www.statabook.com

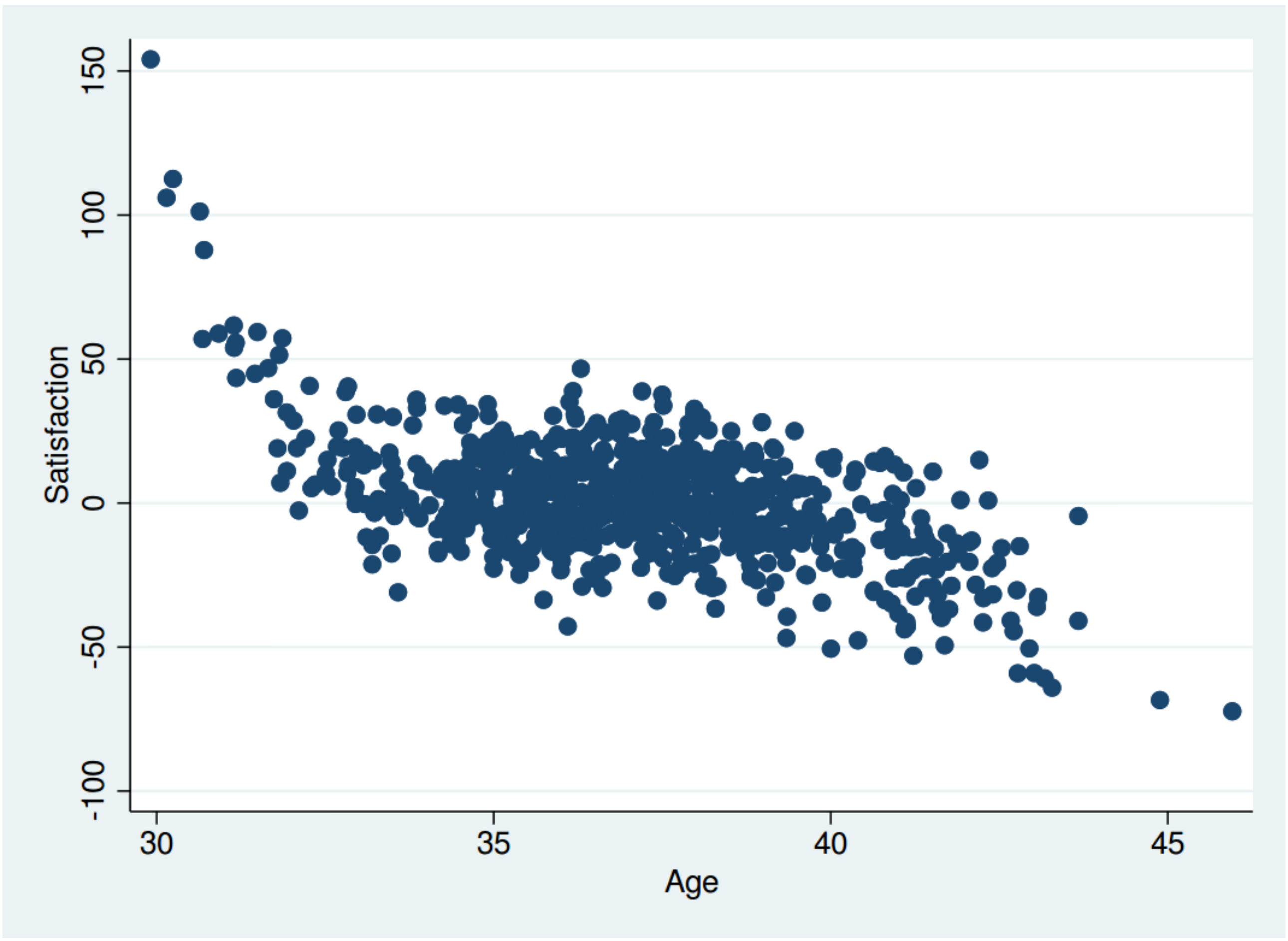
Functional Forms

Version 1.0 (February 2019)

Felix Bittmann (mail@statabook.com)

Finding that two variables are somewhat related is an important first step but often you want to dig deeper. Researching the functional form between two variables can be highly relevant. This is interesting when you assume that a relation between two variables is not linear, therefore, a linear (OLS) regression might run into problems. We will have a simple example to see what can be done. For this reason, I created a dataset with only three variables: ID (the ID of the participant), x (the age of the participant) and y (the life satisfaction of the participant). We are interested between age and satisfaction (if you prefer you can think of any other metric variables as these are made up examples).
Load the dataset (funcform.dta) in Stata and type *describe* to get an overview of the variables. Next, we will use scatterplots to visualize the basic relation. This is a first but important step that is a part of descriptive statistics:

```
use “funcform.dta”, clear
scatter y x
```



Note that the first variable (y) goes to the y-axis, the second one to the x-axis. The relation is interesting. To the left side, there is a steep and negative relation as satisfaction quickly drops with age (30 to 33 years). Then, there is a quite stable part where satisfaction stays relatively constant (between 34 and 40 years). After that, we see that the relation is, again, negative and begins to decrease again. We conclude from this alone that the relationship is not linear. We could fit three straight lines in this scatterplot to describe the partial relations within. However, if we want to use statistical tools like regressions, we have to use other means. As our variable of interest (y) is metric, we will start with regular linear regressions and see how we can improve the model. Our only independent variable is x.

```
regress y c.x
```
[1]

```
      Source |       SS           df       MS      Number of obs   =       750
-------------+----------------------------------   F(1, 748)       =    319.59
       Model |  98846.8928         1  98846.8928   Prob > F        =    0.0000
    Residual |  231350.556       748  309.292187   R-squared       =    0.2994
-------------+----------------------------------   Adj R-squared   =    0.2984
       Total |  330197.449       749  440.851066   Root MSE        =    17.587

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |  -4.410525   .2467137   -17.88   0.000    -4.894859   -3.926191
       _cons |   164.2935   9.181583    17.89   0.000     146.2688    182.3183
------------------------------------------------------------------------------
```

We note that our first result is negative (-4.41), telling us that with every year more, the satisfaction drops by 4.4 points. The effect is highly significant. This seems vaguely OK as we look at the scatterplot as the overall tendency is negative, yet this is very crude and does not capture the overall, more complex relation. Also have a look at R-squared, which tells us that our model can capture only about one-third of the overall variance. Whenever we face such a situation, we can test whether the introduction of higher order terms helps us (polynomial regression). This makes the model more flexible. In the past we would have created these terms manually, for example:

```
generate x_squared = x*x
```

Nowadays we can skip this step as the factor-variable-notation does this automatically. So we type

```
regress y c.x##c.x
```

```
      Source |       SS           df       MS      Number of obs   =       750
-------------+----------------------------------   F(2, 747)       =    161.58
       Model |  99711.7411         2  49855.8706   Prob > F        =    0.0000
    Residual |  230485.707       747  308.548471   R-squared       =    0.3020
-------------+----------------------------------   Adj R-squared   =    0.3001
       Total |  330197.449       749  440.851066   Root MSE        =    17.566

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |  -12.79063   5.011491    -2.55   0.011    -22.62892    -2.95235
             |
     c.x#c.x |   .1124131   .0671442     1.67   0.095    -.0194007    .2442268
             |
       _cons |   319.7083   93.28096     3.43   0.001     136.5843    502.8324
------------------------------------------------------------------------------
```

Here we see that R-squared was slightly improved and both coefficients are significant (the second term is on the edge but OK so far…). Still, this model does not make us happy. We are not surprised as a second term can only capture a U shape or a reversed U shape, but ours looks different. Maybe a third term helps? Either create it manually or type:

```
regress y c.x##c.x##c.x
```

1 Typing a *c.* in front of the independent variable tells Stata that this variable is metric (continuous).

```
      Source |       SS           df       MS      Number of obs   =       750
-------------+----------------------------------   F(3, 746)       =    195.09
       Model |  145166.652         3  48388.8841   Prob > F        =    0.0000
    Residual |  185030.796       746  248.030558   R-squared       =    0.4396
-------------+----------------------------------   Adj R-squared   =    0.4374
       Total |  330197.449       749  440.851066   Root MSE        =    15.749

------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           x |  -848.1967   61.87397   -13.71   0.000    -969.6646   -726.7289
             |
     c.x#c.x |   22.65965   1.666629    13.60   0.000     19.38781    25.93149
             |
 c.x#c.x#c.x |  -.2018579    .014911   -13.54   0.000    -.2311305   -.1725853
             |
       _cons |   10586.73   763.0121    13.87   0.000     9088.825    12084.64
------------------------------------------------------------------------------
```

R-squared is now much higher and suddenly all three terms are highly significant! This is a bit surprising but happens as this model can capture the general relation much better. You could now play around and see if even more terms help. Sometimes this is the case, but in general, three terms are the maximum. Also, keep in mind that often not a *perfect* fit is needed to capture the essence of a relation.

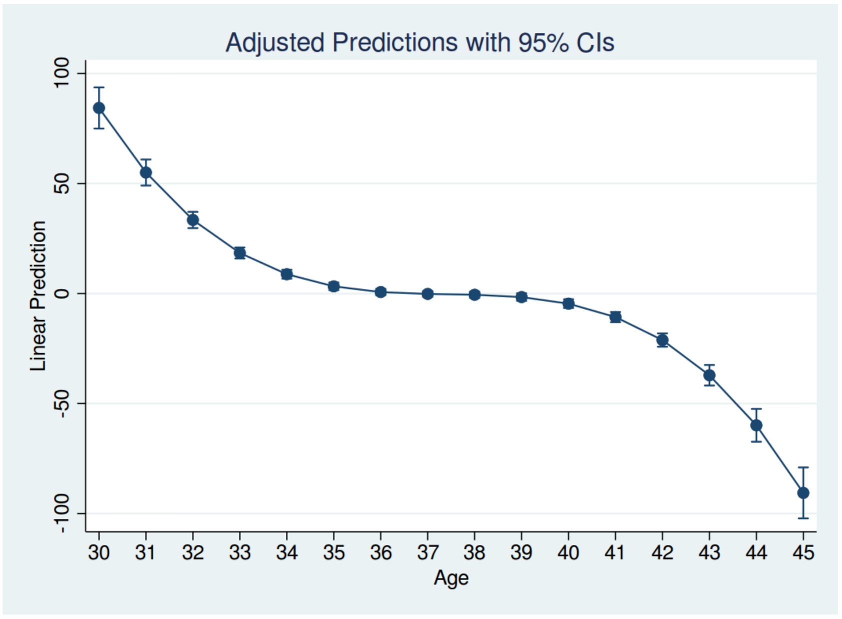
Even our "good" model (or should we say "least bad"?) only consists of numbers, which is good for publication in a table, but quite abstract and will not help you influence policy as not-experts will not understand it. To change this we use Stata's *margin* command which helps us in visualizing. We just specify that we want predicted values for every age point between 30 and 45, which is done by typing

```
margins, at(x=(30(1)45))
```

```
------------------------------------------------------------------------------
             |            Delta-method
             |     Margin   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         _at |
          1  |   84.35122   4.760757    17.72   0.000     75.00514    93.69729
          2  |   55.00775   3.014934    18.25   0.000     49.08898    60.92651
          3  |   33.43801   1.885503    17.73   0.000     29.73649    37.13953
          4  |   18.43086   1.285158    14.34   0.000      15.9079    20.95382
          5  |   8.775142   1.029271     8.53   0.000     6.754529    10.79575
          6  |   3.259714    .888763     3.67   0.000      1.51494    5.004489
          7  |   .6734282   .7671227     0.88   0.380     -.832548    2.179404
          8  |  -.1948636     .70488    -0.28   0.782    -1.578648    1.188921
          9  |  -.5563088   .7587738    -0.73   0.464    -2.045895    .9332772
         10  |  -1.622055    .887403    -1.83   0.068    -3.364159    .1200498
         11  |  -4.603249   1.019256    -4.52   0.000    -6.604199   -2.602298
         12  |  -10.71104   1.176433    -9.10   0.000    -13.02055   -8.401525
         13  |  -21.15657   1.545022   -13.69   0.000    -24.18968   -18.12346
         14  |  -37.15099   2.376063   -15.64   0.000    -41.81556   -32.48643
         15  |  -59.90545   3.804415   -15.75   0.000    -67.37409   -52.43682
         16  |   -90.6311   5.894416   -15.38   0.000    -102.2027   -79.05948
------------------------------------------------------------------------------
```

We receive a long table, which is nice but not good enough. We can use *marginsplot* to create a graph from these numbers.

```
marginsplot
```



This is so much better! The functional form is clearly preserved as the middle part is constant while the outer regions clearly show the steep slopes. However, this line looks somehow “smoothed” as our model does not capture the empirical relationship with all the details. However, this is a very good result that would be fine for publication.

A second option is to use another Stata command which does not rely on your insight but just crunches numbers to get the best fit automatically. This might sound too good, but you only need to put your independent variable(s) in the model and Stata will try to approximate the best fit. Due to statistical methods, no functional forms, higher order terms or even interactions have to be specified, Stata relies on the “big data” and does all the work. This seems really cool, yet has the downside that this can take a *long* time. If you have models with more than only a few variables it can take hours and if you even want confidence intervals in complex models, you better plan with weeks rather than days. Let’s see this in action. The command is *npregress kernel* and was introduced in Stata 15.

```
npregress kernel y x
```

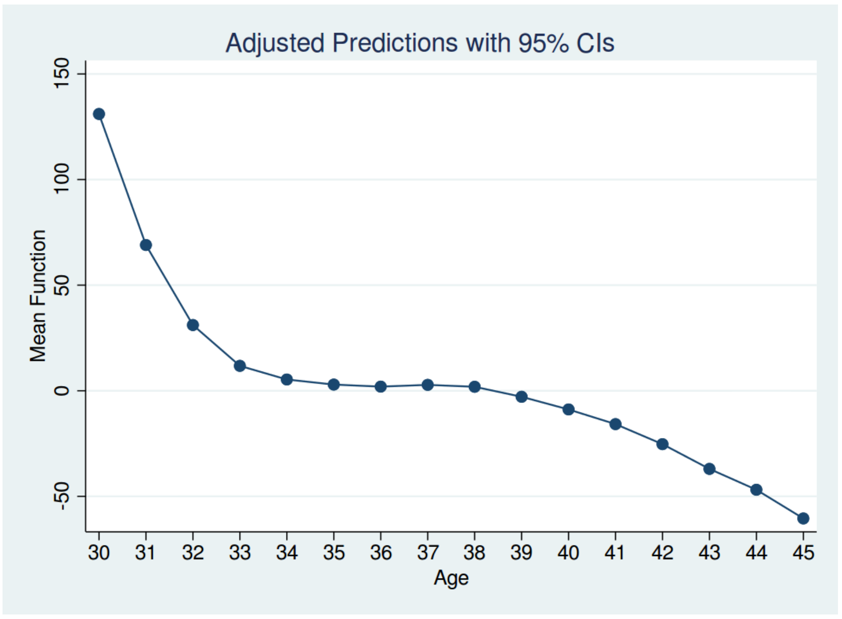
```
Bandwidth
─────────────┬──────────────────────
             │      Mean      Effect
─────────────┼──────────────────────
Mean         │
           x │  .6930077    .9827953
─────────────┴──────────────────────

Local-linear regression                    Number of obs      =         750
Kernel   : epanechnikov                    E(Kernel obs)      =         520
Bandwidth: cross validation                R-squared          =      0.4977
─────────────┬──────────────────────────────────────────────────────────────
           y │   Estimate
─────────────┼──────────────────────────────────────────────────────────────
Mean         │
           y │   .6769911
─────────────┼──────────────────────────────────────────────────────────────
Effect       │
           x │  -4.059561
─────────────┴──────────────────────────────────────────────────────────────
Note: Effect estimates are averages of derivatives.
Note: You may compute standard errors using vce(bootstrap) or reps().
```

The numbers displayed here are not very helpful, however, we see that R-squared is now almost 50 which is clearly better than our first models. We can use exactly the same *margins* command as above to get predicted values (note that we only go to 44 and not 45 as otherwise, we run into problems due to a low number of cases).

```
margins, at(x=(30(1)44))
```

And we use *marginsplot* to have it plotted.

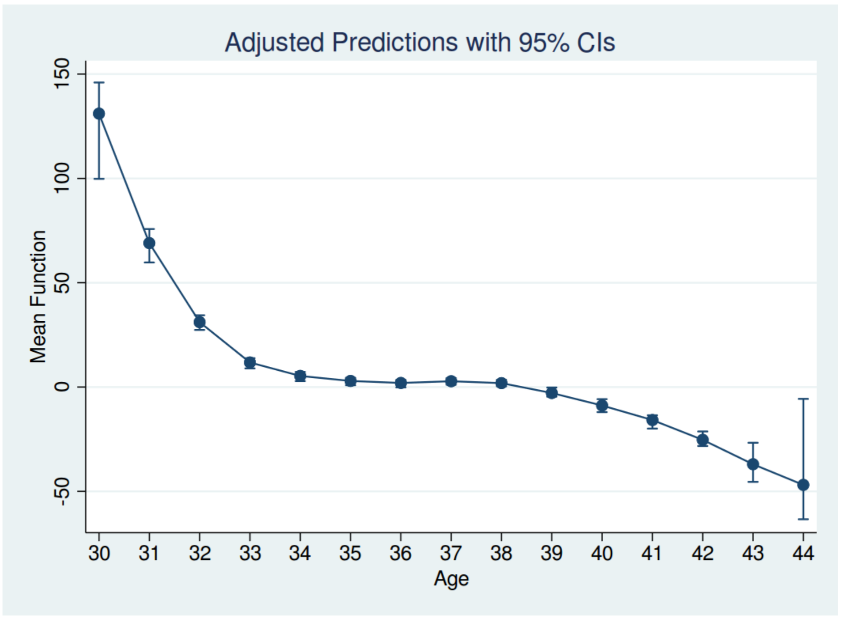
```
marginsplot
```



The result is similar to our first results, however, more complex. The slopes are now quite different and the form is less mathematical perfect as it is given by the actual data. We also note that the confidence intervals are missing, as these must be calculated by bootstrapping. If you want them, type

```
margins, at(x=(30(1)45)) reps(50)
```

This might take five minutes or more. However, now we get the intervals. For a real application, you probably need more replications (try 200 and hope that it is enough).



Also, be careful when you use this to extrapolate into areas without data. As you see from the confidence intervals, at the borders they have become broad quickly. As this method relies on data and not on mathematical functions, results can become total nonsense when you try to make predictions in areas without a sufficient number of cases.

A final word of caution: using a lot of higher order terms can be nice in regular regressions, yet there still exists the problem of **overfitting**. This arises when you try to make your model as flexible as possible so it almost perfectly adapts to the current data. The problem is that the underlying process that generated the data is probably quite different from your function. So, if you get another batch of data and try to use the same function, you probably run into problems. The same goes for predicting values: when you try to predict values, possible outside the range of actual values you see in the data, your predictions might be catastrophic if you have a large overfit. The same is true for *npregress kernel*: having predictions outside the range of actual data can be useless. Therefore, be careful. A technique to prevent this is the following: when you receive your data, randomly sort it and only use 80% of it for creating your model. After being done, apply your generated model to the final 20% of the data that you did not use. If the result is fine, your general model is probably fine as well.