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

Richard Williams, University of Notre Dame, https://www3.nd.edu/~rwilliam/
Last revised September 10, 2024

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, Black people are 4 percentage points more likely to have diabetes than are comparable White people. 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) )
```
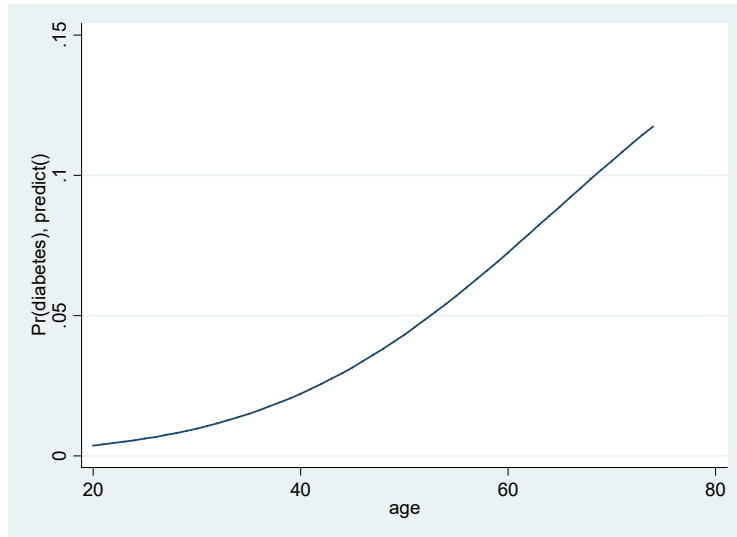
The `show` option is optional; it tells `mcp` to display the `margins` command it generated internally, which can be helpful for making sure you are generating what you want to generate.

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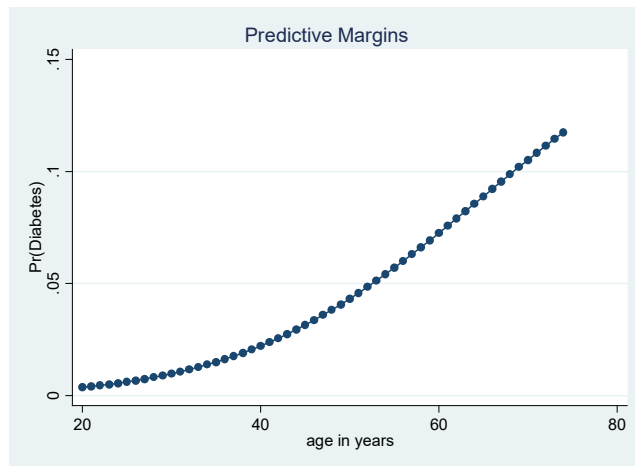
```
. mcp age, show margopts(atmeans)
```

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

```
. mcp age, ci
```
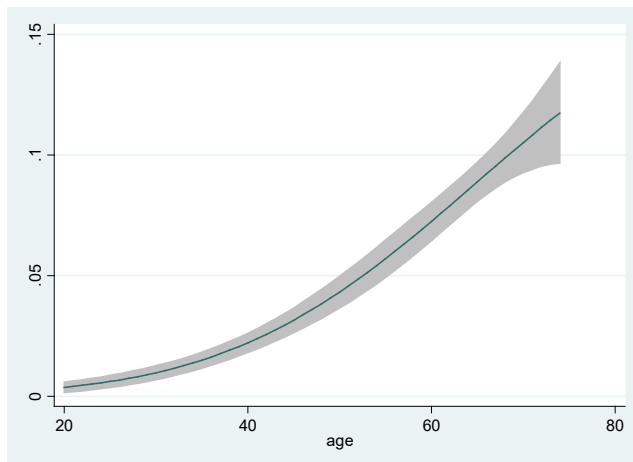
We could do pretty much the same thing with `margins` and `marginsplot`:

```
. 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. While this often works well, there can be problems with this approach.

- First, the continuous variable may have many values; potentially, every case could have a unique value. Estimation can then be very slow and `mcp` may not work at all. Further, it is highly unlikely that you need to plot so many values to get a good graph.
- Second, there may be times when you want to make predictions outside the range of observed values, e.g. you want to see the predicted probability of diabetes for someone who is older than anyone in your sample.
- Conversely, third, you may want to plot only a limited range of the observed values. 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. For example, with the Nhanes2f data, 99% of the cases have a weight of 116 kg or less; but the heaviest person

has a weight of 176 kg. When there are extreme outliers, a large portion of your graph can be taken up plotting values for very rare and atypical cases.

There are various ways of dealing with these issues. First, we can use the `var1` option. As the `mcp` help explains, "If var1(#) is specified, then # equally spaced values of xvar1 are used as plotting positions, encompassing the observed range of xvar1." So, for example, if we specify var1(20), 20 equally spaced points will be plotted:

```
. mcp age, var1(20) show
margins , at( age=( 20 22.84210586548 25.68420982361 28.52631568909 31.36842155457
34.2105255127 37.05263137817 39.8
> 9473724365 42.73684310913 45.57894897461 48.42105102539 51.26315689087
54.10526275635 56.94736862183 59.7894744873
>  62.63158035278 65.47368621826 68.31578826904 71.15789794922 74) )
```
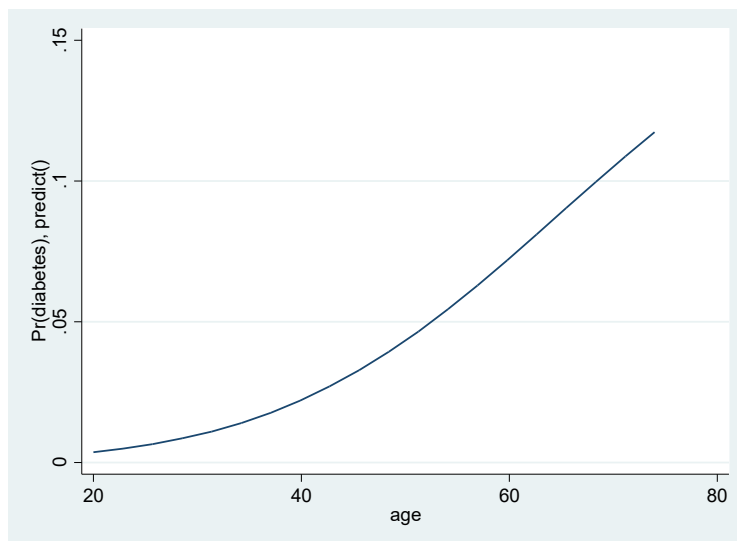

This looks pretty much identical to the earlier graph, but you can specify a larger value (e.g. 50) if you want/need even more precision. I quite frequently specify `var1(20)` or `var1(50)`. You don't have to worry about the xvar having too many values and picking 20 points to plot is probably almost always enough.

Use of `var1` still leaves you open to the other two problems: you don't see predictions outside the observed range, and extreme outliers could still force the X axis to be extended greatly. The `at1` option may therefore be better in many cases.

Using the `at1` option, you can control the range yourself, making it either broader or narrower than the observed range. 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) )
```
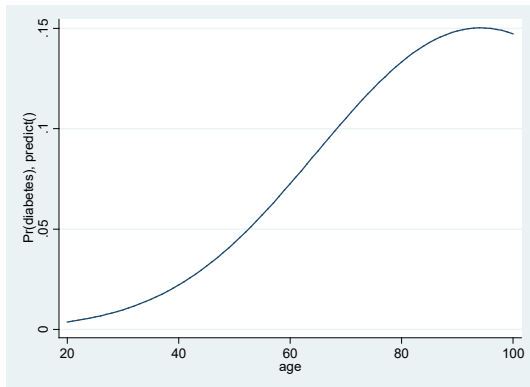
Because of the squared term, we know that at some point the predicted effect of age should start declining, and the graph shows that this happens sometime after age 90 (although again I wouldn't trust a prediction that is so far outside the range of the observed data; in a moment we'll see what may be 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) )
```
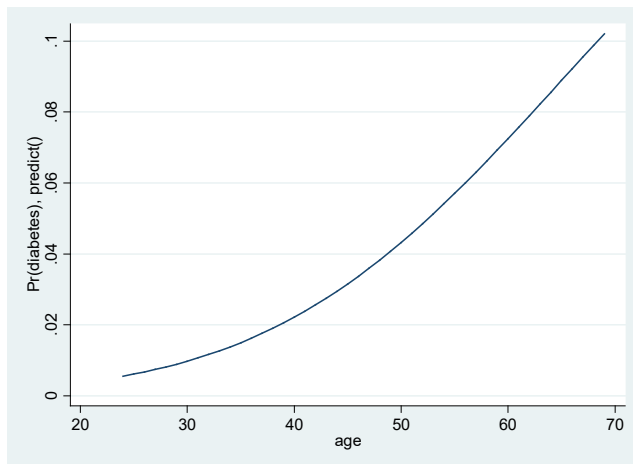

In this case you see that ages less than 23 and greater than 69 were excluded from the graph. Royston provides a more dramatic example, where 99% of the cases have a value of 998 or less but the highest value is 2,380.

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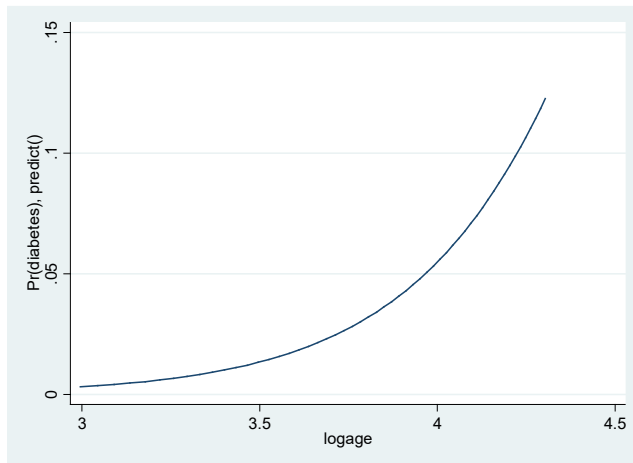
As we see, the effect of logage is highly significant. If we viewed this and the model with X^2 as competing theories, both the BIC and AIC tests favor the model with logage.

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: (NOTE: I find all these commands hard to remember, so when I do something like this I refer back to this handout and use this example as a template.)

```
. 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) )
```
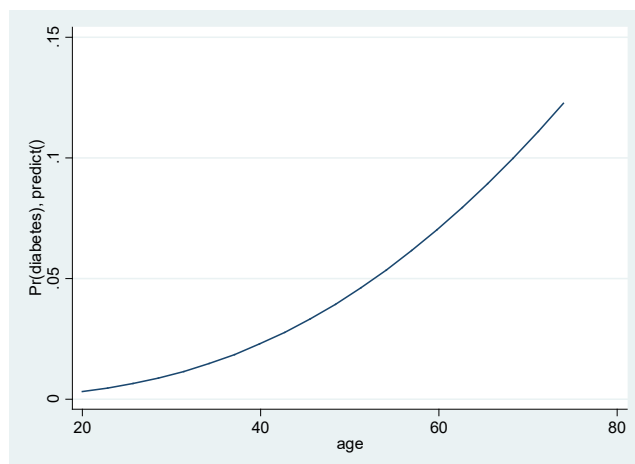

mcp plotted the AAPs for each of the 20 values of logage that had been computed. However, by
saying

```
. 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))
```
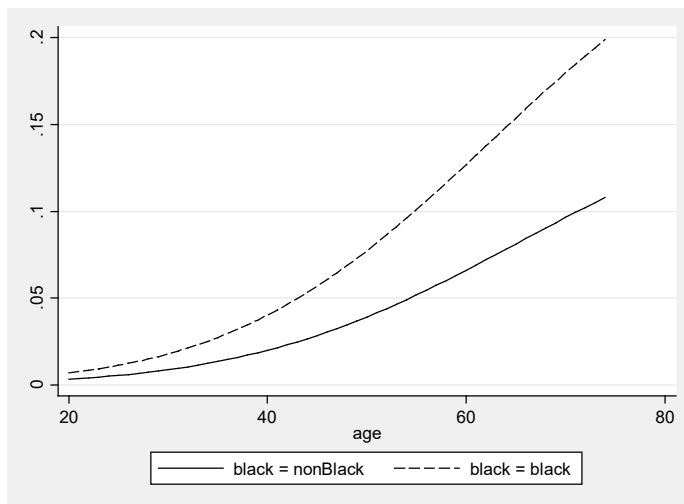

NOTE: The plotopts(scheme(sj)) option is good if your graphics have to be viewed in black and white rather than color – you may wish to use it for all the graphics in a paper (or at least always use the same scheme, whatever it is). See help schemes for more details. Alternatively, as Patrick Royston pointed out to me, if you want all your graphics to use scheme(sj), you can precede your commands with

```
set scheme sj
```

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
```
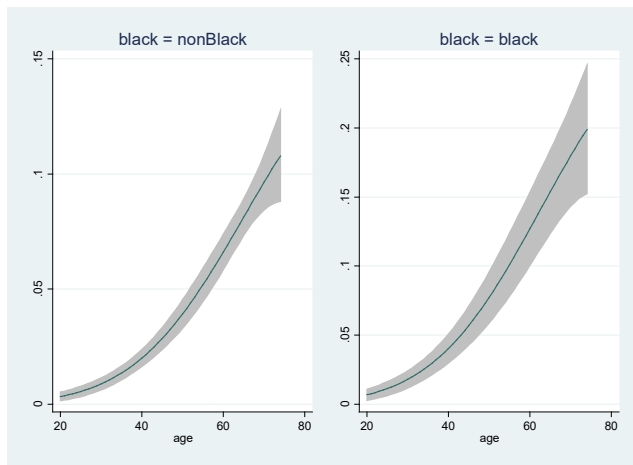

One thing I dislike about the above is that the Y axis is scaled differently for the two groups,
making comparisons difficult across groups. Patrick Royston emailed me and noted that, if you
want to, you can change that by adding the `plotopts(ycommon)` option:

```
.  mcp age black, ci plotopts(ycommon)
```
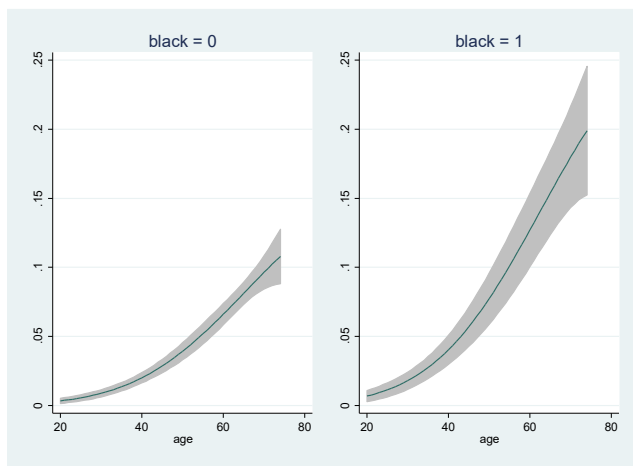

Note that the x2 variable doesn't have to be categorical. You can use a continuous variable, BUT
be sure to limit it to only a few plotting values!!! Otherwise mcp will try to plot a separate line
for each value of x2. In the following example, I add weight to the model; and I have the
adjusted predictions for weight plotted at ages 20 (young), 45 (middle aged) and 70 (old). I also
exclude the extreme outliers on weight.

```
. quietly logit diabetes i.black i.female age c.age#c.age c.weight, nolog
. mcp weight age, at1(% 1 (1) 99) at2(20 45 70) plotopts(scheme(sj)) show
margins , at( weight=( 44.0418 46.61 47.97 49.22 50.01 50.8 51.71 52.28 52.96 53.52
54.21 54.77 55.34 55.91 56.36 56
> .93 57.38 57.83 58.29 58.63 59.08 59.53 59.99 60.33 60.67 61.12 61.58 61.9904 62.26
62.71 63.05 63.39 63.73 64.18
> 64.64 64.98 65.43 65.77 66.11 66.57 66.79 67.13 67.47 67.93 68.38 68.72 69.17 69.51
69.97 70.42 70.76 71.1 71.44 7
```

```
> 1.9 72.24 72.58 73.03 73.48 73.94 74.28 74.73 75.1316 75.52 75.86 76.32 76.89 77.34
77.91 78.36 78.81 79.15 79.61
> 80.06 80.63 81.25 81.76 82.21 82.7404 83.35 83.92 84.6 85.16 85.73 86.52 87.09 87.88
88.68 89.59 90.5884 91.63 93.
> 1 94.24 95.82 97.64 99.68 102.2276 104.9932 108.75 115.7382) age=( 20 45 70))
```
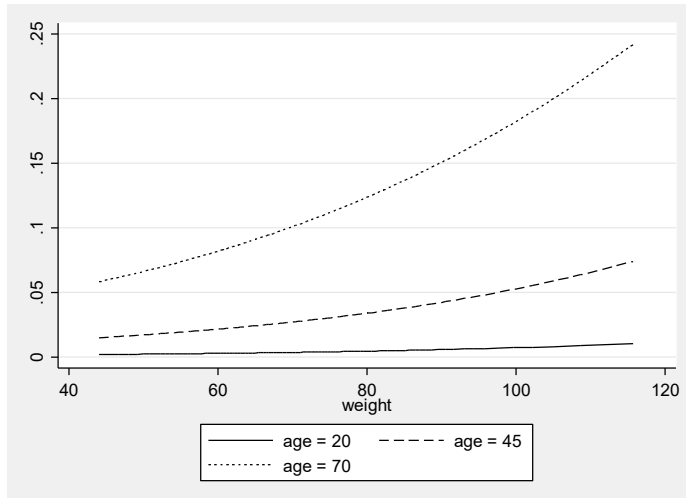

Basically, the graph suggests that young people can get by with being heavier, at least in terms of their likelihood of having diabetes. But the older you are, the greater the risk of being heavier.

Incidentally, see what happens to the graph if I do not exclude outliers on weight:

```
. mcp weight age, var1(30) at2( 20 45 70) plotopts(scheme(sj)) show
margins , at( weight=( 30.84000015259 35.841381073 40.84275817871 45.84413909912
50.84551620483 55.84689712524 60.84
> 827423096 65.84965515137 70.85103607178 75.85241699219 80.8537902832 85.85517120361
90.85655212402 95.85793304443
> 100.8593139648 105.8606872559 110.8620681763 115.8634490967 120.8648300171
125.8662033081 130.8675842285 135.86897
> 27783 140.8703460693 145.8717193604 150.8731079102 155.8744812012 160.8758544922
165.877243042 170.878616333 175.8
> 800048828) age=( 20 45 70))
```
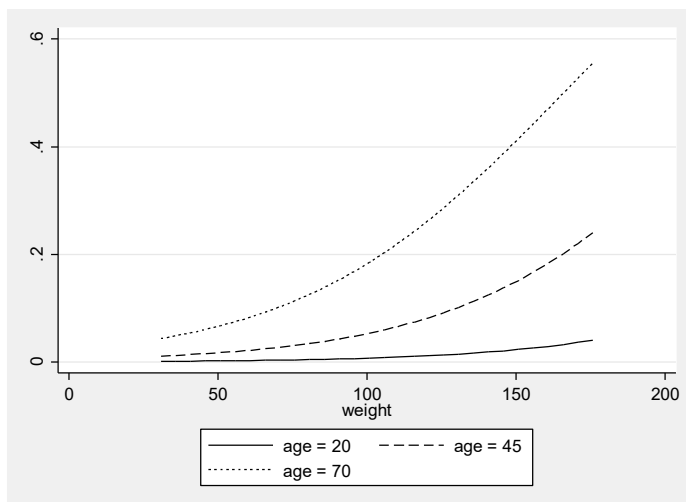
Basically, those 1% of the cases with weight > 116 kg take up a huge portion of the graph. That isn't necessarily bad, but the more extreme the outliers, the more impact they will have on how the graph looks.

OTHER FEATURES

Royston gives much more complicated examples. For example, he shows how to use `mcp` with fractional polynomials and with spline functions. The appendix gives an example for spline functions that Royston emailed me.

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
```
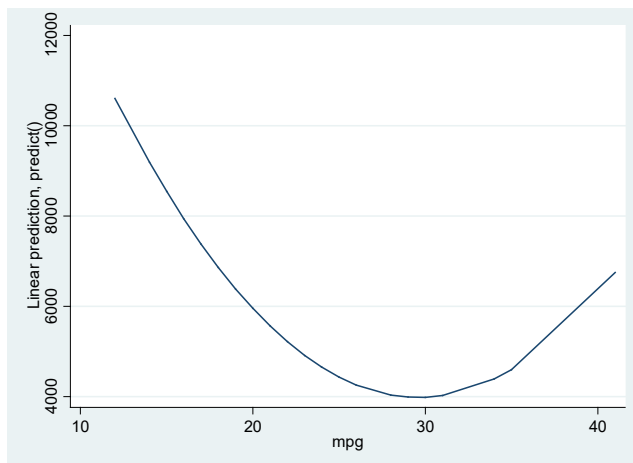

Conversely, `mcp` may be less useful with multiple outcome commands like `ologit` and `mlogit`. Other commands and approaches may be better.

Appendix: Analysis with Spline Models (Optional)

In Section 6.2 of his Stata Journal Article, Patrick Royston notes that `mcp` can be used for the analysis of spline models. He describes the procedure but, because of space limitations, does not provide an actual example. Royston graciously emailed me an example and said I was free to include it in an appendix. If you are not familiar with spline models, my own (somewhat primitive) notes on them are currently located at https://www3.nd.edu/~rwilliam/stats2/l61.pdf.

Here (copied verbatim) is how Royston describes the procedure in section 6.2 of his paper (page 524). FP refers to fractional polynomial models he has earlier discussed.

6.2   Analysis with spline models

The techniques described above for plotting results from FP models can be applied in a similar fashion to spline models or indeed to any model involving nonlinear transformations of continuous predictors. Space limitation prevents me from giving full details. In principle, the steps for spline modeling are as follows:

1. Determine a suitable spline model for the data.

2. Create the required spline basis variables for all continuous covariates with a nonlinear effect. (This is analogous to the `fracgen` operations described above. You can, for example, use Stata's `mkspline` command.)

3. Fit the selected model, including the spline basis variables and, if necessary, any important interactions.

4. For a given continuous predictor whose marginal effect is to be plotted, create a variable holding a limited number of plotting positions.

5. Transform the plotting-positions variable into the requisite number of spline basis variables, using the same knot numbers and positions as in the main analysis. These transformed variables provide the "look-up table" for plotting the fit on the original scale of the predictor (for example, plotting at actual ages rather than at meaningless transformed values).

6. Run `marginscontplot` with an appropriate syntax.

Of course, this simple schema hides many issues connected with spline model selection. These issues are beyond the present scope.

Here is the example Royston sent me.

```
. set scheme sj
. /*
>         Spline example, 4 knots at 5 35 65 95 centiles of age.
> */
. global stub ex3b

. webuse nhanes2f, clear
. keep if sex==1
(5,428 observations deleted)

. gen map = (bpsystol + 2 * bpdiast)/3
```

```
. gen bmi = weight / (height/100)^2

. // Generate splines for age (3 basis variables, zage1-zage3)
. mkspline zage = age, cubic nknots(4)

. matrix knots = r(knots)
. local knots

. forvalues j = 1 / 4 {
  2.        local k = knots[1, `j']
  3.        local knots `knots' `k'
  4. }

. regress map zage1 zage2 zage3 i.race bmi hgb race#c.(zage1 zage2 zage3)
c.bmi#c.(zage1 zage2 zage3) c.hgb#c.(zage1 zage2 zage3)

      Source |       SS           df       MS      Number of obs   =     4,909
-------------+----------------------------------   F(19, 4889)     =     70.80
       Model |  207315.291         19  10911.3311   Prob > F        =    0.0000
    Residual |   753450.02      4,889  154.111274   R-squared       =    0.2158
-------------+----------------------------------   Adj R-squared   =    0.2127
       Total |  960765.311      4,908  195.754953   Root MSE        =    12.414


------------------------------------------------------------------------------
         map |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       zage1 |   .6891498    .866393     0.80   0.426     -1.00937    2.387669
       zage2 |  -.6338776   2.597714    -0.24   0.807    -5.726564    4.458809
       zage3 |    3.00986   5.968327     0.50   0.614    -8.690742    14.71046
             |
        race |
       Black |  -6.056786   5.286132    -1.15   0.252    -16.41998    4.306408
       Other |  -18.51118   10.46613    -1.77   0.077    -39.02949    2.007124
             |
         bmi |   1.161587   .4080617     2.85   0.004     .3616028    1.961571
         hgb |   2.610948   1.545072     1.69   0.091    -.4180872    5.639983
             |
 race#c.zage1 |
       Black |    .253888   .1889982     1.34   0.179    -.1166335    .6244095
       Other |   .5891385   .3719267     1.58   0.113    -.1400049    1.318282
             |
 race#c.zage2 |
       Black |  -.0472001   .5911214    -0.08   0.936    -1.206064    1.111663
       Other |  -1.804639   1.248918    -1.44   0.149     -4.25308    .6438023
             |
 race#c.zage3 |
       Black |  -.5428157   1.385617    -0.39   0.695    -3.259247    2.173615
       Other |   5.148125   3.047475     1.69   0.091    -.8262966    11.12255
             |
c.bmi#c.zage1 |   .0011632   .0143081     0.08   0.935     -.026887    .0292135
             |
c.bmi#c.zage2 |   .0088059   .0442823     0.20   0.842    -.0780072    .0956191
             |
c.bmi#c.zage3 |  -.0671791   .1039884    -0.65   0.518    -.2710431    .1366849
             |
c.hgb#c.zage1 |  -.0440967   .0538712    -0.82   0.413    -.1497084     .061515
             |
c.hgb#c.zage2 |    .066313   .1616061     0.41   0.682    -.2505075    .3831335
             |
c.hgb#c.zage3 |  -.1794604   .3711939    -0.48   0.629    -.9071672    .5482465
             |
       _cons |   24.43988   24.90575     0.98   0.326    -24.38658    73.26634
------------------------------------------------------------------------------
```

```
. // Create spline-transformed plotting ages
. local ages 20 22 24 26 28 30 35 40 45 50 55 60 70

. gen ages = .
(4,909 missing values generated)

. tokenize `ages'

. local i 1

. while "``i''" != "" {
  2.         qui replace ages = ``i'' in `i'
  3.         local ++i
  4. }

. mkspline wage = ages, cubic knots(`knots')
.
. local stuff lp(l - _ _- -.) lwidth(medthick ..)

. marginscontplot age (zage1 zage2 zage3), var1(ages(wage1 wage2 wage3)) ci ///
>  plotopts(title("(a) Age only", placement(west)) leg(off) name(g1, replace))
```
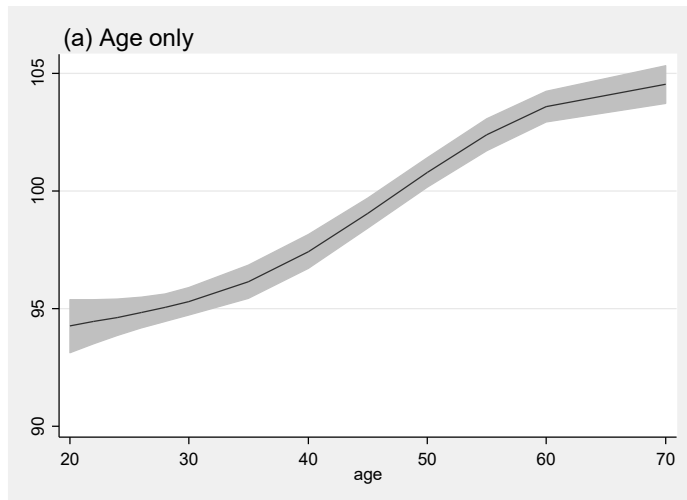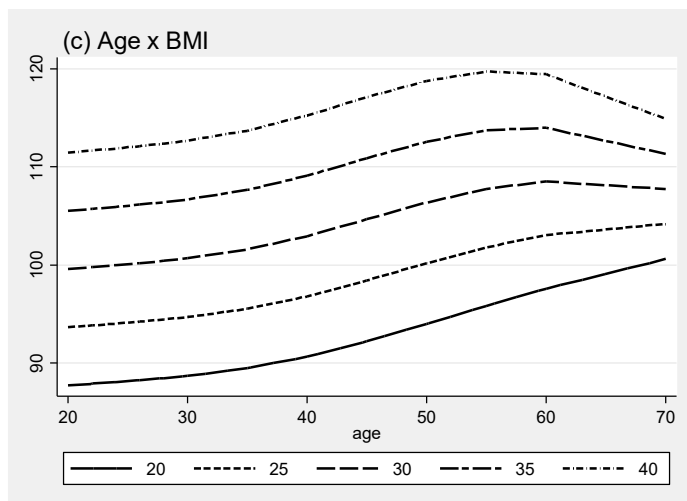

(a) Age only

```
. marginscontplot age (zage1 zage2 zage3) race, var1(ages(wage1 wage2 wage3)) ///
> plotopts(`stuff' title("(b) Age x race", placement(west)) legend(label(1 "White")
label(2 "Black") label(3 "Other") row(1)) name(g2, replace))
```

(b) Age x race
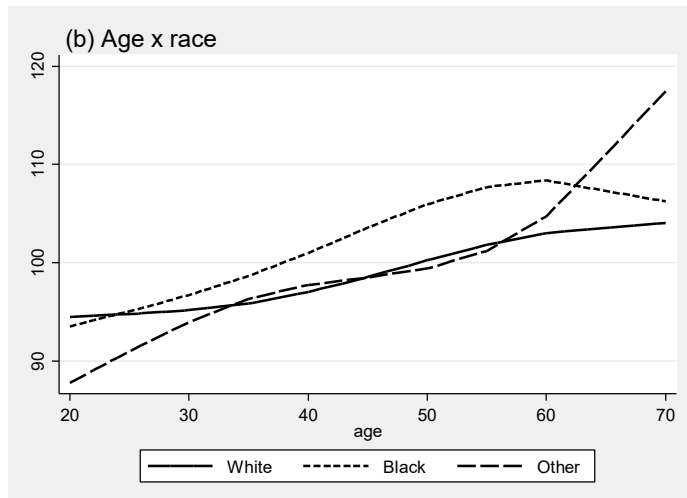
```
. marginscontplot age (zage1 zage2 zage3) bmi, at2(20(5)40) var1(ages(wage1 wage2
wage3)) ///
> plotopts(`stuff' title("(c) Age x BMI", placement(west)) legend(label(1 "20")
label(2 "25") label(3 "30") label(4 "35") label(5 "40") row(1)) name(g3, replace))
```


(c) Age x BMI

```
. marginscontplot age (zage1 zage2 zage3) hgb, at2(13(1)16) var1(ages(wage1 wage2
wage3)) ///
> plotopts(`stuff' title("(d) Age x haemoglobin", placement(west)) legend(label(1
"13") label(2 "14") label(3 "15") label(4 "16") label(5 "17") row(1)) name(g4, re
> place))
```

(d) Age x haemoglobin
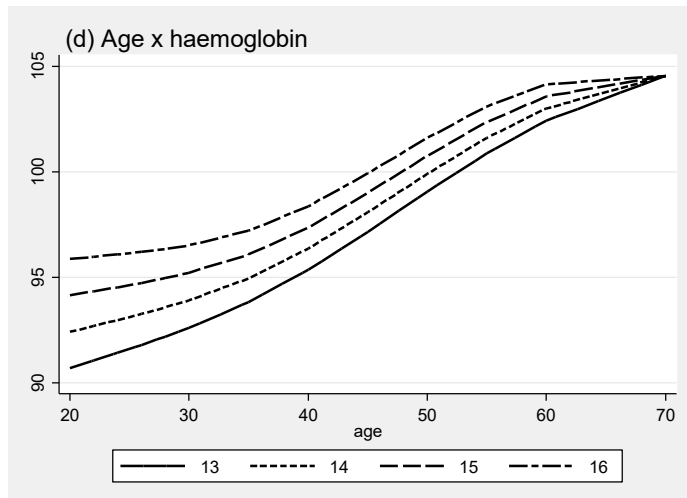
```
. graph combine g1 g2 g3 g4, imargin(small) saving($stub, replace)
(file ex3b.gph saved)
```
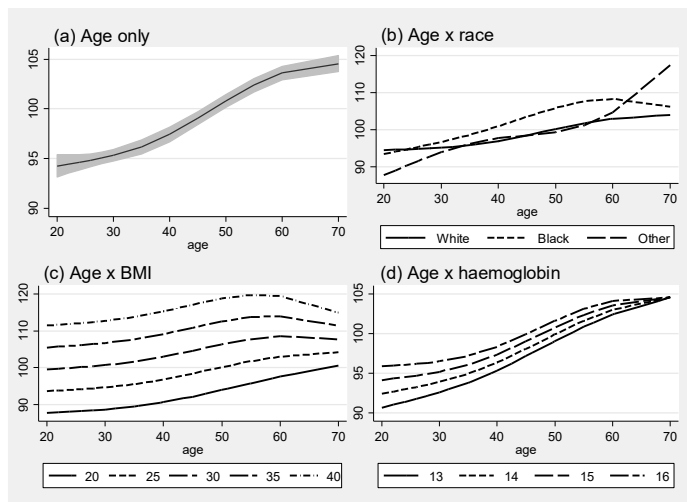


```
. graph export $stub.eps, replace
```
(file ex3b.eps written in EPS format)

```
. graph drop g1 g2 g3 g4
```