Beta-Regression with SPSS

Michael Smithson
School of Psychology, The Australian National University

(email: Michael.Smithson@anu.edu.au)

SPSS Nonlinear Regression syntax components

The beta-regression models can be run in SPSS under its Nonlinear Regression (NLR and CNLR) procedure. This can be done either via the GUI or syntax, but we shall restrict our attention to the syntax approach here. SPSS requires three main components for these models:

1. A formula for the location and dispersion submodels,
2. A formula for the negative log-likelihood kernel, which is the loss-function to be minimized, and
3. Names and starting-values for the model parameters.

Formula for the location and dispersion submodels

In keeping with our notation, we will denote the location submodel parameters by Bj, where j is an index number starting with 0 for the intercept. The location submodel formula takes the form

COMPUTE PRED_ = EXP(#MU)/(1+EXP(#MU))

where #MU is a linear combination of predictors,

```
COMPUTE       #MU = B0 + … .
```

For example, in a model that predicts anxiety (ANX01) by stress (STR01) in the location submodel, the location submodel formula is defined by

```
COMPUTE       #MU = B0 + B1*STR01 .
```

Likewise, we denote the dispersion submodel parameters by Dk, where k is also an index number. The dispersion submodel formula takes the form

```
COMPUTE PHI = EXP(-D0 - … ) .
```

For example, in a model that has stress (STR01) in the dispersion submodel, the dispersion submodel formula is

```
COMPUTE PHI = EXP(-D0 - D1*STR01) .
```

Dependent variable

The '#DV' is the dependent variable, so in our example we would use a compute statement

COMPUTE #DV = ANX01 .

to assign ANX01 as our dependent variable.

Model parameters and starting-values

The easiest way to generate starting-values for the location and dispersion model coefficients is to begin with a null model (i.e., a model that just has B0 and D0), and then add coefficients one variable at a time in each model. The procedure for this is described below.

To start with, obtain the mean and variance of the dependent variable. In the ANALYZE menu under the DESCRIPTTIVE STATISTICS item choose the DESCRIPTIVES option. The DESCRIPTIVES dialog box appears. Click the Options… button and ensure that the Mean and Variance check-boxes have ticks in them. In our example, the resulting syntax should look like this:

```
DESCRIPTIVES
  VARIABLES=ANX01
  /STATISTICS=MEAN VARIANCE .
```

The output should look like this:

Descriptive Statistics

	N	Mean	Variance
anx01	166	.09120	.018
Valid N (listwise)	166		

To obtain a more precise estimate of the variance, double-click on the table and then double-click inside the cell containing ".018." In the example here we will use the more precise value of .01756.

Now given the mean μ and variance σ^2, we obtain our starting-value for B0 and D0 as follows:

$B0 = \ln[\mu/(1 - \mu)]$

and

$D0 = -\ln[(\mu - \mu^2 - \sigma^2)/\sigma^2]$.

In our example these formulas yield the following results:

$B0 = \ln[.09120/(1 - .09120)] = -2.29907$

$D0 = -\ln[(.09120 - (.09120)^2 - .01756)/.01756] = -1.31371$.

Once the null model has been estimated then the coefficients for that model are used as starting-values for a model that includes one Bj or Dk. The starting-values for the additional Bj or Dk can be set to arbitrary values near 0 (e.g., 0.1). Likewise, when the new model has been estimated then its coefficients become the starting-values for the next model. This process is illustrated in the example provided below.

Syntax Shell with instruction comments

The commented syntax shown below can be pasted into an SPSS Syntax window, modified according to the instructions in the comments, and then selected and run.

```
* First, provide Bj and Dk parameter starting-values.
MODEL PROGRAM B0 =   D0 =  .
* Next, type in your dependent variable name after #DV = in the following COMPUTE statment.
COMPUTE       #DV =  .
* Then you must type in the expression for the location submodel where B0 + occurs.
COMPUTE       #MU = B0 + .
* Then you must provide the expression for the dispersion submodel where -D0 - occurs .
COMPUTE PHI = EXP(-D0 - ) .
COMPUTE PRED_ = EXP(#MU)/(1+EXP(#MU)).
COMPUTE RESID_ = #DV  - PRED_ .
COMPUTE LL = LNGAMMA(PHI) - LNGAMMA(PRED_*PHI) - LNGAMMA(MAX(0.001,PHI -
PRED_*PHI)) + PRED_*PHI*LN(#DV) + (PHI - PRED_*PHI)*LN(1 - #DV) - LN(#DV) - LN(1 - #DV) .
COMPUTE LOSS_  = -LL .
* Finally, substitute your dependent variable for DV in the CNLR command on the next line .
CNLR DV
* This subcommand creates a temporary data-file named SPSSFNLR.TMP .
  /OUTFILE='C:\SPSSFNLR.TMP'
  /PRED PRED_
  /LOSS LOSS_
  /CRITERIA STEPLIMIT 2 ISTEP 1E+20 .
```

Example 2 from Smithson & Verkuilen (2005): Anxiety predicted by Stress

Let μ = Anxiety, $x_0 = w_0 = 1$, and $x_1 = w_1$ = Stress. We will end up fitting the following model:

$\mu_i = \exp(\beta_0 + \beta_1 x_{1i})/[1 + \exp(\beta_0 + \beta_1 x_{1i})]$, and

$\phi_i = \exp(-\delta_0 - \delta_1 w_{1i})$.

First, however, we fit the null model. Open the SPSS file called **Example2.sav**.

Null Model

We start by computing the null model for our example, using the syntax shell with the appropriate substitutions for parameter values, submodel formulae, and dependent variable name. The starting-values for B0 and D0 are the ones derived earlier from the formulas using the sample mean and variance.

```
MODEL PROGRAM B0 = -2.29907  D0 = -1.31371 .
COMPUTE       #DV = ANX01 .
COMPUTE       #MU = B0  .
COMPUTE PHI = EXP(-D0) .
COMPUTE PRED_ = EXP(#MU)/(1+EXP(#MU)).
COMPUTE RESID_ = #DV  - PRED_ .
COMPUTE LL = LNGAMMA(PHI) - LNGAMMA(PRED_*PHI) - LNGAMMA(MAX(0.001,PHI -
PRED_*PHI)) + PRED_*PHI*LN(#DV) + (PHI - PRED_*PHI)*LN(1 - #DV) - LN(#DV) - LN(1 - #DV) .
COMPUTE LOSS_  = -LL .
CNLR ANX01
  /OUTFILE='C:\SPSSFNLR.TMP'
  /PRED PRED_
  /LOSS LOSS_
  /CRITERIA STEPLIMIT 2 ISTEP 1E+20 .
```

The relevant output shows the iteration history and the final negative log-likelihood and parameter values:

```
  Iteration  Loss funct          B0          D0
     0.1    -224.838337  -2.2990700  -1.3137100
     1.1    -230.237861  -2.2407348  -1.3781099
     2.1    -231.036414  -2.2810088  -1.4321800
     3.1    -238.188573  -2.1985724  -1.8940184
     4.1    -239.408369  -2.2596088  -1.7871627
     5.1    -239.447580  -2.2424128  -1.7920993
     6.1    -239.447965  -2.2446359  -1.7967448
     7.1    -239.448006  -2.2439540  -1.7956404
     8.1    -239.448006  -2.2439602  -1.7956430

Run stopped after 8 major iterations.
Optimal solution found.
```

Now we use the final parameter values for B0 and D0 as starting-values for the next model, which adds ANX01 to the location submodel. The starting-value for B1 is set at 0.1:

```
MODEL PROGRAM B0 = -2.2439602 B1 = 0.1 D0 = -1.7956430 .
COMPUTE       #DV = ANX01 .
COMPUTE       #MU = B0 + B1*STR01  .
COMPUTE PHI = EXP(-D0) .
COMPUTE PRED_ = EXP(#MU)/(1+EXP(#MU)).
COMPUTE RESID_ = #DV  - PRED_ .
COMPUTE LL = LNGAMMA(PHI) - LNGAMMA(PRED_*PHI) - LNGAMMA(MAX(0.001,PHI -
PRED_*PHI)) + PRED_*PHI*LN(#DV) + (PHI - PRED_*PHI)*LN(1 - #DV) - LN(#DV) - LN(1 - #DV) .
```

```
COMPUTE LOSS_  = -LL .
CNLR ANX01
  /OUTFILE='C:\SPSSFNLR.TMP'
  /PRED PRED_
  /LOSS LOSS_
  /CRITERIA STEPLIMIT 2 ISTEP 1E+20 .
```

The output is shown below. Note that the negative log-likelihood is –283.007, very different from –239.448 in the null model. The chi-square change statistic is 2*(283.007 – 239.448) = 87.118 which is large and of course significant ($p < .0001$). The B1 coefficient is large and positive, so higher STR01 (stress) predicts higher ANX01 (anxiety).

Try running the syntax using different starting-values for the coefficients (but not wildly different) to verify that the same solution is reached from a different starting-point.

```
  Iteration  Loss funct          B0          B1          D0
     0.1    -241.166696  -2.2439602  .100000000  -1.7956430
     1.1    -265.852909  -2.7192444  2.35799613  -1.6778442
     2.1    -269.752507  -2.9112636  3.32646857  -2.0726701
     3.1    -278.543648  -3.6617377  4.45901276  -2.5924360
     4.1    -281.219933  -3.4505803  3.37661324  -2.4937531
     5.1    -282.724561  -3.4145734  3.64367677  -2.3647083
     6.1    -282.996365  -3.4845705  3.77892040  -2.4642702
     7.1    -283.006611  -3.4785729  3.74726737  -2.4572817
     8.1    -283.006657  -3.4789959  3.74955600  -2.4573848
     9.1    -283.006657  -3.4790199  3.74959458  -2.4574117

Run stopped after 9 major iterations.
Optimal solution found.
```

Now we compute the final model in our example with STR01 in the dispersion submodel, again using the syntax shell with the starting values from the preceding model and D0 = 0.1.

```
MODEL PROGRAM B0 = -3.4790199 B1 = 3.74959458 D0 = -2.4574117 D1 = 0.1 .
COMPUTE       #DV = ANX01 .
COMPUTE       #MU = B0 + B1*STR01  .
COMPUTE PHI = EXP(-D0 - D1*STR01) .
COMPUTE PRED_ = EXP(#MU)/(1+EXP(#MU)).
COMPUTE RESID_ = #DV  - PRED_ .
COMPUTE LL = LNGAMMA(PHI) - LNGAMMA(PRED_*PHI) - LNGAMMA(MAX(0.001,PHI -
PRED_*PHI)) + PRED_*PHI*LN(#DV) + (PHI - PRED_*PHI)*LN(1 - #DV) - LN(#DV) - LN(1 - #DV) .
COMPUTE LOSS_  = -LL .
CNLR ANX01
  /OUTFILE='C:\SPSSFNLR.TMP'
  /PRED PRED_
  /LOSS LOSS_
  /CRITERIA STEPLIMIT 2 ISTEP 1E+20 .
```

An abbreviation of the relevant output is shown below. Note that the negative log-likelihood (-301.960) is lower than for the preceding model (–283.007), indicating improved fit. For a loss of 1 degree of freedom, the chi-square gain is 2*(301.960 – 283.007) = 37.906 ($p < .0001$), once again a significant improvement.

```
  Iteration  Loss funct          B0          B1          D0          D1
     0.1    -283.913317  -3.4790199  3.74959458  -2.4574117  .100000000
     1.1    -284.607693  -3.3541934  3.77969286  -2.6867715  .617696966
...
     8.1    -301.960080  -4.0239012  4.94243205  -3.9614916  4.27552137
```

```
     9.1    -301.960087  -4.0237352  4.94148628  -3.9608697  4.27341043
    10.1    -301.960087  -4.0237166  4.94137080  -3.9608459  4.27330871

Run stopped after 10 major iterations.
Optimal solution found.
```

This output does not tell us anything about the standard errors for the individual coefficients, but we can obtain that information and more by using subcommands under the NLR procedure. These matters are taken up in the next section.

Extracting More from CNLR

Fortunately, the CNLR procedure can yield more than just the coefficients and log-likelihood, but I recommend ensuring that a solution is found by the procedure before including the extra commands. In this section I will cover two additional features: Saving predicted values, residuals, and gradient values; and obtaining bootstrap standard-error estimates for the coefficients.

Saving computed variables

Predicted values, residuals, gradient values, and loss-function values for all cases can be obtained by inserting the /SAVE subcommand before the /CRITERIA subcommand. For instance,

```
/SAVE PRED RESID DERIVATIVES
```

will save the predicted values, residuals, and gradient values (derivatives) to the working data-file. The saved variables may then be used in the usual diagnostic fashion. For instance, summing the derivatives for the final model in our example shows the gradient is near 0 at the solution for each parameter, thereby supporting the claim that the solution is the true optimum. This is a good diagnostic for checking whether a stable solution has been found.

Descriptive Statistics

	N	Sum
d(Pred)/dB0	166	.00003387
d(Pred)/dB1	166	.00004017
d(Pred)/dD0	166	.00001283
d(Pred)/dD1	166	-.00001391
Valid N (listwise)	166	

Obtaining bootstrap standard-error estimates

SPSS does not compute the Hessian at the solution, so we cannot obtain asymptotic standard-error estimates in the usual way that we can from SAS or R. However, SPSS does provide bootstrap estimates (the default is 100 samples). To obtain bootstrap estimates of the standard errors (and the corresponding confidence intervals and correlation matrix of the estimates), insert the following subcommand before the /CRITERIA subcommand:

```
/BOOTSTRAP = N
```

where N is the number of samples desired. Usually 1000-2000 samples suffice for accurate estimates. This may take some time for your computer to complete. The standard-error estimates for the final model are displayed in the output below, using 2000 bootstrap samples. Two different kinds of “95% confidence intervals” are displayed: Bootstrap intervals using the standard error estimates and intervals based on excluding the bottom and top 2.5% of the bootstrap distribution.

```
Bootstrap statistics based on 2000 samples
                                    95% Conf. Bounds     95% Trimmed Range
  Parameter  Estimate  Std. Error    Lower      Upper      Lower      Upper
  B0        -4.0237166   .1792980 -4.3753471 -3.6720861 -4.4188157 -3.7218632
  B1         4.9413708   .4511306  4.0566355  5.8261061  4.2188839  5.9716899
  D0        -3.9608459   .3654175 -4.6774849 -3.2442069 -4.8130994 -3.3877883
  D1         4.2733087  1.0189308  2.2750311  6.2715863  2.6290400  6.5239144

  Bootstrap Correlation Matrix of the Parameter Estimates
                  B0        B1        D0        D1
  B0          1.0000    -.8558     .9289    -.8049
  B1          -.8558    1.0000    -.7954     .7867
  D0           .9289    -.7954    1.0000    -.9324
  D1          -.8049     .7867    -.9324    1.0000
```

Example 3: Reading accuracy, dyslexia, and IQ

Here is the syntax for obtaining the final model presented in Example 3.

```
MODEL PROGRAM B0 = 1.0 B1 = -1.0 B2 = 0.5  B3 = -0.5  D0 = -3.0 D1 = -1.5 D2 = -1.5 .
COMPUTE       #DV = accur01 .
COMPUTE       #MU = B0 + B1*grpctr + B2*ziq + B3*ziq*grpctr  .
COMPUTE PHI = EXP(-D0 - D1*grpctr - D2*ziq) .
COMPUTE PRED_ = EXP(#MU)/(1+EXP(#MU)).
COMPUTE RESID_ = #DV  - PRED_ .
COMPUTE LL = LNGAMMA(PHI) - LNGAMMA(PRED_*PHI) - LNGAMMA(MAX(0.001,PHI -
PRED_*PHI)) + PRED_*PHI*LN(#DV) + (PHI - PRED_*PHI)*LN(1 - #DV) - LN(#DV) - LN(1 - #DV) .
COMPUTE LOSS_  = -LL .
CNLR accur01
 /OUTFILE='C:\SPSSFNLR.TMP'
 /PRED PRED_
 /LOSS LOSS_
 /CRITERIA STEPLIMIT 2 ISTEP 1E+20 .
```

The output should look like this:

```
  Iteration  Loss funct          B0          B1          B2          B3
                                 D0          D1          D2

     0.1    -41.2358609  1.00000000  -1.0000000  .500000000  -.50000000
                         -3.0000000  -1.5000000  -1.5000000
     1.1    -52.0743131  1.23791181  -.75943844  .569808880  -.44495212
                         -2.9665688  -1.4661218  -1.4862255
     2.1    -55.2306798  1.30728993  -.68056267  .447885470  -.60003396
                         -2.9149854  -1.4120187  -1.4239025
     3.1    -56.5891563  1.13373998  -.63548447  .637559197  -.64134904
                         -2.5362587  -.94484557  -1.0599372
     4.1    -57.2555813  1.07831374  -.58988463  .558921411  -.54261406
                         -2.5359237  -.91474799  -.93493396
     5.1    -58.5511176  1.10258977  -.63179017  .491314488  -.42781287
                         -2.5248567  -.96331019  -.67400183
     6.1    -58.8322128  1.11600628  -.64647562  .487452312  -.41820379
                         -2.5732471  -.94699910  -.63011270
     7.1    -59.6214373  1.25298278  -.79506580  .366629364  -.30140637
                         -2.6814224  -.86339826  -.60020635
     8.1    -62.9206508  1.27386224  -.79527097  .273799035  -.30734374
                         -3.2260366  -1.4586572  -.46850363
     9.1    -64.7004302  1.21095544  -.78469907  .353647761  -.39761024
                         -3.1278638  -1.4385474  -.76381603
    10.1    -65.7172344  1.15768085  -.77688475  .426452157  -.53054839
                         -3.2280838  -1.6039456  -1.1029484
```

```
    11.1    -65.8303224  1.12796417  -.75915424  .454971535  -.54775187
                         -3.2731020  -1.6934661  -1.1153256
    12.1    -65.8808783  1.12579375  -.74537544  .474735787  -.56897001
                         -3.2870316  -1.7057132  -1.1481467
    13.1    -65.8956491  1.12191557  -.73973359  .484150093  -.58317070
                         -3.3039576  -1.7372644  -1.2083426
    14.1    -65.9005228  1.12385920  -.74067037  .482580364  -.57713139
                         -3.3063839  -1.7475568  -1.2210027
    15.1    -65.9020784  1.12417102  -.74254880  .485960125  -.58089051
                         -3.3057333  -1.7461811  -1.2281810
    16.1    -65.9021098  1.12330713  -.74173331  .486181241  -.58111549
                         -3.3043440  -1.7465847  -1.2287694
    17.1    -65.9021114  1.12321673  -.74163702  .486392535  -.58127975
                         -3.3044468  -1.7466197  -1.2291498
    18.1    -65.9021114  1.12321888  -.74164108  .486387024  -.58127429
                         -3.3044537  -1.7465987  -1.2291456

Run stopped after 18 major iterations.
Optimal solution found.
```

Example 1: Reading verdict, conflict, and confidence

Here is the syntax for obtaining the final model presented in Example 1. In this example, the BOOTSTRTAP subcommand has been included in order to get standard errors and confidence intervals.

```
MODEL PROGRAM B0 = 0.88  B1 = -0.01 B2 = 0.15 B3 = 0.19 D0 = -1.1 D1 0.34 D2 = -0.22 D3 =
0.1 .
COMPUTE       #DV = crc99 .
COMPUTE       #MU = B0 + B1*vert + B2*confl + B3*vert*confl .
COMPUTE PHI = EXP(-D0 - D1*vert - D2*confl - D3*vert*confl) .
COMPUTE PRED_ = EXP(#MU)/(1+EXP(#MU)).
COMPUTE RESID_ = #DV  - PRED_ .
COMPUTE LL = LNGAMMA(PHI) - LNGAMMA(PRED_*PHI) - LNGAMMA(MAX(0.001,PHI -
PRED_*PHI)) + PRED_*PHI*LN(#DV) + (PHI - PRED_*PHI)*LN(1 - #DV) - LN(#DV) - LN(1 - #DV) .
COMPUTE LOSS_  = -LL .
CNLR crc99
  /OUTFILE='C:\SPSSFNLR.TMP'
  /PRED PRED_
  /LOSS LOSS_
  /BOOTSTRAP = 2000
  /CRITERIA STEPLIMIT 2 ISTEP 1E+20 .
```

The output should look like this:

```
  Iteration  Loss funct          B0          B1          B2          B3
                                 D0          D1          D2          D3
     0.1     -34.9085769  .880000000  -.01000000  .150000000  .190000000
                         -1.1000000  .340000000  -.22000000  .100000000
     1.1     -37.3837903  .854114160  -.01352682  .141216450  .123295595
                         -1.0750089  .348666272  -.23938747  -.41601123
     2.1     -38.3036594  .901527733  .005383527  .148328767  .239277117
                         -1.0796703  .354097351  -.22098482  -.50233557
     3.1     -38.3620928  .897693928  .004632521  .149942759  .244658617
                         -1.0897280  .349956587  -.22054732  -.50500512
     4.1     -38.4600905  .923765783  .003431096  .182316023  .237652918
                         -1.1126248  .333737311  -.22422807  -.50091418
     5.1     -38.7730514  .975850782  .019573069  .144565851  .242035050
                         -1.1698152  .297687943  -.30820316  -.44569287
     6.1     -39.0006763  .970773319  .032449817  .147301838  .243910485
                         -1.1762844  .301388449  -.30804126  -.43068312
```

```
     7.1    -39.2460674  .964555675  .017172531  .147229870  .248990665
                        -1.1752455  .300065897  -.29469110  -.41427284
     8.1    -39.6043992  .925196640  -.02142601  .191011296  .280511231
                        -1.1774119  .357277873  -.31437756  -.37243096
     9.1    -40.0772449  .906372736  .009075866  .170377549  .286421782
                        -1.1795314  .313600158  -.21457579  -.31217524
    10.1    -40.0804244  .910070346  .007972938  .166769970  .283984116
                        -1.1852429  .320723141  -.21045266  -.31208524
    11.1    -40.0940088  .912302538  .005095002  .168337251  .279139066
                        -1.1715331  .331510669  -.22080541  -.31771004
    12.1    -40.0945123  .912031114  .004976252  .168554233  .280008921
                        -1.1736998  .330041115  -.21956476  -.31622213
    13.1    -40.0945126  .912006925  .005015933  .168521090  .280030961
                        -1.1737012  .330038964  -.21954536  -.31618009

Run stopped after 13 major iterations.
Optimal solution found.

Bootstrap statistics based on 2000 samples
  Loss function value: -40.094513

                                    95% Conf. Bounds     95% Trimmed Range
  Parameter  Estimate  Std. Error    Lower      Upper      Lower      Upper
  B0          .9120069   .1088115   .6986112  1.1254027   .7186656  1.1421216
  B1          .0050159   .1093781  -.2094912   .2195230  -.1903118   .2284498
  B2          .1685211   .1086506  -.0445592   .3816014  -.0572269   .3749880
  B3          .2800310   .1092399   .0657949   .4942670   .0520406   .4756850
  D0        -1.1737012   .1609672 -1.4893824  -.8580201 -1.6283350 -1.0032642
  D1          .3300390   .1630611   .0102515   .6498265  -.0131310   .6563368
  D2         -.2195454   .1638997  -.5409775   .1018867  -.5243025   .1216315
  D3         -.3161801   .1646773  -.6391372   .0067770  -.6240925   .0151662

  Bootstrap Correlation Matrix of the Parameter Estimates
                 B0        B1        B2        B3        D0        D1        D2
  B0          1.0000     .3870    -.2288    -.3361    -.2507    -.3430     .1007
  B1           .3870    1.0000    -.3307    -.2273    -.3654    -.2760     .3493
  B2          -.2288    -.3307    1.0000     .4162     .0893     .3930    -.2759
  B3          -.3361    -.2273     .4162    1.0000     .3551     .1508    -.3360
  D0          -.2507    -.3654     .0893     .3551    1.0000    -.0554    -.1800
  D1          -.3430    -.2760     .3930     .1508    -.0554    1.0000    -.1697
  D2           .1007     .3493    -.2759    -.3360    -.1800    -.1697    1.0000
  D3           .3672     .1274    -.3564    -.2720    -.1715    -.1968    -.0309

                 D3
  B0           .3672
  B1           .1274
  B2          -.3564
  B3          -.2720
  D0          -.1715
  D1          -.1968
  D2          -.0309
  D3          1.0000
```

Reference

Smithson, M. and Verkuilen, J. (2005). A Better Lemon-Squeezer? Maximum Likelihood Regression with Beta-Distributed Dependent Variables.