The Stata Journal

The *Stata Journal* publishes reviewed papers together with shorter notes or comments, regular columns, book reviews, and other material of interest to Stata users. Examples of the types of papers include 1) expository papers that link the use of Stata commands or programs to associated principles, such as those that will serve as tutorials for users first encountering a new field of statistics or a major new technique; 2) papers that go "beyond the Stata manual" in explaining key features or uses of Stata that are of interest to intermediate or advanced users of Stata; 3) papers that discuss new commands or Stata programs of interest either to a wide spectrum of users (e.g., in data management or graphics) or to some large segment of Stata users (e.g., in survey statistics, survival analysis, panel analysis, or limited dependent variable modeling); 4) papers analyzing the statistical properties of new or existing estimators and tests in Stata; 5) papers that could be of interest or usefulness to researchers, especially in fields that are of practical importance but are not often included in texts or other journals, such as the use of Stata in managing datasets, especially large datasets, with advice from hard-won experience; and 6) papers of interest to those who teach, including Stata with topics such as extended examples of techniques and interpretation of results, simulations of statistical concepts, and overviews of subject areas.

The *Stata Journal* is indexed and abstracted by *CompuMath Citation Index*, *Current Contents/Social and Behavioral Sciences*, *RePEc: Research Papers in Economics*, *Science Citation Index Expanded* (also known as *SciSearch*, *Scopus*, and *Social Sciences Citation Index*.

For more information on the *Stata Journal*, including information for authors, see the webpage

http://www.stata-journal.com

Subscriptions are available from StataCorp, 4905 Lakeway Drive, College Station, Texas 77845, telephone 979-696-4600 or 800-STATA-PC, fax 979-696-4601, or online at

http://www.stata.com/bookstore/sj.html

Subscription rates listed below include both a printed and an electronic copy unless otherwise mentioned.

U.S. and Canada		Elsewhere	
Printed & electronic		**Printed & electronic**	
1-year subscription	$ 98	1-year subscription	$138
2-year subscription	$165	2-year subscription	$245
3-year subscription	$225	3-year subscription	$345
1-year student subscription	$ 75	1-year student subscription	$ 99
1-year university library subscription	$125	1-year university library subscription	$165
2-year university library subscription	$215	2-year university library subscription	$295
3-year university library subscription	$315	3-year university library subscription	$435
1-year institutional subscription	$245	1-year institutional subscription	$285
2-year institutional subscription	$445	2-year institutional subscription	$525
3-year institutional subscription	$645	3-year institutional subscription	$765
Electronic only		**Electronic only**	
1-year subscription	$ 75	1-year subscription	$ 75
2-year subscription	$125	2-year subscription	$125
3-year subscription	$165	3-year subscription	$165
1-year student subscription	$ 45	1-year student subscription	$ 45

Back issues of the *Stata Journal* may be ordered online at

http://www.stata.com/bookstore/sjj.html

Individual articles three or more years old may be accessed online without charge. More recent articles may be ordered online.

http://www.stata-journal.com/archives.html

The *Stata Journal* is published quarterly by the Stata Press, College Station, Texas, USA.

Address changes should be sent to the *Stata Journal*, StataCorp, 4905 Lakeway Drive, College Station, TX 77845, USA, or emailed to sj@stata.com.

The Stata Journal (2013)
13, Number 3, pp. 510–527

marginscontplot: Plotting the marginal effects of continuous predictors

Patrick Royston
Hub for Trials Methodology Research
MRC Clinical Trials Unit at University College London
London, UK
pr@ctu.mrc.ac.uk

Abstract. I provide a new tool (`marginscontplot`) for plotting the marginal effect of continuous covariates in regression models. The plots may be univariate or according to levels or user-selected values of a second covariate. Nonlinear relationships involving transformed covariates may be plotted on the original scale.

Keywords: gr0056, marginscontplot, mcp, regression models, continuous covariates, margins, graphing marginal effects, interactions

1 Introduction

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.

2 Comments on margins

To the beginner, the meaning of the term "margin" may be somewhat elusive. It certainly was for me. Beginners may find useful a recent article by Williams (2012). The author concentrates on using the `margins` command with categorical covariates. Here we are concerned more with continuous covariates, although categorical covariates are also supported.

Consider the simple but illustrative example of a regression in the iconic `auto.dta` of `mpg` on `foreign` and `weight`. We use the modern approach and specify `foreign` as a factor variable, letting Stata take care of any dummy variables that are needed:

```
. sysuse auto
(1978 Automobile Data)

. regress mpg i.foreign weight

      Source |       SS       df       MS              Number of obs =      74
-------------+------------------------------           F(  2,    71) =   69.75
       Model |   1619.2877     2  809.643849           Prob > F      =  0.0000
    Residual |  824.171761    71   11.608053           R-squared     =  0.6627
-------------+------------------------------           Adj R-squared =  0.6532
       Total |  2443.45946    73  33.4720474           Root MSE      =  3.4071

------------------------------------------------------------------------------
         mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
   1.foreign |  -1.650029   1.075994    -1.53   0.130      -3.7955    .4954422
      weight |  -.0065879   .0006371   -10.34   0.000    -.0078583   -.0053175
       _cons |    41.6797   2.165547    19.25   0.000     37.36172    45.99768
------------------------------------------------------------------------------
```

We apply the `margins` command to get the marginal effect of foreign manufacture, adjusting for vehicle weight:

```
. margins foreign

Predictive margins                                Number of obs   =         74
Model VCE    : OLS

Expression   : Linear prediction, predict()

------------------------------------------------------------------------------
             |            Delta-method
             |     Margin   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
     foreign |
          0  |   21.78785   .5091123    42.80   0.000        20.79    22.78569
          1  |   20.13782   .8535566    23.59   0.000     18.46488    21.81076
------------------------------------------------------------------------------
```

We see that weight for weight, the marginal means (in miles per gallon) are 21.79 (0.51) for domestic cars and 20.14 (0.85) for foreign cars. For clarification, we now demonstrate how the same estimates (without standard errors) can be obtained by manipulating predictions from the regression model directly:

```
. preserve

. replace foreign = 0
(22 real changes made)

. predict margin0
(option xb assumed; fitted values)

. summarize margin0

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
     margin0 |        74    21.78785    5.120063   9.794333   30.08502

. restore

. preserve
```

```
. replace foreign = 1
(52 real changes made)
. predict margin1
(option xb assumed; fitted values)
. summarize margin1
    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
     margin1 |        74    20.13782    5.120063   8.144303   28.43499
. restore
```

We see that the mean prediction for `foreign` set to 0 for the entire sample is identical to the margin for `foreign` = 0 and similarly for `foreign` = 1. The estimated margin is essentially an out-of-sample prediction: we pretend that the weights of the vehicles do not change when we pretend that all of them were domestic or all were foreign.

`margins` handles the presence of interaction terms in the model with aplomb:

```
. regress mpg i.foreign##c.weight
  (output omitted)
. margins foreign
Predictive margins                                Number of obs   =         74
Model VCE    : OLS
Expression   : Linear prediction, predict()

------------------------------------------------------------------------------
             |            Delta-method
             |     Margin   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
     foreign |
          0  |   21.60544   .4967815    43.49   0.000     20.63177    22.57912
          1  |   17.43754   1.360464    12.82   0.000     14.77108      20.104
------------------------------------------------------------------------------
```

Let us return to the simpler model (`regress mpg i.foreign weight`). Suppose we want the marginal effects of `weight` on `mpg`. We could type, for example,

```
. margins, at(weight=3000)
```

or

```
. margins foreign, at(weight=3000)
```

to get the miles per gallon predicted at a roughly central value of a car weight of 3,000 pounds, without or with distinguishing the effect of `foreign`. However, there are no fewer than 64 distinct values of `weight`. We cannot validly type `margins weight` to get its marginal effect, because `weight` is a continuous variable and Stata does not allow this syntax; `weight` would have to be a categorical variable for this to work. Even if we could type it, it would not be much help to see a table of the estimated margins for the 64 values of `weight`.

We would really like to create a visual assessment, for example, a plot of the margins against `weight`. The ado-file `marginscontplot` is designed to help us do just that.

3 A simple example of marginscontplot

We continue with the `auto.dta` example. The following code uses standard Stata commands, `margins` followed by `marginsplot`, to give a plot with 95% pointwise confidence intervals of the marginal effect of `weight` on `mpg`, adjusting for `foreign`:

```
. quietly regress mpg i.foreign weight
. quietly margins, at(weight = (1760(100)4840))
. marginsplot
  Variables that uniquely identify margins: weight
```

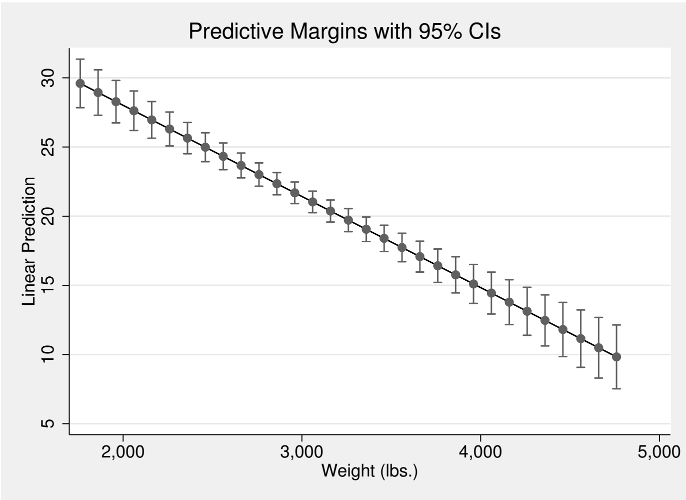

Figure 1. Marginal effect of `weight` on `mpg` with a pointwise 95% confidence interval, from a linear regression model. Produced by using `margins` and `marginsplot`.

The plot is shown in figure 1. The appearance could be improved by using a more appropriate rendition of the confidence intervals, but the required information is basically all there.

Figure 2 shows a similar plot, this time produced by a single `marginscontplot` command:

```
. quietly regress mpg i.foreign weight
. marginscontplot weight, ci
```

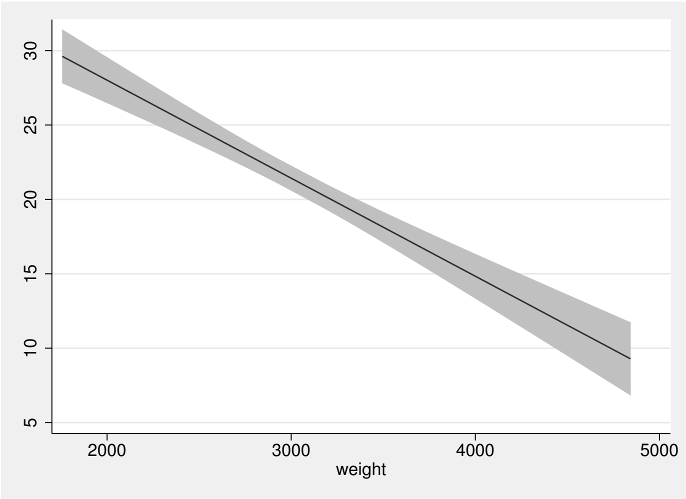

Figure 2. Marginal effect of `weight` on `mpg` with a pointwise 95% confidence interval, from a linear regression model. Produced by using `marginscontplot`.

Although you do not see it, `marginscontplot` is working behind the scenes to generate such plots. Let x be the covariate that is used. By default, `marginscontplot` extracts the unique values of x observed in the sample and evaluates the margin at each unique value, together with a pointwise confidence interval if the `ci` option has been specified. To do this, it calls the `margins` command with a suitably constructed `at()` option that defines the plotting values. The constructed `at()` option resembles the `margins` option `at(weight = (1760(100)4840))` we have already seen, except that all 64 unique values of `weight` are spelled out. `marginscontplot` saves the resulting estimates to a separate, temporary file, which it loads, manipulates if necessary, and plots.

If we complicate the model by adding an interaction term, no difficulty accrues (figure not shown):

```
. regress mpg i.foreign##c.weight
. marginscontplot weight, ci
```

In the case of a linear function (as here), we only actually need two values, for example, the minimum and maximum of observed x, to define the line. However, this would not display the confidence interval accurately. A practical suggestion is 10 to 20 values, equally spaced over the range of x. The number of such values is controlled by the `var1()` and `var2()` options of `marginscontplot`. Alternatively, you can supply your own choice of x values by including them directly in the `at1()` or `at2()` option, for example,

```
. marginscontplot weight, ci var1(20)
. marginscontplot weight, ci at1(2000(500)4500)
```

Here we are working with univariate plots (just one variable), and we need only the `at1()` or `var1()` option to specify the range of x values for plotting. Later, we shall see how the `at2()` and `var2()` options are used for plotting at values of a second covariate.

4 An example from survival analysis

We use the German breast cancer dataset to illustrate two issues:

1. How `marginscontplot` handles margins in nonlinear regression models such as `stcox`.
2. The use of the percentile feature of the `at1()` option to limit the range of plotting values.

We load the data by using the `webuse` command, and we `stset` it for use as survival data:

```
. webuse brcancer
(German breast cancer data)
. stset rectime, failure(censrec)
     failure event:  censrec != 0 & censrec < .
obs. time interval:  (0, rectime]
 exit on or before:  failure
------------------------------------------------------------------------------
      686  total obs.
        0  exclusions
------------------------------------------------------------------------------
      686  obs. remaining, representing
      299  failures in single record/single failure data
   771400  total analysis time at risk, at risk from t =         0
                             earliest observed entry t =         0
                                  last observed exit t =      2659
```

Suppose we fit a Cox proportional hazards model on the continuous covariates `x5e` and `x6`. We wish to see the effect of `x6` on the relative hazard of an event:

```
. quietly stcox x5e x6
. marginscontplot x6, ci
```

The result is shown in figure 3(a).

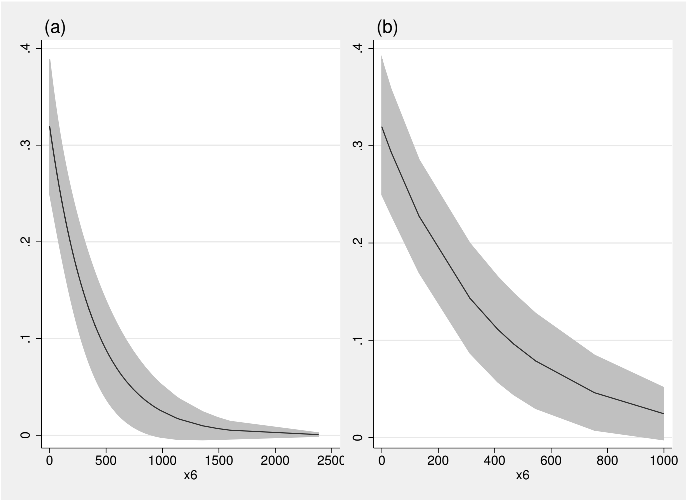

Figure 3. Marginal effect of `x6` on the relative hazard in a Cox model, with pointwise 95% confidence intervals. (a) Full range of `x6`; (b) range restricted to $(1, 99)$ centiles.

It appears that larger values of `x6` are associated with a (dramatically) lower relative hazard. Note that although we have fit `x6` as a linear term in the Cox model, its marginal effect appears as a curve rather than a straight line. The reason is that by default, `predict` for `stcox` predicts the relative hazard, whereas the regression coefficient, `_b[x6]`, and hence, the linear predictor `xb`, acts on the log relative-hazard scale. If we wanted to see the effect of `x6` on the log relative-hazard, we would use the `margopts()` option with `predict(xb)` to specify the linear predictor (figure not shown):

```
. marginscontplot x6, ci margopts(predict(xb))
```

The observed distribution of `x6` is markedly skewed (coefficient of skewness = 4.8), having a concentration of values at 0 and a small number of large values. For example, the 99th centile is 998 fmol/l, whereas the range extends out to 2,380 fmol/l. We can limit the plotting values according to chosen centiles of `x6` by using the `%` prefix in the `at1()` option, for example,

```
. marginscontplot x6, ci at1(%1 10 25 50 75 90 95(1)99)
```

This restricts the range of the plot [see figure 3(b)].

5 Examples with transformed x

5.1 Log transformation

Suppose the relationship between the response variable, y, and x is log linear. Such a situation is not uncommon. We wish to model $E(y)$ as a linear function of log x, and we want to graph the relationship on the original scale of x, not the scale of log x. Let us return to `auto.dta` for an example that achieves the aim. We model `mpg` as a log-linear function of `weight`:

```
. quietly generate logwt = ln(weight)
. quietly regress mpg logwt
```

Suppose we decide to plot at 20 values of `weight` equally spaced between the observed lowest and the highest weights. Stata's `range` command can conveniently be used to create a new variable (say, `w`) containing such values:

```
. summarize weight
. range w r(min) r(max) 20
```

Next we log-transform `w`, and use `w` and the transformed values in the `var1()` option of `marginscontplot`:

```
. quietly generate logw = ln(w)
. marginscontplot weight(logwt), var1(w(logw)) ci
```

The result is shown in figure 4.

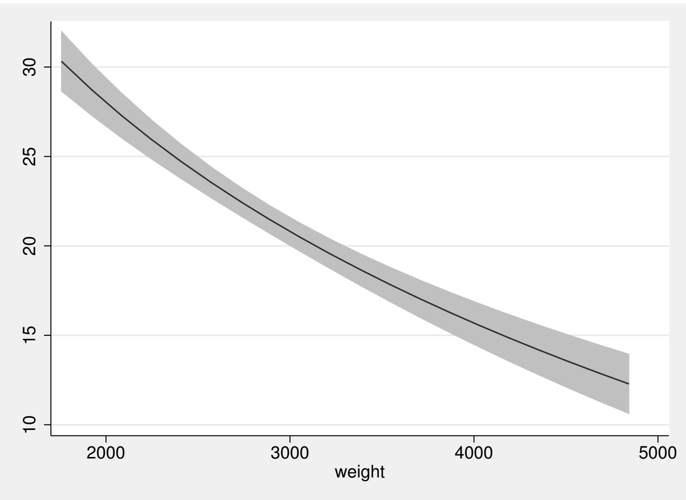

Figure 4. Plot of the marginal effect of `weight` on `mpg` in `auto.dta` when log(`weight`) is fit.

5.2 Fractional polynomials

What if the relationship between the response variable, y, and x is more complex? A simple class of functions for modeling quite a wide range of nonlinear functions is fractional polynomials (FPs) (Royston and Altman 1994). These are essentially extensions of ordinary polynomials that allow noninteger and negative values of the polynomial power transformations. This greatly increases their flexibility. For example, a quadratic $\beta_0 + \beta_1 x^1 + \beta_2 x^2$ generalizes to the FP2 function $\beta_0 + \beta_1 x^{p_1} + \beta_2 x^{p_2}$, where p_1 and p_2 are powers belonging to the restricted set $S = \{-2, -1, -0.5, 0, 0, 5, 1, 2, 3\}$. By convention, x^0 means $\ln(x)$. Also included are "repeated powers" models of the form $\beta_0 + \beta_1 x^{p_1} + \beta_2 x^{p_1} \ln(x)$. FP1 functions have the simple form $\beta_0 + \beta_1 x^{p_1}$ for p_1 in S.

Many details of FPs and their modeling, both in a univariate and in a multivariable context, are provided in Royston and Sauerbrei (2008). In Stata, univariate FPs are implemented in the command `fracpoly`, and multivariable FPs (models) in `mfp`. In Stata 13, `mfp` is unchanged, but `fracpoly` has been superseded by a new command, `fp` (although `fracpoly` continues to work). `marginsconplot` works properly with `fp`. For example, the information carried by FP-transformed variables created by `fp` or `fp generate` is used appropriately by `marginscontplot`.

Consider extending the `auto.dta` example to allow an FP function of `weight`:

```
. fracpoly: regress mpg weight foreign
........
-> gen double Iweig__1 = X^-2-.1096835742 if e(sample)
-> gen double Iweig__2 = X^-2*ln(X)-.121208886 if e(sample)
   (where: X = weight/1000)
      Source |       SS       df       MS              Number of obs =      74
-------------+------------------------------           F(  3,    70) =   52.95
       Model |  1696.05946     3  565.353152           Prob > F      =  0.0000
    Residual |  747.400002    70  10.6771429           R-squared     =  0.6941
-------------+------------------------------           Adj R-squared =  0.6810
       Total |  2443.45946    73  33.4720474           Root MSE      =  3.2676

------------------------------------------------------------------------------
         mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
    Iweig__1 |   15.88527   20.60329     0.77   0.443    -25.2067    56.97724
    Iweig__2 |   127.9349   47.53106     2.69   0.009    33.13723    222.7327
     foreign |  -2.222516   1.053782    -2.11   0.039   -4.324218   -.1208133
       _cons |   20.95519    .569564    36.79   0.000    19.81923    22.09115
------------------------------------------------------------------------------
Deviance:   381.13. Best powers of weight among 44 models fit: -2 -2.
```

`fracpoly` has determined that the best-fitting FP2 model has powers $(-2, -2)$ for `weight`; that is, a function of the form $\beta_1 x^{-2} + \beta_2 x^{-2} \ln(x)$ has been selected. How do we produce a margins plot of the fitted function of `weight`? We need to tell the software the `weight` values at which we want to plot the fit and the corresponding FP-transformed values on which the regression model has been fit. Suppose we decide to plot at 20 values of `weight` equally spaced between the observed lowest and the highest weights. We take the approach described above in the log-linear example to create a new variable (say, `w1`) containing such values:

```
. quietly summarize weight
. range w1 r(min) r(max) 20
```

Now we compute the required FP transformation of `w1`, namely, x^{-2} and $x^{-2}\ln(x)$. Some care is needed. We see from the `fracpoly` output given above that for each new variable `fracpoly` created, namely, `Iweig__1` and `Iweig__2`, it has actually scaled `weight` by dividing by 1,000 before creating each transformation, and then added a constant. For the regression parameters to be valid for out-of-sample prediction at new variables, say, `w1a` and `w1b`, derived from `w1`, we must mimic that process precisely:

```
. generate w1a = (w1/1000)^-2-.1096835742
. generate w1b = (w1/1000)^-2 * ln(w1/1000)-.121208886
```

To create the margins plot, including a pointwise 95% confidence interval for the fit, we run `marginscontplot` with the `var1()` and `ci` options. The `var1()` option provides a look-up table between our plotting positions (stored in `w1`) and their FP transformations (stored in `w1a` and `w1b`):

```
. marginscontplot weight (Iweig__1 Iweig__2), var1(w1 (w1a w1b)) ci
```

The syntax `weight (Iweig__1 Iweig__2)` has a flavor similar to `var1(w1 (w1a w1b))`. The original predictor is `weight`, and `Iweig__1 Iweig__2` expresses how `weight` appears in the regression model. The resulting graph is shown in figure 5.

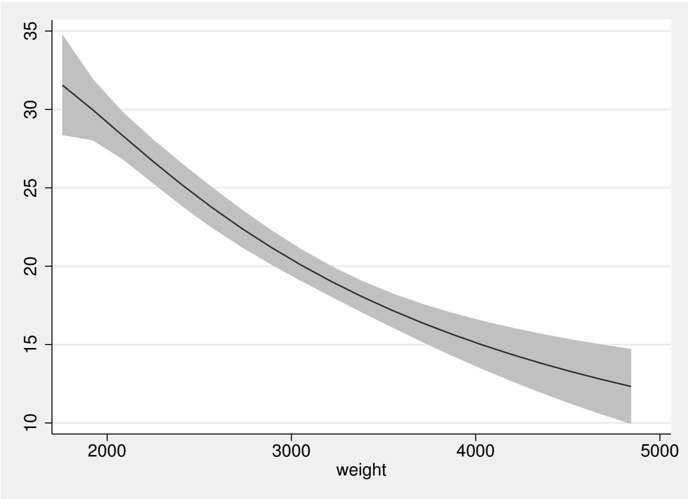

Figure 5. Marginal effect of `weight` on `mpg` with a pointwise 95% confidence interval, from a two-term fractional polynomial regression model.

The fitted function shows curvature in the functional form (cf. figure 4).

We can easily elaborate the plot to show the effect of `weight` according to `foreign` status as follows (see figure 6):

```
. marginscontplot weight (Iweig__1 Iweig__2) foreign, var1(w1 (w1a w1b)) ci
```

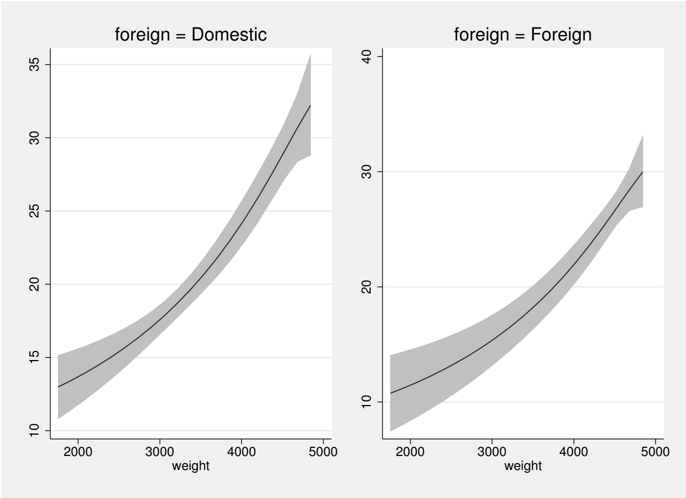

Figure 6. Marginal effect of `weight` on `mpg` by the two values of `foreign` with a pointwise 95% confidence interval, from a fractional polynomial regression model.

Note that the x dimension for the plot is always the first-mentioned variable, and the "by-variable" is the second, namely, `foreign`. The program figures out that margins for `weight` at the two distinct values of `foreign` are needed. If the `ci` option is omitted, the same two lines are plotted on the same graph, making them easier to compare directly but sacrificing information on their precision.

The program `fracpoly` stores in characteristics the details of the rather complicated FP transformations it performed:

```
. char list Iweig__1[fp]
  Iweig__1[fp]:              X^-2-.1096835742: X = weight/1000
. char list Iweig__2[fp]
  Iweig__2[fp]:              X^-2*ln(X)-.121208886: X = weight/1000
```

To make the plotting process easier, `marginscontplot` can use this information internally to transform user-specified values of `weight` to the values it needs to compute the margins:

```
. marginscontplot weight (Iweig__1 Iweig__2), var1(20) ci
```

In this context, the option `var1(20)` says "plot the fit at 20 values of `weight` equally spaced between the minimum and maximum".

A third way to achieve the same result is to generate the FP-transformed variables for weight yourself by using fracgen, run the regression, and finally run marginscontplot:

```
. fracgen weight -2 -2
. local fp2_weight `r(names)´
. display "`fp2_weight´"
weight_1 weight_2
. regress mpg i.foreign `fp2_weight´
. marginscontplot weight (`fp2_weight´), var1(20) ci
```

fracgen creates FP-transformed variables, here called weight_1 and weight_2, silently storing their details in characteristics. For convenience, I have stored these names in a local macro called fp2_weight. By default, fracgen omits the constants −0.1096835742 and −0.121208886, but this has no effect on marginscontplot.

6 A more complex example

6.1 Analysis with fractional polynomials

We move from auto.dta to data from a biomedical survey, nhanes2f.dta. The dataset is available via the command webuse nhanes2f.

The dataset comprises observations of 10,337 individuals: 4,909 males and 5,428 females. We concentrate on modeling blood pressure in the males (sex==1). There are two measures of blood pressure in the dataset: bpsystol (systolic blood pressure) and bpdiast (diastolic blood pressure). Because they are physiologically and statistically highly correlated, we model a composite measure known as mean arterial pressure (MAP):

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

Known predictors of blood pressure are age and body mass index (BMI, equal to weight in kg divided by the square of height in m). Modeling MAP with mfp shows that hemoglobin (hgb) and race (race, coded 1 = white, 2 = black, and 3 = other) are also significant predictors of MAP in a multivariable context. Age needs an FP2 model with powers $(-2, -1)$, whereas BMI and hemoglobin seem to have linear effects.

There are six possible two-way multiplicative interactions between the four predictors. Of these, three are highly significant ($P < 0.001$) in a model that includes all the main effects and interactions: FP2(age) × race, FP2(age) × BMI, and FP2(age) × hemoglobin. (In these expressions, we include two variables to represent the FP2 transformation of age.) We thus arrive at a rather complex model that includes four main effects and three interactions, all of which involve a nonlinear transformation of age:

```
. fracgen age -2 -1
-> gen double age_1 = X^-2
-> gen double age_2 = X^-1
   (where: X = age/10)

. regress map age_1 age_2 i.race bmi hgb race#c.(age_1 age_2)
> c.bmi#c.(age_1 age_2) c.hgb#c.(age_1 age_2)

      Source |       SS       df       MS              Number of obs =    4909
-------------+------------------------------           F( 14,  4894) =   94.56
       Model |  204564.182    14  14611.7273           Prob > F      =  0.0000
    Residual |  756201.129  4894  154.515964           R-squared     =  0.2129
-------------+------------------------------           Adj R-squared =  0.2107
       Total |  960765.311  4908  195.754953           Root MSE      =   12.43

------------------------------------------------------------------------------
         map |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       age_1 |   777.3663   275.0243     2.83   0.005     238.1953    1316.537
       age_2 |  -583.2943   160.1266    -3.64   0.000    -897.2143   -269.3742
             |
        race |
          2  |   1.062011   4.643472     0.23   0.819    -8.041279     10.1653
          3  |   24.75007   9.477148     2.61   0.009     6.170607    43.32953
             |
         bmi |   .0241705     .35309     0.07   0.945    -.6680443    .7163853
         hgb |  -1.907729   1.248318    -1.53   0.127    -4.354992    .5395346
             |
race#c.age_1 |
          2  |  -77.04519   60.46379    -1.27   0.203    -195.5814    41.49096
          3  |   155.6329   118.9581     1.31   0.191     -77.5783    388.8441
             |
race#c.age_2 |
          2  |   30.53207    35.9084     0.85   0.395     -39.8645    100.9286
          3  |  -135.8852   71.43399    -1.90   0.057    -275.9278    4.157523
             |
c.bmi#c.age_1|  -11.19275   4.635915    -2.41   0.016    -20.28122   -2.104274
             |
c.bmi#c.age_2|   7.566864   2.732302     2.77   0.006     2.210326     12.9234
             |
c.hgb#c.age_1|  -22.17118   17.11388    -1.30   0.195    -55.72206    11.37971
             |
c.hgb#c.age_2|   17.56146   9.997191     1.76   0.079    -2.037522    37.16044
             |
       _cons |   147.2989   19.92625     7.39   0.000     108.2345    186.3633
------------------------------------------------------------------------------
```

The table of estimates looks pretty complicated. How do we interpret the results? To gain understanding, we work with graphs created by `marginscontplot`. We start with the marginal effect of age alone, and then we explore the three age interactions. Figure 7(a) shows the marginal relationship between MAP and age.

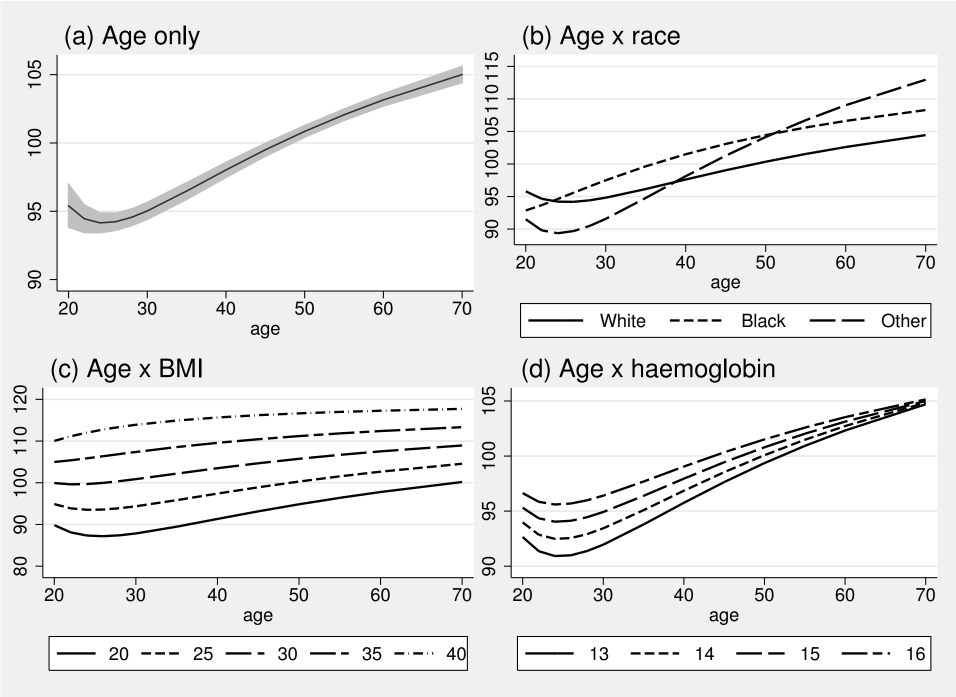

Figure 7. Marginal plots of the effect of `age` on `map` in `nhanes2f.dta`. a) Age alone; b) age by race; c) age by BMI at five selected values of BMI; d) age by hemoglobin at four selected values of hemoglobin. The model includes three interactions with FP2 transformations of `age`.

Mean MAP shows a clear increase with age. (The inflection point at about 25 years may be an artifact of the FP2 model. It seems more plausible that the underlying curve is roughly "flat" between about 20 and 30 years, but the FP2 family is not flexible enough to mimic this behavior.)

The remaining three plots show the interactions with age. In figure 7(b), we see somewhat different shapes of the functions across races, against a generally increasing trend. Figure 7(c) shows there is a strong effect of BMI at all ages, which is a little stronger in younger people. Comparing it with figure 7(a), we see that the effect of age adjusted for BMI is noticeably smaller than the marginal effect of age. Finally, figure 7(d) suggests that higher hemoglobin is associated with higher blood pressure at young ages but not in older people.

An example of the `marginscontplot` command needed to produce such plots is given below for panel (c):

```
. marginscontplot age (age_1 age_2) bmi, at1(20(2)30 35(5)60 70) at2(20(5)40)
> plotopts(lpattern(l - _ _- -.) lwidth(medthick ..) name(g3, replace)
> 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)))
```

Much of the code here is needed in `plotopts()` to produce a satisfying plot appearance.

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. However, I am happy to provide an example file on request suggesting how to produce the spline model equivalent of figure 7.

7 The marginscontplot command

7.1 Syntax

The syntax of `marginscontplot` is as follows:

{`marginscontplot`|`mcp`} *xvar1* [`(`*xvar1a* [*xvar1b* ...] `)`] [*xvar2* [`(`*xvar2a* [*xvar2b* ...] `)`]] [*if*] [*in*] [`,` `at(`*at_list*`)` `at1(`[`%`] *at1_list*`)` `at2(`[`%`] *at2_list*`)` `ci` `margopts(`*string*`)` `nograph` `plotopts(`*twoway_options*`)` `saving(`*filename* [`, replace`] `)` `showmarginscmd` `var1(`#|*var1_spec*`)` `var2(`#|*var2_spec*`)`]

The options are described below.

7.2 Description

`marginscontplot` provides a graph of the marginal effect of a continuous predictor on the response variable in the most recently fit regression model. When only *xvar1* is provided, the plot of marginal effects is univariate at values of *xvar1* specified by the `at1()` or `var1()` option. When both *xvar1* and *xvar2* are provided, the plot of marginal effects is against values of *xvar1* specified by the `at1()` or `var1()` option for fixed values of *xvar2* specified by the `at2()` or `var2()` option. A line is plotted for each specified value of *xvar2*.

`marginscontplot` has the distinctive ability to plot marginal effects on the original scale of *xvar1* or *xvar2*, even when the model includes transformed values of *xvar1* or *xvar2* but not *xvar1* or *xvar2* themselves. Such a situation arises in FP or spline modeling, for example, where nonlinear relationships with continuous predictors are to be approximated, and transformed covariates are included in the model to achieve this.

`mcp` is a synonym for `marginscontplot` for those who prefer to type less.

7.3 Options

`at(`*at_list*`)` fixes values of model covariates other than *xvar1* and *xvar2*. *at_list* has syntax *varname1* = # [*varname2* = # ...]. By default, predictions for such covariates are made at the observed values and averaged across observations.

`at1(`[`%`] *at1_list*`)` defines the plotting positions for *xvar1* through the numlist *at1_list*. If the prefix `%` is included, *at1_list* is interpreted as percentiles of the distribution of *xvar1*. If `at1()` is omitted, all the observed values of *xvar1* are used if feasible. Note that *xvar1* is always treated as the primary plotting variable on the x dimension.

`at2(`[`%`] *at2_list*`)` defines the plotting positions for *xvar2* through the numlist *at2_list*. If the prefix `%` is included, *at2_list* is interpreted as percentiles of the distribution of *xvar2*. If `at2()` is omitted, all the observed values of *xvar2* are used if feasible. Note that *xvar2* is always treated as the secondary "by-variable" for plotting purposes.

`ci` displays pointwise confidence intervals for the fitted values on the margins plot. For legibility, if more than one line is specified, each line is plotted on a separate graph.

`margopts(`*string*`)` supplies options to the `margins` command. The option most likely to be needed is `predict(xb)`, which means that predicted values and, hence, margins are on the scale of the linear predictor. For example, in a logistic regression model, the default predictions are of the event probabilities. Specifying the option `margopts(predict(xb))` gives margins on the scale of the linear predictor, that is, the predicted log odds of an event.

Note that the margins are calculated with the default setting, `asobserved`, for `margins`. See `help margins` for further information.

`nograph` suppresses the graph of marginal effects.

`plotopts(`*twoway_options*`)` are options of `graph twoway`; see [G-3] ***twoway_options***.

`saving(`*filename*[`, replace`]`)` saves the calculated margins and their confidence intervals to a file (*filename*`.dta`). This can be useful for fine-tuning the plot or tabulating the results.

`showmarginscmd` displays the `margins` command that `marginscontplot` creates and issues to Stata to do the calculations necessary for constructing the plot. This information can be helpful in fine-tuning the command or identifying problems.

`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.

`var2(`#|*var2_spec*`)` specifies plotting values of *xvar2*. If `var2(`#`)` is specified, then # equally spaced values of *xvar2* are used as plotting positions, encompassing the observed range of *xvar2*. Alternatively, *var2_spec* may be used to specify transformed plotting values of *xvar2*. The syntax of *var2_spec* is *var2* [`(`*var2a* [*var2b* `...`]`)`]. *var2* is a variable holding user-specified plotting values of *xvar2*. *var2a* is a variable holding transformed values of *var2* and similarly for *var2b* ... if required.

7.4 Remarks

The version of `var1()` with *var1_spec* is appropriate for use after any covariate transformation is used in the model and you want a plot with the original (untransformed) covariate on the horizontal axis. This includes simple transformations such as logs and more complicated situations. For example, the model may involve an FP model in *xvar1* using `fracpoly` or `mfp`. Alternatively, FP transformations of *xvar1* may be calculated using `fracgen`, and the required model fit to the transformed variables before applying `marginscontplot`. The same facility is also available for the `var2()` option. It works in the same way but with *xvar2* instead of *xvar1*.

`marginscontplot` has been designed to handle quite high-dimensional cases, that is, cases where many margins must be estimated. Be aware, however, that the number of margins is limited by the maximum matrix size; see `help matsize`. This can be increased if necessary by using the `set matsize` # command. `marginscontplot` tells you the smallest value of # needed to accommodate the case in question.

8 Concluding comments

When continuous variables are modeled, especially in a nonlinear fashion, plotting plays a vital role in understanding the nature of their association with the response variable. This is particularly true when interactions are involved because tables of regression coefficients are inadequate tools for understanding relationships. I hope that `marginscontplot` will continue the good work started by `margins` and `marginsplot` in understanding outputs from such models.

9 Acknowledgment

I thank an anonymous reviewer for most helpful comments and suggestions.

10 References

Royston, P., and D. G. Altman. 1994. Regression using fractional polynomials of continuous covariates: Parsimonious parametric modelling (with discussion). *Journal of the Royal Statistical Society, Series C* 43: 429–467.

Royston, P., and W. Sauerbrei. 2008. *Multivariable Model-building: A Pragmatic Approach to Regression Analysis Based on Fractional Polynomials for Modelling Continuous Variables*. Chichester, UK: Wiley.

Williams, R. 2012. Using the margins command to estimate and interpret adjusted predictions and marginal effects. *Stata Journal* 12: 308–331.

About the author

Patrick Royston is a medical statistician with more than 30 years of experience and with a strong interest in biostatistical methods and in statistical computing and algorithms. He works largely in methodological issues in the design and analysis of clinical trials and observational studies. He is currently focusing on alternative outcome measures in trials with a time-to-event outcome; on problems of model building and validation with survival data, including prognostic factor studies and treatment-covariate interactions; on parametric modeling of survival data; and on novel clinical trial designs.