Understanding & Interpreting the Effects of Continuous Variables: The MCP Command

References: "marginscontplot: Plotting the marginal effects of continuous predictors", by Patrick Royston. Stata Journal Volume 13 Number 3: pp. 510-527. http://www.stata-journal.com/article.html?article=gr0056.

Introduction

We have talked a lot about ways to make effects of variables in nonlinear models easier to understand and interpret. However, the primary emphasis has been on categorical independent variables, e.g. race, religion, gender. Some methods that are helpful with categorical independent variables are not so helpful for continuous independent variables, or else need modification. Consider, for example, Average Marginal Effects:

```
. * Set up data
. set more off
. webuse nhanes2f, clear
. keep if !missing(diabetes, black, female, age)
(2 observations deleted)
. label define black 0 "nonBlack" 1 "black"
. label define female 0 "male" 1 "female"
. label values female female
. label values black black
. quietly logit diabetes i.black i.female age c.age#c.age, nolog
. margins, dydx(*)

Average marginal effects                          Number of obs   =      10335
Model VCE    : OIM

Expression   : Pr(diabetes), predict()
dy/dx w.r.t. : 1.black 1.female age

------------------------------------------------------------------------------
             |            Delta-method
             |      dy/dx   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       black |
      black  |   .0404163   .0087456     4.62   0.000     .0232752    .0575574
             |
      female |
     female  |   .0069065   .0041324     1.67   0.095    -.0011928    .0150059
         age |   .0021114   .0002542     8.30   0.000     .0016131    .0026096
------------------------------------------------------------------------------
Note: dy/dx for factor levels is the discrete change from the base level.
```

The AME for black tells us that, on average, blacks are 4 percentage points more likely to have diabetes than are comparable whites. That obscures a lot of individual-level variation (e.g. racial differences vary greatly by age) but still, it gives us a general idea of how great differences are. The AME for age, however, tells us what the instantaneous rate of change for age is. This may or may not be a very good approximation of the effect of a one unit change in age, and in any event it is much less intuitive than the AMEs for the categorical variables are.

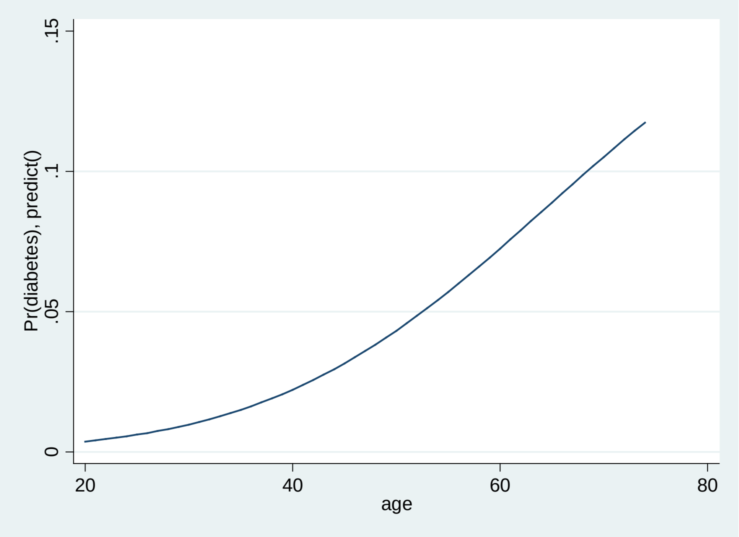
Patrick Royston's `mcp` (aka `marginscontplot`) command, which was introduced in September 2013, tries to address such concerns. The introduction to Royston's article says

> The developers of Stata 11 and 12 have clearly put much effort into creating the margins and marginsplot commands. Their work appears to have been well received by users. However, margins and marginsplot are naturally focused on margins for categorical (factor) variables, and continuous predictors are arguably rather neglected. In this article, I present a new command, marginscontplot, which provides facilities to plot the marginal effect of a continuous predictor in a meaningful way for a wide range of regression models. In principle, it can handle any regression command for which margins is applicable and makes sense. This includes all the familiar commands such as regress, logit, probit, poisson, glm, stcox, streg, and xtreg. marginscontplot is also known as mcp for those who dislike typing the full command name. You may use marginscontplot and mcp interchangeably.

`mcp` probably isn't critical (you could do the same things with the `margins` command) but it greatly simplifies some tasks. I will cover a few highlights, but read the article and the help file to discover other powerful and useful features.

The basic command

```
. logit diabetes i.black i.female age c.age#c.age, nolog

Logistic regression                               Number of obs   =      10335
                                                  LR chi2(4)      =     381.03
                                                  Prob > chi2     =     0.0000
Log likelihood = -1808.5522                       Pseudo R2       =     0.0953

------------------------------------------------------------------------------
    diabetes |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       black |
      black  |   .7207406   .1266509     5.69   0.000     .4725093    .9689718
             |
      female |
     female  |   .1566863   .0942032     1.66   0.096    -.0279486    .3413212
         age |   .1324622   .0291223     4.55   0.000     .0753836    .1895408
             |
 c.age#c.age |  -.0007031   .0002753    -2.55   0.011    -.0012428   -.0001635
             |
       _cons |   -8.14958   .7455986   -10.93   0.000    -9.610926   -6.688233
------------------------------------------------------------------------------

. est store m1
. mcp age, show
margins , at( age=( 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45
46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62
>  63 64 65 66 67 68 69 70 71 72 73 74) )
```



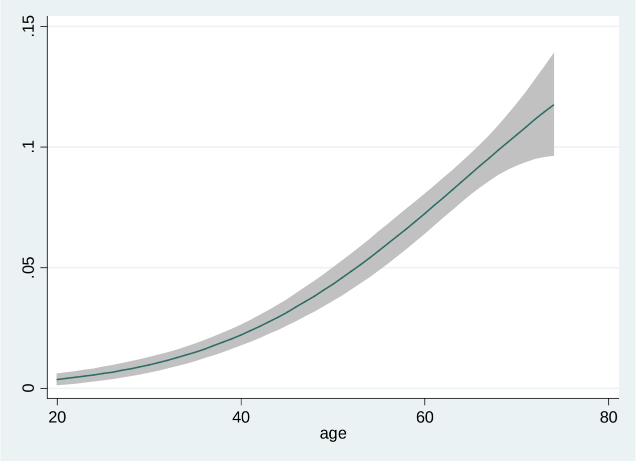
The show option is optional; it tells mcp to display the margins command it generated internally, which can be helpful for making sure you are generating what you want to generate.

Basically, what mcp did was compute the average adjusted prediction (AAP) for each of the observed values of age. Or, if you prefer, it computed the average adjusted prediction (AAP) at representative values of age, i.e. it is computing APRs (adjusted predictions at representative values) for age while all other variables were left at their observed values. If we preferred to get the APMs (adjusted predictions at the means of the other variables in the model) we could give the command

```
. mcp age, show margopts(atmeans)
```

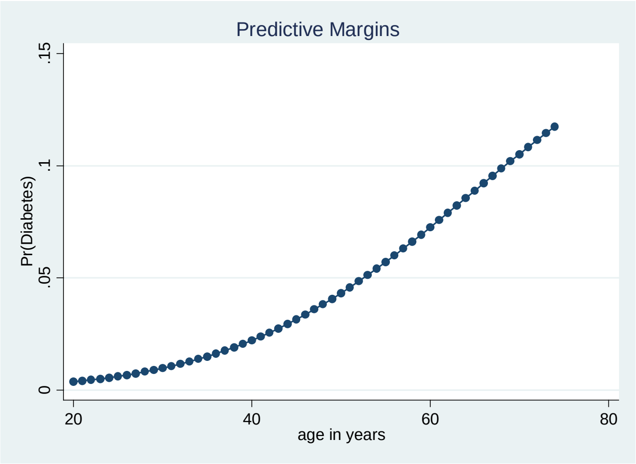
If we want to get the same graph but with the confidence intervals for the predictions,

```
. mcp age, ci
```



We could do pretty much the same thing with `margins` and `marginsplot`:

```
. quietly margins, at(age=(20(1)74))
. marginsplot, noci
```



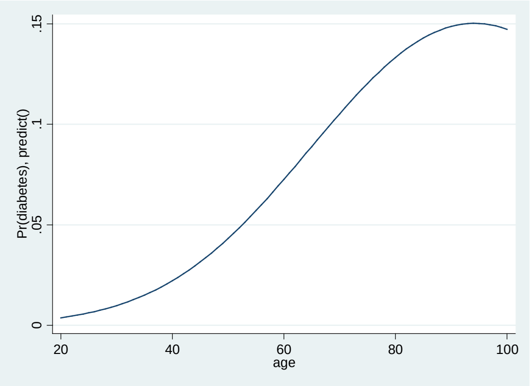
All of the graphs make clear that, at least in this sample, young people have a very low likelihood of getting diabetes. However, the predicted probability of diabetes rises with age and is around 12% when people reach their 70s. This is probably much more informative than the logistic regression coefficients (whose interpretation is further complicated by the fact that the model includes both age and age^2). It is also probably more informative than simply looking at the AMEs. As we saw before, the AMEs tell us that the instantaneous rate of change for age is .0021114. Other than being positive and significant, it is hard to say much about that effect. The average adjusted predictions for different values of age provide a much clearer picture.

The `mcp` command saved us the trouble of figuring out what the `at` option should be and it also internally provided options that made the graph look nicer (which again we could have done ourselves but `mcp` saved us the work). These are relatively minor advantages but still nice. `mcp` has other options that further add to its usefulness.

Controlling the range of plotted values

By default, `mcp` plots AAPs for all of the observed values of the continuous variable. Using the `at1` option, you can control the range yourself, making it either broader or narrower. In the example below, I extend the range of age to go between 20 and 100 rather than 20 and 74. In this particular case, this is probably a bad idea, since 100 is well outside the observed range of the data, but it may make more sense in other situations.

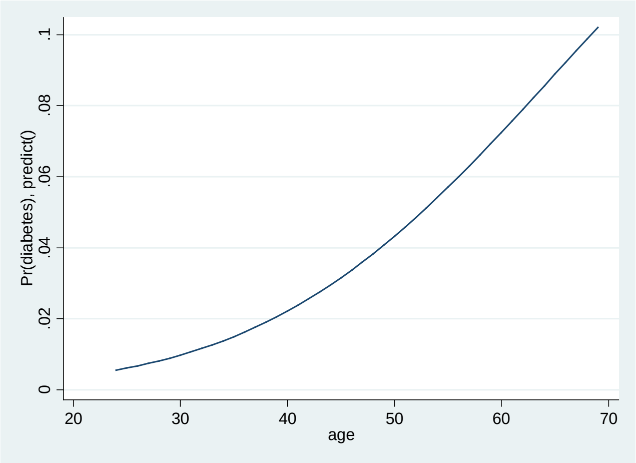
```
. mcp age, at1(20(1)100) show
margins , at( age=( 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45
46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62
>  63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93
94 95 96 97 98 99 100) )
```



Because of the squared term, we know that at some point the predicted effect of age should start declining, and the graph shows that this happens sometime after age 90 (although again I wouldn't trust a prediction that is so far outside the range of the observed data; in a moment we'll see what may be a better way to model the data).

Sometimes we have extreme values at either end of the distribution. These can zap your graphs because the extreme cases force you to extend the axes. As Royston points out, "We can limit the plotting values according to chosen centiles… by using the % prefix in the at1() option." So, for example, suppose I want to exclude the bottom and top 10%. I could do something like the following:

```
. mcp age, at1(%10 (1) 90) show
margins , at( age=( 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66
>  67 68 69) )
```



In this case you see that ages less than 23 and greater than 69 were excluded from the graph. Royston provides a more dramatic example, where 99% of the cases have a value of 998 or less but the highest value is 2,380.

Alternative approaches for specifying the range may also be helpful when there are a large number of unique values for the continuous variable, as `mcp` will try to compute the AAP for each one, which can be very, very slow.

Models with transformed X

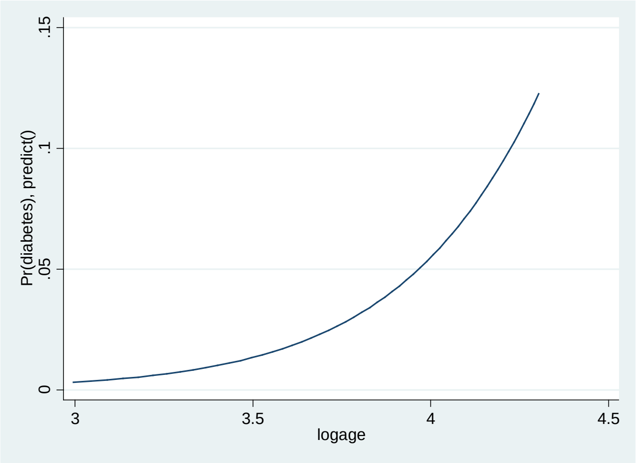
This may be the feature I like most about `mcp`. Sometimes it makes sense to use things like the log of X rather than X in the model. So, we are going to modify our current example to use the natural log of X rather than X and X^2.

```
. generate logage = log(age)
. logit diabetes i.black i.female logage, nolog

Logistic regression                               Number of obs   =      10335
                                                  LR chi2(3)      =     381.88
                                                  Prob > chi2     =     0.0000
Log likelihood = -1808.1268                       Pseudo R2       =     0.0955

------------------------------------------------------------------------------
    diabetes |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       black |
      black  |   .7203993   .1266702     5.69   0.000     .4721303    .9686682
             |
      female |
     female  |   .1559777    .094211     1.66   0.098    -.0286724    .3406278
      logage |   2.907695   .1949347    14.92   0.000      2.52563     3.28976
       _cons |   -14.6621   .7984324   -18.36   0.000      -16.227    -13.0972
------------------------------------------------------------------------------

. est store m2
. lrtest m1 m2, stats

Likelihood-ratio test                                 LR chi2(1)  =     -0.85
(Assumption: m2 nested in m1)                         Prob > chi2 =    1.0000

Akaike's information criterion and Bayesian information criterion

-----------------------------------------------------------------------------
       Model |    Obs    ll(null)   ll(model)     df          AIC         BIC
-------------+---------------------------------------------------------------
          m2 |  10335   -1999.067   -1808.127      4     3624.254    3653.227
          m1 |  10335   -1999.067   -1808.552      5     3627.104    3663.321
-----------------------------------------------------------------------------
               Note:  N=Obs used in calculating BIC; see [R] BIC note
```

As we see, the effect of logage is highly significant. If we viewed this and the model with X^2 as competing theories, both the BIC and AIC tests favor the model with logage.

Plotting the results with `mcp`,

```
. mcp logage
```



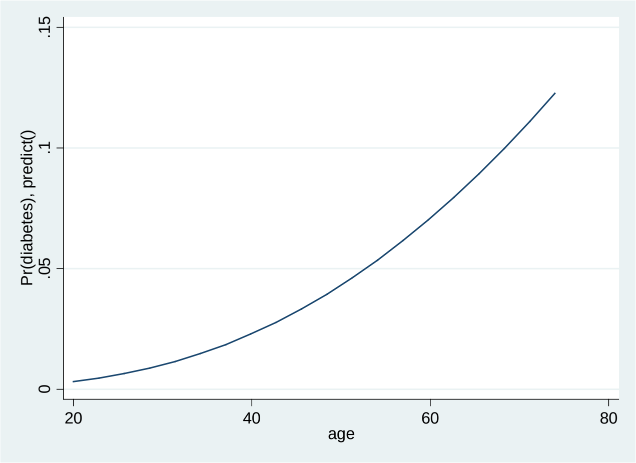
This shows us that, as the log of age goes up, the probability of diabetes increases. However, most of us are not used to thinking in terms of the log of age. We would rather see the values plotted against age. Here is how you can do that:

```
. summarize age

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
         age |      10335    47.56584    17.21752         20         74

. range w1 r(min) r(max) 20
(10315 missing values generated)

. generate logw1 = log(w1)
(10315 missing values generated)

. fre w1 logw1, tab(5)

w1
--------------------------------------------------------------
                 |      Freq.    Percent      Valid       Cum.
-----------------+--------------------------------------------
Valid   20       |          1       0.01       5.00       5.00
        22.84211 |          1       0.01       5.00      10.00
        25.68421 |          1       0.01       5.00      15.00
        28.52632 |          1       0.01       5.00      20.00
        31.36842 |          1       0.01       5.00      25.00
        :        |          :          :          :          :
        62.63158 |          1       0.01       5.00      80.00
        65.47369 |          1       0.01       5.00      85.00
        68.31579 |          1       0.01       5.00      90.00
        71.1579  |          1       0.01       5.00      95.00
        74       |          1       0.01       5.00     100.00
        Total    |         20       0.19     100.00
Missing .        |      10315      99.81
Total            |      10335     100.00
--------------------------------------------------------------
```

```
logw1
-------------------------------------------------------------
                 |      Freq.    Percent      Valid       Cum.
-----------------+-------------------------------------------
Valid   2.995732 |          1       0.01       5.00       5.00
        3.128606 |          1       0.01       5.00      10.00
        3.245876 |          1       0.01       5.00      15.00
        3.350827 |          1       0.01       5.00      20.00
        3.445802 |          1       0.01       5.00      25.00
        :        |          :          :          :          :
        4.137269 |          1       0.01       5.00      80.00
        4.181648 |          1       0.01       5.00      85.00
        4.224141 |          1       0.01       5.00      90.00
        4.264901 |          1       0.01       5.00      95.00
        4.304065 |          1       0.01       5.00     100.00
        Total    |         20       0.19     100.00
Missing .        |      10315      99.81
Total            |      10335     100.00
-------------------------------------------------------------
```

What the `range` command did was divide age up into 20 equally spaced intervals (you can use more intervals or less if you like). logw1 is the natural log of those 20 values. We can now do the following:

```
. mcp age (logage), var1(w1 (logw1)) show
margins , at( logage=( 2.995732307434082 3.128605604171753 3.245876312255859 3.350826978683472
3.445801734924316 3.532533407211304 3.61233925819397
> 3.686244487762451 3.755061388015747 3.819445848464966 3.879934549331665 3.936972379684448
3.990931510925293 4.04212760925293 4.090829849243164 4.1
> 37269496917725 4.181648254394531 4.224141120910645 4.264901161193848 4.304065227508545) )
```



`mcp` plotted the AAPs for each of the 20 values of logage that had been computed. However, by saying

```
. mcp age (logage), var1(w1 (logw1)) show
```

`mcp` knew to show the corresponding value of age for each value of logage. As for the var1 option, the help says

> var1(#|var1_spec) specifies plotting values of xvar1. If var1(#) is specified, then # equally spaced values of xvar1 are used as plotting positions, encompassing the observed range of xvar1. Alternatively, var1_spec may be used to specify transformed plotting values of xvar1. The syntax of var1_spec is var1 [(var1a [var1b ...])]. var1 is a variable holding user-specified plotting values

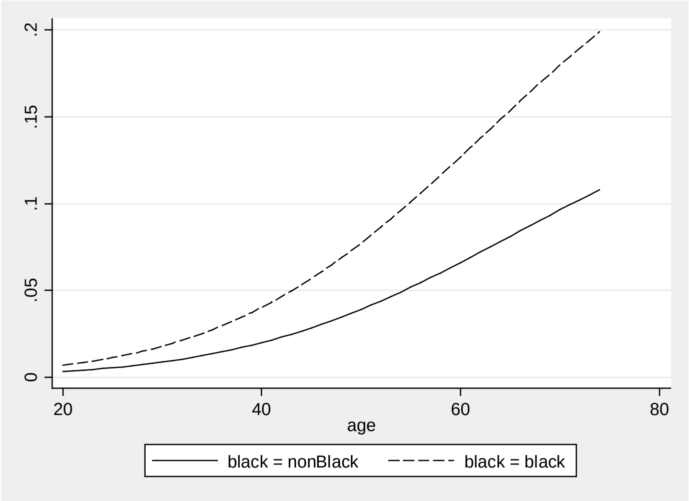
of xvar1. var1a is a variable holding transformed values of var1 and similarly for var1b ... if required.

It may be easier just to use this as a template rather than trying to understand exactly what is being done! But see Royston's article or the help file for `mcp` for more details.

Plotting multiple groups simultaneously

You can specify both a continuous variable followed by a categorical variable if you want to see plots for separate groups, e.g.

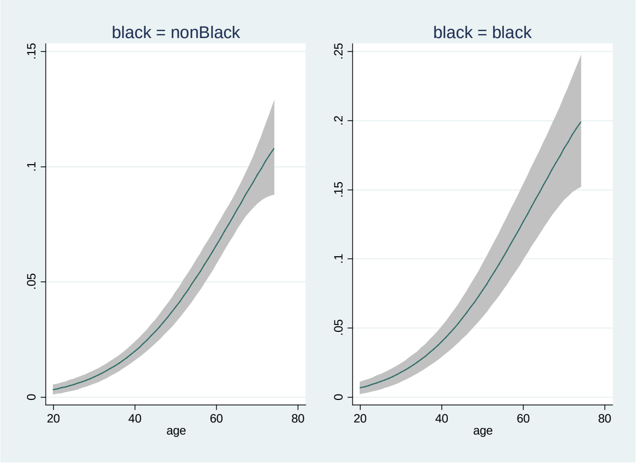
```
. quietly logit diabetes i.black i.female age c.age#c.age, nolog
. mcp age black, plotopts(scheme(sj))
```



NOTE: The `plotopts(scheme(sj))` option is good if your graphics have to be viewed in black and white rather than color – you may wish to use it for all the graphics in a paper (or at least always use the same scheme, whatever it is. See `help schemes` for more details.

As we have seen before, racial differences in the likelihood of having diabetes are very small at younger ages, but increase greatly as people get older. If you want to view the confidence intervals as well, `mcp` plots each group separately to make the graph easier to read.

```
. mcp age black, ci
```



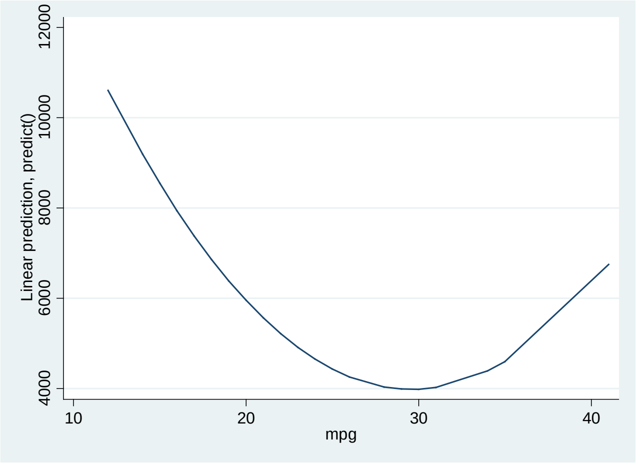
One thing I dislike about the above is that the Y axis is scaled differently for the two groups. There is probably some way to fix that but I haven’t figured out what it is yet.

Other features

Royston gives much more complicated examples. For example, he shows how to use `mcp` with fractional polynomials and with spline functions.

We’ve been focusing on logistic regression, but mcp can be useful with many other techniques, especially if there is some sort of nonlinearity in the effects, e.g. a linear regression model that includes a squared term:

```
. sysuse auto, clear
(1978 Automobile Data)
. quietly reg price mpg c.mpg#c.mpg
. mcp mpg
```



Conversely, like `margins` itself, `mcp` may be less useful with multiple outcome commands like `ologit` and `mlogit`. Other commands and approaches may be better.