g. 0.5

zeps: Termination criterion, e.g. 0.1E-4. If step width smaller than zeps, program stops.

itm: Maximum number of iterations, e.g. 200. If itm is reached, program stops, even if step width larger than zeps

Data in data table dlr

Starting values of parameters for optimization lp

Output:

Number of iterations (if equal to itm minimum was not yet reached and you can start a new computation by

Parameter CHI^2 (sum of squared deviations of model of y and real y, this was the value that was to minimize) and R^2 (coefficient of determination, i.e. share of explained variance of y), list of parameters lp and list of estimated values of dependent variable y, lym.

3. Application and example:

1st step: Write the function cfitfu

Function takes parameter in list lp and dependent x values in list lx and returns estimated dependent variable y for one data point.

Example:

We have the logistic model $y = 1 / (1 + e^{-(a + b \cdot x_1 + c \cdot x_2)})$

Then a, b and c are the parameters lp[1], lp[2] and lp[3], x1 and x2 are the x-values lx[1] and lx[2]

Then the function looks like this:

```
cfitfu()
```

```
Func
```

```
Return 1./(1+e^(-(lp[1]+lp[2]*lx[1]+lp[3]*lx[2])))
```

```
EndFunc
```

2nd step: Create data set dlr. Save values of the m independent variables x(i) in columns 1 to m, the values of the dependent variable in column m+1.

Example (source: Backhaus et.al.: Multivariate Analysemethoden, 14th edition, Springer, p. 288):

You want to explain the purchase (1=yes, 0=no) of an item by the income and gender of the person.

You have 30 occasions. Then the matrix can look like this (column 1: Monthly income in Euro, column 2: Gender (0=female, 1=male), column 3: Purchase (yes=1, no=0):

2,53	0	1
2,37	1	0
2,72	1	1
2,54	0	0
3,2	1	1
2,94	0	1

Iter: Number of iteration, here: 2

fmin: Minimum of function in simplex reached so far, here 7.500000

fmax: Maximum of function in simplex reached so far, here 8.193914

zak: Step width for simplex, here 0.456435

6th step: Interpret results

After about 25 minutes the following information is displayed:

```

  F5  F6  F7  F8  F9  F10  F11  F12  F13  F14  F15  F16  F17  F18  F19  F20  F21  F22  F23  F24  F25  F26  F27  F28  F29  F30  F31  F32  F33  F34  F35  F36  F37  F38  F39  F40  F41  F42  F43  F44  F45  F46  F47  F48  F49  F50  F51  F52  F53  F54  F55  F56  F57  F58  F59  F60  F61  F62  F63  F64  F65  F66  F67  F68  F69  F70  F71  F72  F73  F74  F75  F76  F77  F78  F79  F80  F81  F82  F83  F84  F85  F86  F87  F88  F89  F90  F91  F92  F93  F94  F95  F96  F97  F98  F99  F100
--Iter,fdiff,CHI^2,R^2:
118
3.000000E-13
4.623398
.380795
Par:lp CHI2:zfn Vest:lym
WELDMERAC RAD AUTO FUNC
```

Interpretation:

Process stopped after 118 iterations. Difference between greatest and smallest function value in simplex was $3 \cdot 10^{-13}$, so the process converged.

Value of sum of squared deviations between y and estimated y is 4.623398 which corresponds with an R^2 of 0.380795 which means that 38.1% of the variance of the y-variable can be explained by the model.

Parameters can be seen in list lp. If you type lp ENTER in home screen you get:

{-93.214592 36.771022 33.466466}.

These are the estimated values for a, b and c.

So our model is:

$$y = 1/(e^{(-93.214592+36.771022 \cdot x_1+33.466466 \cdot x_2)})$$

Estimated Y can be seen in list lym. This gives:

```

0.454152
1.000000
1.000000
0.545822
1.000000
1.000000
1.000000
1.000000
1.000000
1.000000
0.002312
1.000000
0.999972
2.358604 E-7
0.999470
0.999981
4.130621 E-9
4.130621 E-9
1.000000
```

3.817855 E-12
 1.632903 E-7
 0.990060
 0.002323
 1.691354 E-15
 0.117365
 2.486372 E-19
 0.00370
 4.278479 E-17
 0.990060
 0.883623
 2.443032 E-15

4. Program listing:

```

cfit(zs0,zeps,itm)
Prgm
Local i1,i2,i3,j1,j2,zfmin,zfmax,zd0,imin,imax,msi,lfw,lrc,zak
0→i3
Data>Mat dlr,md
colDim(md)-1→zm
rowDim(md)→zn
newList(zn)→lym
newList(zm)→lx
Define cfitqs()=Prgm
Local i1,i2
0→zfn
For i1,1,zn
  For i2,1,zm
    md[i1,i2]→lx[i2]
  EndFor
  cfitfu()*1.→lym[i1]
  zfn+(md[i1,zm+1]-lym[i1])^2→zfn
EndFor
EndPrgm
0→zr2
0→zfn
For i1,1,zn
  zr2+md[i1,zm+1]→zr2
  zfn+md[i1,zm+1]^2→zfn
EndFor
(zfn-zr2^2/zn)/(zn-1.)→zr2
dim(lp)→zd0
zs0*1.→zs0
newMat(zd0,zd0+1)→msi
newList(zd0+1)→lfw
0→j1 © Simplex
For i1,1,zd0
  j1+i1→j1
  1.→j2
  For i2,i1+1,zd0+1
    ``(j1)/i1/j2*zs0→msi[i1,i2]
  
```

```

i1+1→j2
EndFor
For i2,1,zd0+1
  msi[i1,i2]+lp[i1]→msi[i1,i2]
EndFor
EndFor
Lbl sfv © New simplex f(x)
For i1,1,zd0+1
  For i2,1,zd0
    msi[i2,i1]→lp[i2]
  EndFor
  cfitqs()
  zfn→lfw[i1]
EndFor
Lbl fvas © Max and min f(x)
1.E99→zfmin
-1.E99→zfmax
For i1,1,zd0+1
  If lfw[i1]<zfmin Then
    i1→imin
    lfw[imin]→zfmin
  EndIf
  If lfw[i1]>zfmax Then
    i1→imax
    lfw[imax]→zfmax
  EndIf
EndFor
Mat>list(sum(msiT))→lrc © refl.c+new p
For i1,1,zd0
  (lrc[i1]-msi[i1,imax])/zd0→lrc[i1]
  2.*lrc[i1]-msi[i1,imax]→lp[i1]
EndFor
cfitqs()
If zfn≥zfmax Then © Case new f > fmax
  For i1,1,zd0
    1.5*lrc[i1]-.5*msi[i1,imax]→lp[i1]
  EndFor
  cfitqs()
  If zfn≥zfmax Then
    For i1,1,zd0
      For i2,1,zd0+1
        (msi[i1,i2]+msi[i1,imin]).5→msi[i1,i2]
      EndFor
    EndFor
    Goto sfv
  EndIf
EndIf
If zfn<zfmin Then © Case f<fmin
  zfn→zfmin
  For i1,1,zd0
    3.*lrc[i1]-2.*msi[i1,imax]→lp[i1]
  EndFor
  cfitqs()
  If zfn>zfmin Then
    For i1,1,zd0

```

```

2.*lrc[i1]-msi[i1,imax]→lp[i1]
EndFor
zfmin→zfn
Else
zfn→zfmin
EndIf
EndIf
If zfn<zfmax Then
For i1,1,zd0
lp[i1]→msi[i1,imax]
EndFor
zfn→lfw[imax]
EndIf
0.→zak
For i1,1,zd0+1
For i2,1,zd0
zak+(msi[i2,i1]-msi[i2,imin])^2→zak
EndFor
EndFor
√(zak/zd0)→zak
i3+1→i3
Disp "Iter - fmin - fmax - zak -"
Disp i3,zfmin,zfmax,zak
If zak<zeps Then
Goto fin
EndIf
If i3>=itm Then
Goto fin
EndIf
Goto fvas
Lbl fin
For i1,1,zd0
msi[i1,imin]→lp[i1]
EndFor
cfits()
1.-zfn/(zn-1)/zr2→zr2
Disp "--Iter,fdiff,CHI^2,R^2: "
Disp i3,zfmax-zfmin,zfn,zr2
Disp "Par:lp CHI2:zfn Yest:lym"
EndPrgm

cfitfu()
Func
Return 1./(1+e^(-(lp[1]+lp[2]*lx[1]+lp[3]*lx[2])))
EndFunc

```