fp — Fractional polynomial regression

Description

fp <*term*>: *est_cmd* fits models with the "best"-fitting fractional polynomial substituted for <*term*> wherever it appears in *est_cmd*. fp <weight>: regress mpg <weight> foreign would fit a regression model of mpg on a fractional polynomial in weight and (linear) foreign.

By specifying option fp(), you may set the exact powers to be used. Otherwise, a search through all possible fractional polynomials up to the degree set by dimension() with the powers set by powers() is performed.

fp without arguments redisplays the previous estimation results, just as typing *est_cmd* would. You can type either one. fp will include a fractional polynomial comparison table.

fp generate creates fractional polynomial power variables for a given set of powers. For instance, fp <weight>: regress mpg <weight> foreign might produce the fractional polynomial $\texttt{weight}^{(-2,-1)}$ and store $\texttt{weight}^{-2}$ in weight_1 and $\texttt{weight}^{-1}$ in weight_2. Typing fp generate weight^(-2 -1) would allow you to create the same variables in another dataset.

See [R] **mfp** for multivariable fractional polynomial models.

Quick start

Fit models with fractional polynomials

Find optimal second-degree fractional polynomial of x1 in regression of y on x2 and x3

```
fp <x1>: regress y <x1> x2 x3
```

Same as above, but search only powers of −1, −0.5, 1, and 2.

```
fp <x1>, power(-1 -.5 1 2): regress y <x1> x2 x3
```

Same as above, but allow search to include third-degree fractional polynomials

```
fp <x1>, power(-1 -.5 1 2) dimension(3): regress y <x1> x2 x3
```

Fit model including $\texttt{x1}^{-2}$ and $\texttt{x1}^{2}$ without performing search

```
fp <x1>, fp(-2 2): regress y <x1> x2 x3
```

Rescale x1 to nonextreme positive values when computing fractional polynomials

```
fp <x1>, scale: regress y <x1> x2 x3
```

Same as above, and center fractional polynomial of x1 at its scaled mean

```
fp <x1>, center scale: regress y <x1> x2 x3
```

Set fractional polynomial to zero for nonpositive values of x1

```
fp <x1>, zero: regress y <x1> x2 x3
```

Same as above, and include an indicator variable in the model for nonpositive values of x1

```
fp <x1>, catzero: regress y <x1> x2 x3
```

Create variables corresponding to fractional polynomial powers

Generate `x1_1` and `x1_2` corresponding to $\texttt{x1}^{-2}$ and $\texttt{x1}^{2}$

```
fp generate x1^(-2 2)
```

Same as above, but generate fractional polynomial variables with automatic scaling and centering

```
fp generate x1^(-2 2), center scale
```

Note: In the above examples, `regress` could be replaced with any estimation command allowing the `fp` prefix.

Menu

fp

Statistics > Linear models and related > Fractional polynomials > Fractional polynomial regression

fp generate

Statistics > Linear models and related > Fractional polynomials > Create fractional polynomial variables

Syntax

Estimation

`fp` <*term*> [, *est_options*] : *est_cmd*

Specify that fractional powers of varname be calculated during estimation

`fp` <*term*>(*varname*) [, *est_options*] : *est_cmd*

Replay estimation results

`fp` [, *replay_options*]

Create specified fractional polynomial power variables

`fp generate` [*type*] [*newvar* `=`] *varname*`^`(*numlist*) [*if*] [*in*] [, *gen_options*]

est_cmd may be almost any estimation command that stores the `e(ll)` result. To confirm whether `fp` works with a specific *est_cmd*, see the documentation for that *est_cmd*. *est_cmd* may not contain other prefix commands; see **[U] 11.1.10 Prefix commands**.

Instances of <*term*> (with the angle brackets) that occur within *est_cmd* are replaced in *est_cmd* by a varlist containing the fractional powers of the variable *term*. These variables will be named *term*_1, *term*_2, ...

`fp` performs *est_cmd* with this substitution, fitting a fractional polynomial regression in *term*.

est_options	Description
Main	
powers(# # ... #)	powers to be searched; default is powers(-2 -1 -.5 0 .5 1 2 3)
dimension(#)	maximum degree of fractional polynomial; default is dimension(2)
fp(# # ... #)	use specified fractional polynomial
Options	
classic	perform automatic scaling and centering and omit comparison table
replace	replace existing fractional polynomial power variables named *term*_1, *term*_2, ...
all	generate *term*_1, *term*_2, ... in all observations; default is in observations if esample()
scale(#_*a* #_*b*)	use (*term*+*a*)/*b*; default is to use variable term as is
scale	specify *a* and *b* automatically
center(#_*c*)	report centered-on-*c* results; default is uncentered results
center	specify *c* to be the mean of (scaled) *term*
zero	set *term*_1, *term*_2, ... to zero if scaled *term* ≤ 0; default is to issue an error message
catzero	same as zero and include *term*_0 $=$ (*term* ≤ 0) among fractional polynomial power variables
Reporting	
replay_options	specify how results are displayed

replay_options	Description
Reporting	
nocompare	do not display model-comparison test results
reporting_options	any options allowed by *est_cmd* for replaying estimation results

gen_options	Description
Main	
replace	replace existing fractional polynomial power variables named *term*_1, *term*_2, ...
scale(#_*a* #_*b*)	use (*term*+*a*)/*b*; default is to use variable term as is
scale	specify *a* and *b* automatically
center(#_*c*)	report centered-on-*c* results; default is uncentered results
center	specify *c* to be the mean of (scaled) *term*
zero	set *term*_1, *term*_2, ... to zero if scaled *term* ≤ 0; default is to issue an error message
catzero	same as zero and include *term*_0 $=$ (*term* ≤ 0) among fractional polynomial power variables

collect is allowed with fp and fp generate; see [U] **11.1.10 Prefix commands**.

Options

Options are presented under the following headings:

Options for fp
Options for fp generate

Options for fp

Main

`powers(# # ... #)` specifies that a search be performed and details about the search provided. `powers()` works with the `dimension()` option; see below. The default is `powers(-2 -1 -.5 0 .5 1 2 3)`.

`dimension(#)` specifies the maximum degree of the fractional polynomial to be searched. The default is `dimension(2)`.

If the defaults for both `powers()` and `dimension()` are used, then the fractional polynomial could be any of the following 44 possibilities:

$$
\begin{array}{c}
\textit{term}^{(-2)} \\
\textit{term}^{(-1)} \\
\vdots \\
\textit{term}^{(3)} \\
\textit{term}^{(-2)}, \textit{term}^{(-2)} \\
\textit{term}^{(-2)}, \textit{term}^{(-1)} \\
\vdots \\
\textit{term}^{(-2)}, \textit{term}^{(3)} \\
\textit{term}^{(-1)}, \textit{term}^{(-2)} \\
\vdots \\
\textit{term}^{(3)}, \textit{term}^{(3)}
\end{array}
$$

`fp(# # ... #)` specifies that no search be performed and that the fractional polynomial specified be used. `fp()` is an alternative to `powers()` and `dimension()`.

Options

`classic` performs automatic scaling and centering and omits the comparison table. Specifying `classic` is equivalent to specifying `scale`, `center`, and `nocompare`.

`replace` replaces existing fractional polynomial power variables named *term*_1, *term*_2, ...

`all` specifies that *term*_1, *term*_2, ... be filled in for all observations in the dataset rather than just for those in `e(sample)`.

`scale(#_a #_b)` specifies that *term* be scaled in the way specified, namely, that (*term*+*a*)/*b* be calculated. All values of the scaled term are required to be greater than zero unless you specify options `zero` or `catzero`. Values should not be too large or too close to zero, because by default, cubic powers and squared reciprocal powers will be considered. When `scale(a b)` is specified, values in the variable *term* are not modified; `fp` merely remembers to scale the values whenever powers are calculated.

You will probably not use `scale(`*a b*`)` for values of *a* and *b* that you create yourself, although you could. It is usually easier just to `generate` a scaled variable. For instance, if *term* is `age`, and `age` in your data is required to be greater than or equal to 20, you might generate an `age5` variable, for use as *term*:

```
. generate age5 = (age-19)/5
```

`scale(`*a b*`)` is useful when you previously fit a model using automatic scaling (option `scale`) in one dataset and now want to create the fractional polynomials in another. In the first dataset, `fp` with `scale` added `notes` to the dataset concerning the values of *a* and *b*. You can see them by typing

```
. notes
```

You can then use `fp generate, scale(`*a b*`)` in the second dataset.

The default is to use *term* as it is used in calculating fractional powers; thus, *term*'s values are required to be greater than zero unless you specify options `zero` or `catzero`. Values should not be too large, because by default, cubic powers will be considered.

`scale` specifies that *term* be scaled to be greater than zero and not too large in calculating fractional powers. See *Scaling* for more details. When `scale` is specified, values in the variable *term* are not modified; `fp` merely remembers to scale the values whenever powers are calculated.

`center(`*#_c*`)` reports results for the fractional polynomial in (scaled) *term*, centered on *c*. The default is to perform no centering.

$term^{(p_1,p_2,\ldots,p_m)}$-$c^{(p_1,p_2,\ldots,p_m)}$ is reported. This makes the constant coefficient (intercept) easier to interpret. See *Centering* for more details.

`center` performs `center(`*c*`)`, where *c* is the mean of (scaled) *term*.

`zero` and `catzero` specify how nonpositive values of *term* are to be handled. By default, nonpositive values of *term* are not allowed, because we will be calculating natural logarithms and fractional powers of *term*. Thus, an error message is issued.

`zero` sets the fractional polynomial value to zero for nonpositive values of (scaled) *term*.

`catzero` sets the fractional polynomial value to zero for nonpositive values of (scaled) *term* and includes a dummy variable indicating where nonpositive values of (scaled) *term* appear in the model.

Reporting

`nocompare` suppresses display of the comparison tests.

reporting_options are any options allowed by *est_cmd* for replaying estimation results.

Options for fp generate

Main

`replace` replaces existing fractional polynomial power variables named *term_1*, *term_2*, ...

`scale(`*#_a #_b*`)` specifies that *term* be scaled in the way specified, namely, that (*term*+*a*)/*b* be calculated. All values of the scaled term are required to be greater than zero unless you specify options `zero` or `catzero`. Values should not be too large or too close to zero, because by default, cubic powers and squared reciprocal powers will be considered. When `scale(`*a b*`)` is specified, values in the variable *term* are not modified; `fp` merely remembers to scale the values whenever powers are calculated.

You will probably not use `scale(`*a b*`)` for values of *a* and *b* that you create yourself, although you could. It is usually easier just to `generate` a scaled variable. For instance, if *term* is `age`, and `age` in your data is required to be greater than or equal to 20, you might generate an `age5` variable, for use as *term*:

```
. generate age5 = (age-19)/5
```

`scale(`*a b*`)` is useful when you previously fit a model using automatic scaling (option `scale`) in one dataset and now want to create the fractional polynomials in another. In the first dataset, `fp` with `scale` added `notes` to the dataset concerning the values of *a* and *b*. You can see them by typing

```
. notes
```

You can then use `fp generate, scale(`*a b*`)` in the second dataset.

The default is to use *term* as it is used in calculating fractional powers; thus, *term*'s values are required to be greater than zero unless you specify options `zero` or `catzero`. Values should not be too large, because by default, cubic powers will be considered.

`scale` specifies that *term* be scaled to be greater than zero and not too large in calculating fractional powers. See *Scaling* for more details. When `scale` is specified, values in the variable *term* are not modified; `fp` merely remembers to scale the values whenever powers are calculated.

`center(`*#_c*`)` reports results for the fractional polynomial in (scaled) *term*, centered on *c*. The default is to perform no centering.

$term^{(p_1,p_2,\ldots,p_m)}$-$c^{(p_1,p_2,\ldots,p_m)}$ is reported. This makes the constant coefficient (intercept) easier to interpret. See *Centering* for more details.

`center` performs `center(`*c*`)`, where *c* is the mean of (scaled) *term*.

`zero` and `catzero` specify how nonpositive values of *term* are to be handled. By default, nonpositive values of *term* are not allowed, because we will be calculating natural logarithms and fractional powers of *term*. Thus, an error message is issued.

`zero` sets the fractional polynomial value to zero for nonpositive values of (scaled) *term*.

`catzero` sets the fractional polynomial value to zero for nonpositive values of (scaled) *term* and includes a dummy variable indicating where nonpositive values of (scaled) *term* appear in the model.

Remarks and examples

stata.com

Remarks are presented under the following headings:

Fractional polynomial regression

Regression models based on fractional polynomial functions of a continuous covariate are described by Royston and Altman (1994).

Fractional polynomials increase the flexibility afforded by the family of conventional polynomial models. Although polynomials are popular in data analysis, linear and quadratic functions are limited in their range of curve shapes, whereas cubic and higher-order curves often produce undesirable artifacts such as edge effects and waves.

Fractional polynomials differ from regular polynomials in that 1) they allow logarithms, 2) they allow noninteger powers, and 3) they allow powers to be repeated.

We will write a fractional polynomial in x as

$$x^{(p_1,p_2,\ldots,p_m)\prime}\boldsymbol{\beta}$$

We will write $x^{(p)}$ to mean a regular power except that $x^{(0)}$ is to be interpreted as meaning $\ln(x)$ rather than $x^{(0)}=1$.

Then if there are no repeated powers in $(p_1,p_2,\ldots,p_m)$,

$$x^{(p_1,p_2,\ldots,p_m)\prime}\boldsymbol{\beta}=\beta_0+\beta_1 x^{(p_1)}+\beta_2 x^{(p_2)}+\cdots+\beta_m x^{(p_m)}$$

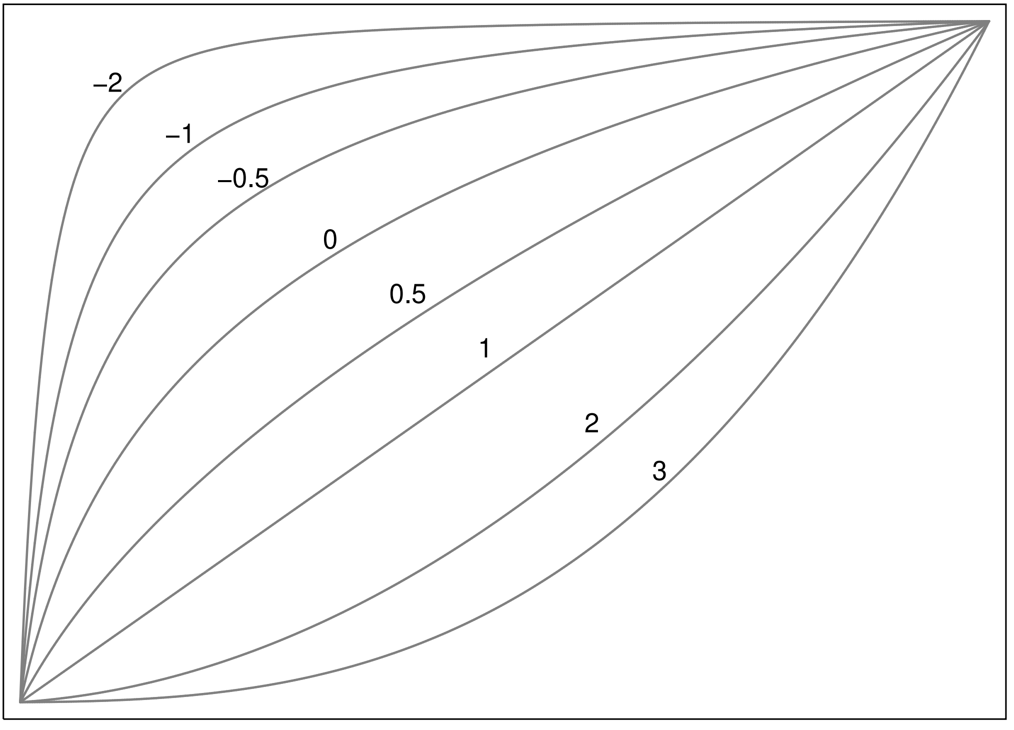
Powers are allowed to repeat in fractional polynomials. Each time a power repeats, it is multiplied by another $\ln(x)$. As an extreme case, consider the fractional polynomial with all-repeated powers, say, m of them,

$$x^{(p,p,\ldots,p)\prime}\boldsymbol{\beta}=\beta_0+\beta_1 x^{(p)}+\beta_2 x^{(p)}\ln(x)+\cdots+\beta_m x^{(p)}\{\ln(x)\}^{m-1}$$

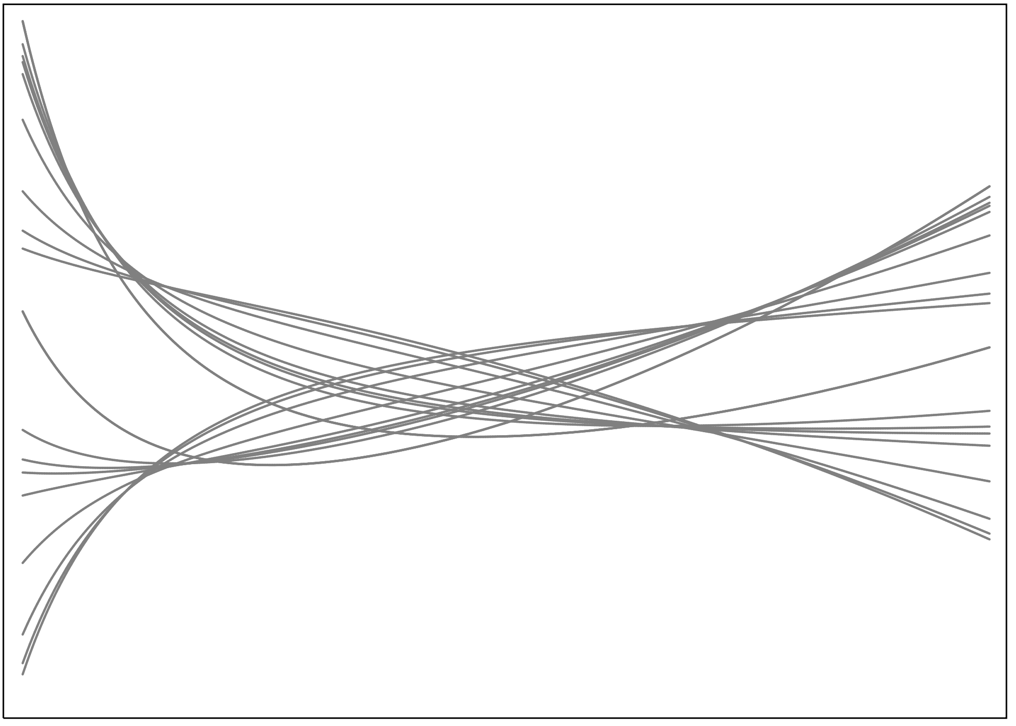
Thus, the fractional polynomial $x^{(0,0,2)\prime}\boldsymbol{\beta}$ would be

$$\begin{aligned} x^{(0,0,2)\prime}\boldsymbol{\beta} &= \beta_0+\beta_1 x^{(0)}+\beta_2 x^{(0)}\ln(x)+\beta_3 x^{(2)} \\ &= \beta_0+\beta_1\ln(x)+\beta_2\{\ln(x)\}^2+\beta_3 x^2 \end{aligned}$$

With this definition, we can obtain a much wider range of shapes than can be obtained with regular polynomials. The following graphs appeared in Royston and Sauerbrei (2008, sec. 4.5). The first graph shows the shapes of differing fractional polynomials.



The second graph shows some of the curve shapes available with different βs for the degree-2 fractional polynomial, $x^{(-2,2)}$.



In modeling a fractional polynomial, Royston and Sauerbrei (2008) recommend choosing powers from among $\{-2,-1,-0.5,0,0.5,1,2,3\}$. By default, `fp` chooses powers from this set, but other powers can be explicitly specified in the `powers()` option.

`fp` <*term*>`:` *est_cmd* fits models with the terms of the best-fitting fractional polynomial substituted for <*term*> wherever it appears in *est_cmd*. We will demonstrate with `auto.dta`, which contains repair records and other information about a variety of vehicles in 1978.

We use `fp` to find the best fractional polynomial in automobile weight (lbs.) (`weight`) for the linear regression of miles per gallon (`mpg`) on `weight` and an indicator of whether the vehicle is foreign (`foreign`).

By default, `fp` will fit degree-2 fractional polynomial (FP2) models and choose the fractional powers from the set $\{-2,-1,-0.5,0,0.5,1,2,3\}$. Because car weight is measured in pounds and will have a cubic transformation applied to it, we shrink it to a smaller scale before estimation by dividing by 1,000.

We modify the existing `weight` variable for conciseness and to facilitate the comparison of tables. When applying a data transformation in practice, rather than modifying the existing variables, you should create new variables that hold the transformed values.

```
. use https://www.stata-press.com/data/r18/auto
(1978 automobile data)
. replace weight = weight/1000
variable weight was int now float
(74 real changes made)
. fp <weight>: regress mpg <weight> foreign
(fitting 44 models)
(....10%....20%....30%....40%....50%....60%....70%....80%....90%....100%)
Fractional polynomial comparisons:
```

weight	Test df	Deviance	Residual std. dev.	Deviance diff.	P	Powers
omitted	4	456.347	5.356	75.216	0.000	
linear	3	388.366	3.407	7.236	0.082	1
m = 1	2	381.806	3.259	0.675	0.733	-.5
m = 2	0	381.131	3.268	0.000	--	-2 -2

```
Note: Test df is degrees of freedom, and P = P > F is sig. level for tests
      comparing models vs. model with m = 2 based on deviance difference,
      F(df, 68).
```

Source	SS	df	MS			
Model	1696.05949	3	565.353163	Number of obs	=	74
Residual	747.399969	70	10.6771424	F(3, 70)	=	52.95
				Prob > F	=	0.0000
				R-squared	=	0.6941
				Adj R-squared	=	0.6810
Total	2443.45946	73	33.4720474	Root MSE	=	3.2676

mpg	Coefficient	Std. err.	t	P>|t|	[95% conf.	interval]
weight_1	15.88527	20.60329	0.77	0.443	-25.20669	56.97724
weight_2	127.9349	47.53106	2.69	0.009	33.13723	222.7326
foreign	-2.222515	1.053782	-2.11	0.039	-4.324218	-.1208131
_cons	3.705981	3.367949	1.10	0.275	-3.011182	10.42314

fp begins by showing the model-comparison table. This table shows the best fractional polynomial model of weight for each examined degree, m, which is obtained by searching through all possible power combinations. The row labeled omitted describes the null model, which entirely omits weight from the model. A separate row is provided for the model with a linear function of weight because it is often the default when including a predictor in the model.

The fractional powers of the models are shown in the Powers column. An estimate of the residual standard error is given in the Residual std. dev. column. The model deviance, which we define as twice the negative log likelihood, is given in the Deviance column. The Deviance diff. column reports the difference in deviance compared with the model with the lowest deviance, which is always the model with the highest-degree fractional polynomial.

The Test df column displays the degrees of freedom used when testing a model's fit against the fit of the model with the lowest deviance. For normal error models such as linear regression, a partial F test is performed, and Test df is the numerator degrees of freedom of the F test. In other settings, a likelihood-ratio test is performed, and Test df is the degrees of freedom of the χ^2 statistic. In both cases, the p-value for the test is reported in column P.

Under robust variance estimation and some other cases (see [R] **lrtest**), the likelihood-ratio test cannot be performed. When the likelihood-ratio test cannot be performed on the model specified in *est_cmd*, fp still reports the model-comparison table, but the comparison tests are not performed.

fp reports the "best" model as the model with the lowest deviance; however, users may choose a more efficient model based on the comparison table. They may choose the lowest degree model that the partial F test (or likelihood-ratio test) fails to reject in favor of the lowest deviance model.

After the comparison table, the results of the estimation command for the lowest deviance model are shown. Here the best model has terms weight$^{(-2,-2)}$. However, based on the model-comparison table, we can reject the model without weight and the linear model at the 0.1 significance level. We fail to reject the m = 1 model at any reasonable level. We will choose the FP1 model, which includes weight$^{(-.5)}$.

We use fp again to estimate the parameters for this model. We use the fp() option to specify what powers we want to use; this option specifies that we do not want to perform a search for the best powers. We also specify the replace option to overwrite the previously created fractional polynomial power variables.

```
. fp <weight>, fp(-.5) replace: regress mpg <weight> foreign
-> regress mpg weight_1 foreign

      Source |       SS           df       MS      Number of obs   =        74
-------------+----------------------------------   F(2, 71)        =     79.51
       Model |  1689.20865         2  844.604325   Prob > F        =    0.0000
    Residual |   754.25081        71  10.6232508   R-squared       =    0.6913
-------------+----------------------------------   Adj R-squared   =    0.6826
       Total |  2443.45946        73  33.4720474   Root MSE        =    3.2593

------------------------------------------------------------------------------
         mpg | Coefficient  Std. err.      t    P>|t|     [95% conf. interval]
-------------+----------------------------------------------------------------
    weight_1 |   66.89665   6.021749    11.11   0.000     54.88963    78.90368
     foreign |  -2.095622   1.043513    -2.01   0.048    -4.176329   -.0149157
       _cons |  -17.58651   3.397992    -5.18   0.000    -24.36192   -10.81111
------------------------------------------------------------------------------
```

Alternatively, we can use fp generate to create the fractional polynomial variable corresponding to weight$^{(-.5)}$ and then use regress. We store weight$^{(-.5)}$ in the new variable wgt_nsqrt.

```
. fp generate wgt_nsqrt=weight^(-.5)

. regress mpg wgt_nsqrt foreign

      Source |       SS           df       MS      Number of obs   =        74
-------------+----------------------------------   F(2, 71)        =     79.51
       Model |  1689.20874         2  844.604371   Prob > F        =    0.0000
    Residual |  754.250718        71  10.6232495   R-squared       =    0.6913
-------------+----------------------------------   Adj R-squared   =    0.6826
       Total |  2443.45946        73  33.4720474   Root MSE        =    3.2593

------------------------------------------------------------------------------
         mpg | Coefficient  Std. err.      t    P>|t|     [95% conf. interval]
-------------+----------------------------------------------------------------
 wgt_nsqrt_1 |   66.89665   6.021748    11.11   0.000     54.88963    78.90368
     foreign |  -2.095622   1.043513    -2.01   0.048    -4.176328   -.0149155
       _cons |  -17.58651   3.397991    -5.18   0.000    -24.36191   -10.81111
------------------------------------------------------------------------------
```

Scaling

Fractional polynomials are defined only for positive *term* variables. By default, fp will assume that the variable x is positive and attempt to compute fractional powers of x. If the positive value assumption is incorrect, an error will be reported and estimation will not be performed.

If the values of the variable are too large or too small, the reported results of `fp` may be difficult to interpret. By default, cubic powers and squared reciprocal powers will be considered in the search for the best fractional polynomial in *term*.

We can scale the variable x to 1) make it positive and 2) ensure its magnitude is not too large or too small.

Suppose you have data on hospital patients with `age` as a fractional polynomial variable of interest. `age` is required to be greater than or equal to 20, so you might generate an `age5` variable by typing

```
. generate age5 = (age-19)/5
```

A unit change in `age5` is equivalent to a five-year change in age, and the minimum value of `age5` is $1/5$ instead of 20.

In the automobile example of *Fractional polynomial regression*, our *term* variable was automobile weight (lbs.). Cars weigh in the thousands of pounds, so cubing their weight figures results in large numbers. We prevented this from being a problem by shrinking the weight by 1,000; that is, we typed

```
. replace weight = weight/1000
```

Calendar year is another type of variable that can have a problematically large magnitude. We can shrink this by dividing by 10, making a unit change correspond to a decade.

```
. generate decade = calendar_year/10
```

You may also have a variable that measures deviation from zero. Perhaps `x` has already been demeaned and is symmetrically about zero. The fractional polynomial in `x` will be undefined for half of its domain. We can shift the location of `x`, making it positive by subtracting its minimum and adding a small number to it. Suppose `x` ranges from -4 to 4; we could use

```
. generate newx = x+5
```

Rescaling ourselves provides easily communicated results. We can tell exactly how the scaling was performed and how it should be performed in similar applications.

Alternatively, `fp` can scale the fractional polynomial variable so that its values are positive and the magnitude of the values are not too large. This can be done automatically or by directly specifying the scaling values.

Scaling can be automatically performed with `fp` by specifying the `scale` option. If *term* has nonpositive values, the minimum value of *term* is subtracted from each observation of *term*. In this case, the counting interval, the minimum distance between the sorted values of *term*, is also added to each observation of *term*.

After adjusting the location of *term* so that its minimum value is positive, creating *term**, automatic scaling will divide each observation of *term* by a power of ten. The exponent of this scaling factor is given by

$$p = \log_{10}\left\{\max(term^*) - \min(term^*)\right\}$$

$$p^* = \mathrm{sign}(p)\mathrm{floor}\left(|p|\right)$$

Rather than letting `fp` automatically choose the scaling of *term*, you may specify adjustment and scale factors a and b by using the `scale(`*a b*`)` option. Fractional powers are then calculated using the (*term*+*a*)/*b* values.

When `scale` or `scale(`*a b*`)` is specified, values in the variable *term* are not modified; `fp` merely remembers to scale the values whenever powers are calculated.

In addition to fp, both scale and scale(*a b*) may be used with fp generate.

You will probably not use scale(*a b*) with fp for values of *a* and *b* that you create yourself, although you could. As we demonstrated earlier, it is usually easier just to generate a scaled variable.

scale(*a b*) is useful when you previously fit a model using scale in one dataset and now want to create the fractional polynomials in another. In the first dataset, fp with scale added notes to the dataset concerning the values of *a* and *b*. You can see them by typing

```
. notes
```

You can then use fp generate, scale(*a b*) in the second dataset.

When you apply the scaling rules of a previously fit model to new data with the scale(*a b*) option, it is possible that the scaled term may have nonpositive values. fp will be unable to calculate the fractional powers of the term in this case and will issue an error.

The options zero and catzero cause fp and fp generate to output zero values for each fractional polynomial variable when the input (scaled) fractional polynomial variable is nonpositive. Specifying catzero causes a dummy variable indicating nonpositive values of the (scaled) fractional polynomial variable to be included in the model. A detailed example of the use of catzero and zero is shown in example 3 below.

Using the scaling options, we can fit our previous model again using the auto.dta. We specify scale(0 1000) so that fp will shrink the magnitude of weight in estimating the regression. This is done for demonstration purposes because our scaling rule is simple. As mentioned before, in practice, you would probably only use scale(*a b*) when applying the scaling rules from a previous analysis. Allowing fp to scale does have the advantage of not altering the original variable, weight.

```
. use https://www.stata-press.com/data/r18/auto, clear
(1978 automobile data)

. fp <weight>, fp(-.5) scale(0 1000): regress mpg <weight> foreign
-> regress mpg weight_1 foreign

      Source |       SS           df       MS      Number of obs   =        74
-------------+----------------------------------   F(2, 71)        =     79.51
       Model |  1689.20861         2  844.604307   Prob > F        =    0.0000
    Residual |  754.250846        71  10.6232514   R-squared       =    0.6913
-------------+----------------------------------   Adj R-squared   =    0.6826
       Total |  2443.45946        73  33.4720474   Root MSE        =    3.2593

------------------------------------------------------------------------------
         mpg | Coefficient  Std. err.      t    P>|t|     [95% conf. interval]
-------------+----------------------------------------------------------------
    weight_1 |   66.89665   6.021749    11.11   0.000     54.88963    78.90368
     foreign |  -2.095622   1.043513    -2.01   0.048    -4.176329   -.0149159
       _cons |  -17.58651   3.397992    -5.18   0.000    -24.36192   -10.81111
------------------------------------------------------------------------------
```

The scaling is clearly indicated in the variable notes for the generated variable weight_1.

```
. notes weight_1

weight_1:
  1.  fp term 1 of x^(-.5), where x is weight scaled.
  2.  Scaling was user specified: x = (weight+a)/b where a=0 and b=1000
  3.  Fractional polynomial variables created by fp <weight>, fp(-.5)
      scale(0 1000): regress mpg <weight> foreign
  4.  To re-create the fractional polynomial variables, for instance, in
      another dataset, type fp gen double weight^(-.5), scale(0 1000)
```

Centering

The fractional polynomial of *term*, centered on c is

$$\left(term^{(p_1,\ldots,p_m)} - c^{(p_1,\ldots,p_m)}\right)'\boldsymbol{\beta}$$

The intercept of a centered fractional polynomial can be interpreted as the effect at zero for all the covariates. When we center the fractional polynomial terms using c, the intercept is now interpreted as the effect at $term = c$ and zero values for the other covariates.

Suppose we wanted to center the fractional polynomial of x with powers (0, 0, 2) at $x = c$.

$$\begin{aligned}
&\left(x^{(0,0,2)} - c^{(0,0,2)}\right)'\boldsymbol{\beta} \\
&= \beta_0 + \beta_1\left(x^{(0)} - c^{(0)}\right) + \beta_2\left\{x^{(0)}\ln(x) - c^{(0)}\ln(c)\right\} + \beta_3\left(x^{(2)} - c^{(2)}\right) \\
&= \beta_0 + \beta_1\{\ln(x) - \ln(c)\} + \beta_2\left[\{\ln(x)\}^2 - \{\ln(c)\}^2\right] + \beta_3\left(x^2 - c^2\right)
\end{aligned}$$

When `center` is specified, `fp` centers based on the sample mean of (scaled) *term*. A previously chosen value for centering, c, may also be specified in `center(`*c*`)`. This would be done when applying the results of a previous model fitting to a new dataset.

The `center` and `center(`*c*`)` options may be used in `fp` or `fp generate`.

Returning to the model of mileage per gallon based on automobile weight and foreign origin, we refit the model with the fractional polynomial of `weight` centered at its scaled mean.

```
. use https://www.stata-press.com/data/r18/auto, clear
(1978 automobile data)

. fp <weight>, fp(-.5) scale(0 1000) center: regress mpg <weight> foreign
-> regress mpg weight_1 foreign

      Source |       SS           df       MS      Number of obs   =        74
-------------+----------------------------------   F(2, 71)        =     79.51
       Model |  1689.20861         2  844.604307   Prob > F        =    0.0000
    Residual |  754.250846        71  10.6232514   R-squared       =    0.6913
-------------+----------------------------------   Adj R-squared   =    0.6826
       Total |  2443.45946        73  33.4720474   Root MSE        =    3.2593

------------------------------------------------------------------------------
         mpg | Coefficient  Std. err.      t    P>|t|     [95% conf. interval]
-------------+----------------------------------------------------------------
    weight_1 |   66.89665   6.021749    11.11   0.000     54.88963    78.90368
     foreign |  -2.095622   1.043513    -2.01   0.048    -4.176329   -.0149159
       _cons |   20.91163   .4624143    45.22   0.000      19.9896    21.83366
------------------------------------------------------------------------------
```

Note that the coefficients for `weight_1` and `foreign` do not change. Only the intercept `_cons` changes. It can be interpreted as the estimated average miles per gallon of an American-made car of average weight.

Examples

▷ Example 1: Linear regression

Consider the serum immunoglobulin G (IgG) dataset from Isaacs et al. (1983), which consists of 298 independent observations in young children. The dependent variable `sqrtigg` is the square root of the IgG concentration, and the independent variable `age` is the age of each child. (Preliminary Box−Cox analysis shows that a square root transformation removes the skewness in IgG.)

The aim is to find a model that accurately predicts the mean of `sqrtigg` given `age`. We use `fp` to find the best FP2 model (the default option). We specify `center` for automatic centering. The age of each child is small in magnitude and positive, so we do not use the scaling options of `fp` or scale ourselves.

```
. use https://www.stata-press.com/data/r18/igg, clear
(Immunoglobulin in children)
. fp <age>, scale center: regress sqrtigg <age>
(fitting 44 models)
(....10%....20%....30%....40%....50%....60%....70%....80%....90%....100%)
Fractional polynomial comparisons:
```

age	Test df	Deviance	Residual std. dev.	Deviance diff.	P	Powers
omitted	4	427.539	0.497	108.090	0.000	
linear	3	337.561	0.428	18.113	0.000	1
m = 1	2	327.436	0.421	7.987	0.020	0
m = 2	0	319.448	0.416	0.000	--	-2 2

```
Note: Test df is degrees of freedom, and P = P > F is sig. level for tests
      comparing models vs. model with m = 2 based on deviance difference,
      F(df, 293).
```

Source	SS	df	MS
Model	22.2846976	2	11.1423488
Residual	50.9676492	295	.172771692
Total	73.2523469	297	.246640898

Number of obs	=	298
F(2, 295)	=	64.49
Prob > F	=	0.0000
R-squared	=	0.3042
Adj R-squared	=	0.2995
Root MSE	=	.41566

sqrtigg	Coefficient	Std. err.	t	P>|t|	[95% conf.	interval]
age_1	-.1562156	.027416	-5.70	0.000	-.2101713	-.10226
age_2	.0148405	.0027767	5.34	0.000	.0093757	.0203052
_cons	2.283145	.0305739	74.68	0.000	2.222974	2.343315

The new variables created by `fp` contain the best-fitting fractional polynomial powers of `age`, as centered by `fp`. For example, `age_1` is centered by subtracting the mean of `age` raised to the power -2.

The variables created by `fp` and `fp generate` are centered or scaled as specified by the user, which is reflected in the estimated regression coefficients and intercept. Centering does have its advantages (see *Centering* earlier in this entry). By default, `fp` will not perform scaling or centering. For a more detailed discussion, see Royston and Sauerbrei (2008, sec. 4.11).

The fitted curve has an asymmetric S shape. The best model has powers $(-2, 2)$ and deviance 319.448. We reject lesser degree models: the null, linear, and natural log power models at the 0.05 level. As many as 44 models have been fit in the search for the best powers. Now let's look at models of degree ≤ 4. The highest allowed degree is specified in `dimension()`. We overwrite the previously generated fractional polynomial power variables by including `replace`.

```
. fp <age>, dimension(4) center replace: regress sqrtigg <age>
(fitting 494 models)
(....10%....20%....30%....40%....50%....60%....70%....80%....90%....100%)
Fractional polynomial comparisons:
```

age	Test df	Deviance	Residual std. dev.	Deviance diff.	P	Powers
omitted	8	427.539	0.497	109.795	0.000	
linear	7	337.561	0.428	19.818	0.007	1
m = 1	6	327.436	0.421	9.692	0.149	0
m = 2	4	319.448	0.416	1.705	0.798	-2 2
m = 3	2	319.275	0.416	1.532	0.476	-2 1 1
m = 4	0	317.744	0.416	0.000	--	0 3 3 3

```
Note: Test df is degrees of freedom, and P = P > F is sig. level for tests
      comparing models vs. model with m = 4 based on deviance difference,
      F(df, 289).
```

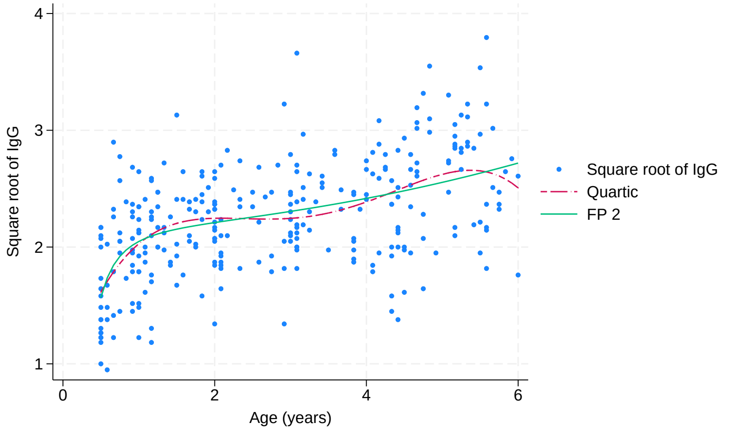
Source	SS	df	MS			
				Number of obs	=	298
				F(4, 293)	=	32.63
Model	22.5754541	4	5.64386353	Prob > F	=	0.0000
Residual	50.6768927	293	.172958678	R-squared	=	0.3082
				Adj R-squared	=	0.2987
Total	73.2523469	297	.246640898	Root MSE	=	.41588

sqrtigg	Coefficient	Std. err.	t	P>|t|	[95% conf.	interval]
age_1	.8761824	.1898721	4.61	0.000	.5024962	1.249869
age_2	-.1922029	.0684934	-2.81	0.005	-.3270044	-.0574015
age_3	.2043794	.074947	2.73	0.007	.0568767	.3518821
age_4	-.0560067	.0212969	-2.63	0.009	-.097921	-.0140924
_cons	2.238735	.0482705	46.38	0.000	2.143734	2.333736

It appears that the FP4 model is not significantly different from the other fractional polynomial models (at the 0.05 level).

Let's compare the curve shape from the $m = 2$ model with that from a conventional quartic polynomial whose fit turns out to be significantly better than a cubic (not shown). We use the ability of `fp` both to generate the required powers of `age`, namely, $(1, 2, 3, 4)$ for the quartic and $(-2, 2)$ for the second-degree fractional polynomial, and to fit the model. The `fp()` option is used to specify the powers. We use `predict` to obtain the fitted values of each regression. We fit both models with `fp` and graph the resulting curves with `twoway scatter`.

```
. fp <age>, center fp(1 2 3 4) replace: regress sqrtigg <age>
-> regress sqrtigg age_1 age_2 age_3 age_4

      Source |       SS           df       MS      Number of obs   =       298
-------------+----------------------------------   F(4, 293)       =     32.65
       Model |  22.5835458         4  5.64588646   Prob > F        =    0.0000
    Residual |   50.668801       293  .172931061   R-squared       =    0.3083
-------------+----------------------------------   Adj R-squared   =    0.2989
       Total |  73.2523469       297  .246640898   Root MSE        =    .41585

------------------------------------------------------------------------------
     sqrtigg | Coefficient  Std. err.      t    P>|t|     [95% conf. interval]
-------------+----------------------------------------------------------------
       age_1 |   2.047831   .4595962     4.46   0.000     1.143302    2.952359
       age_2 |  -1.058902   .2822803    -3.75   0.000    -1.614456   -.5033479
       age_3 |   .2284917   .0667591     3.42   0.001     .0971037    .3598798
       age_4 |  -.0168534   .0053321    -3.16   0.002    -.0273475   -.0063594
       _cons |   2.240012   .0480157    46.65   0.000     2.145512    2.334511
------------------------------------------------------------------------------

. predict fit1
(option xb assumed; fitted values)

. label variable fit1 "Quartic"

. fp <age>, center fp(-2 2) replace: regress sqrtigg <age>
-> regress sqrtigg age_1 age_2

      Source |       SS           df       MS      Number of obs   =       298
-------------+----------------------------------   F(2, 295)       =     64.49
       Model |  22.2846976         2  11.1423488   Prob > F        =    0.0000
    Residual |  50.9676492       295  .172771692   R-squared       =    0.3042
-------------+----------------------------------   Adj R-squared   =    0.2995
       Total |  73.2523469       297  .246640898   Root MSE        =    .41566

------------------------------------------------------------------------------
     sqrtigg | Coefficient  Std. err.      t    P>|t|     [95% conf. interval]
-------------+----------------------------------------------------------------
       age_1 |  -.1562156    .027416    -5.70   0.000    -.2101713    -.10226
       age_2 |   .0148405   .0027767     5.34   0.000     .0093757    .0203052
       _cons |   2.283145   .0305739    74.68   0.000     2.222974    2.343315
------------------------------------------------------------------------------

. predict fit2
(option xb assumed; fitted values)

. label variable fit2 "FP 2"

. scatter sqrtigg fit1 fit2 age, c(. l l) m(o i i) msize(small)
> lpattern(. -_.) ytitle("Square root of IgG") xtitle("Age (years)")
```



The quartic curve has an unsatisfactory wavy appearance that is implausible for the known behavior of IgG, the serum level of which increases throughout early life. The fractional polynomial curve (FP2) increases monotonically and is therefore biologically the more plausible curve. The two models have approximately the same deviance.

◁

▷ Example 2: Cox regression

Data from Smith et al. (1992) contain times to complete healing of leg ulcers in a randomized, controlled clinical trial of two treatments in 192 elderly patients. Several covariates were available, of which an important one is `mthson`, the number of months since the recorded onset of the ulcer. This time is recorded in whole months, not fractions of a month; therefore, some zero values are recorded.

Because the response variable is time to an event of interest and some (in fact, about one-half) of the times are censored, using Cox regression to analyze the data is appropriate. We consider fractional polynomials in `mthson`, adjusting for four other covariates: `age`; `ulcarea`, the area of tissue initially affected by the ulcer; `deepppg`, a binary variable indicating the presence or absence of deep vein involvement; and `treat`, a binary variable indicating treatment type.

We fit fractional polynomials of degrees 1 and 2 with `fp`. We specify `scale` to perform automatic scaling on `mthson`. This makes it positive and ensures that its magnitude is not too large. (See *Scaling* for more details.) The display option `nohr` is specified before the colon so that the coefficients and not the hazard ratios are displayed.

The `center` option is specified to obtain automatic centering. `age` and `ulcarea` are also demeaned by using `summarize` and then subtracting the returned result `r(mean)`.

In Cox regression, there is no constant term, so we cannot see the effects of centering in the table of regression estimates. The effects would be present if we were to graph the baseline hazard or survival function because these functions are defined with all predictors set equal to 0.

In these graphs, we will see the estimated baseline hazard or survival function under no deep vein involvement or treatment and under mean age, ulcer area, and number of months since the recorded onset of the ulcer.

```
. use https://www.stata-press.com/data/r18/legulcer2, clear
(Leg ulcer clinical trial)

. stset ttevent, fail(healed)

Survival-time data settings

         Failure event: healed!=0 & healed<.
Observed time interval: (0, ttevent]
     Exit on or before: failure

--------------------------------------------------------------------------
        192  total observations
          0  exclusions
--------------------------------------------------------------------------
        192  observations remaining, representing
         92  failures in single-record/single-failure data
     13,825  total analysis time at risk and under observation
                                                At risk from t =         0
                                     Earliest observed entry t =         0
                                          Last observed exit t =       206

. quietly sum age

. replace age = age - r(mean)
variable age was byte now float
(192 real changes made)

. quietly sum ulcarea

. replace ulcarea = ulcarea - r(mean)
variable ulcarea was int now float
(192 real changes made)

. fp <mthson>, center scale nohr: stcox <mthson> age ulcarea deepppg treat
(fitting 44 models)
(....10%....20%....30%....40%....50%....60%....70%....80%....90%....100%)

Fractional polynomial comparisons:
-------------------------------------------------------------------
             |   Test             Deviance
      mthson |    df    Deviance     diff.         P    Powers
-------------+-----------------------------------------------------
     omitted |     4     754.345    17.636     0.001
      linear |     3     751.680    14.971     0.002    1
       m = 1 |     2     738.969     2.260     0.323    -.5
       m = 2 |     0     736.709     0.000        --    .5 .5
-------------------------------------------------------------------
Note: Test df is degrees of freedom, and P = P > chi2 is sig. level
      for tests comparing models vs. model with m = 2 based on
      deviance difference, chi2.

Cox regression with Breslow method for ties

No. of subjects =    192                            Number of obs =    192
No. of failures =     92
Time at risk    = 13,825
                                                    LR chi2(6)    = 108.59
Log likelihood = -368.35446                         Prob > chi2   = 0.0000

------------------------------------------------------------------------------
          _t | Coefficient  Std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
    mthson_1 |   -2.81425   .6996385    -4.02   0.000    -4.185516   -1.442984
    mthson_2 |   1.541451   .4703143     3.28   0.001     .6196521     2.46325
         age |  -.0261111   .0087983    -2.97   0.003    -.0433556   -.0088667
     ulcarea |  -.0017491    .000359    -4.87   0.000    -.0024527   -.0010455
     deepppg |  -.5850499   .2163173    -2.70   0.007    -1.009024   -.1610758
       treat |  -.1624663   .2171048    -0.75   0.454    -.5879838    .2630513
------------------------------------------------------------------------------
```

The best-fitting fractional polynomial of degree 2 has powers $(0.5, 0.5)$ and deviance 736.709. However, this model does not fit significantly better than the fractional polynomial of degree 1 (at the 0.05 level), which has power -0.5 and deviance 738.969. We prefer the model with $m = 1$.

```
. fp <mthson>, replace center scale nohr fp(-.5): stcox <mthson> age ulcarea
> deepppg treat
-> stcox mthson_1 age ulcarea deepppg treat
Cox regression with Breslow method for ties

No. of subjects =     192                              Number of obs =    192
No. of failures =      92
Time at risk    = 13,825
                                                       LR chi2(5)    = 106.33
Log likelihood = -369.48426                            Prob > chi2   = 0.0000
```

_t	Coefficient	Std. err.	z	P>|z|	[95% conf. interval]	
mthson_1	.1985592	.0493922	4.02	0.000	.1017523	.2953662
age	-.02691	.0087875	-3.06	0.002	-.0441331	-.0096868
ulcarea	-.0017416	.0003482	-5.00	0.000	-.0024241	-.0010591
deepppg	-.5740759	.2185134	-2.63	0.009	-1.002354	-.1457975
treat	-.1798575	.2175726	-0.83	0.408	-.6062921	.246577

The hazard for healing is much higher for patients whose ulcer is of recent onset than for those who have had an ulcer for many months.

A more appropriate analysis of this dataset, if one wanted to model all the predictors, possibly with fractional polynomial functions, would be to use `mfp`; see [R] **mfp**.

◁

▷ Example 3: Logistic regression

The `zero` option permits fitting a fractional polynomial model to the positive values of a covariate, taking nonpositive values as zero. An application is the assessment of the effect of cigarette smoking as a risk factor. Whitehall 1 is an epidemiological study, which was examined in Royston and Sauerbrei (2008), of 18,403 male British Civil Servants employed in London. We examine the data collected in Whitehall 1 and use logistic regression to model the odds of death based on a fractional polynomial in the number of cigarettes smoked.

Nonsmokers may be qualitatively different from smokers, so the effect of smoking (regarded as a continuous variable) may not be continuous between zero cigarettes and one cigarette. To allow for this possibility, we model the risk as a constant for the nonsmokers and as a fractional polynomial function of the number of cigarettes for the smokers, adjusted for age.

The dependent variable `all10` is an indicator of whether the individual passed away in the 10 years under study. `cigs` is the number of cigarettes consumed per day. After loading the data, we demean `age` and create a dummy variable, `nonsmoker`. We then use `fp` to fit the model.

```
. use https://www.stata-press.com/data/r18/smoking, clear
(Smoking and mortality data)

. quietly sum age

. replace age = age - r(mean)
variable age was byte now float
(17,260 real changes made)

. generate byte nonsmoker = cond(cigs==0, 1, 0) if cigs < .

. fp <cigs>, zero: logit all10 <cigs> nonsmoker age
(fitting 44 models)
(....10%....20%....30%....40%....50%....60%....70%....80%....90%....100%)

Fractional polynomial comparisons:

-------------------------------------------------------------------
             |   Test                Deviance
        cigs |    df    Deviance        diff.        P    Powers
-------------+-----------------------------------------------------
     omitted |     4    9990.804       46.096    0.000
      linear |     3    9958.801       14.093    0.003    1
       m = 1 |     2    9946.603        1.895    0.388    0
       m = 2 |     0    9944.708        0.000       --    -1 -1
-------------------------------------------------------------------
Note: Test df is degrees of freedom, and P = P > chi2 is sig. level
      for tests comparing models vs. model with m = 2 based on
      deviance difference, chi2.

Logistic regression                                    Number of obs =  17,260
                                                       LR chi2(4)    = 1029.03
                                                       Prob > chi2   =  0.0000
Log likelihood = -4972.3539                            Pseudo R2     =  0.0938

------------------------------------------------------------------------------
       all10 | Coefficient  Std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
      cigs_1 |  -1.285867   .3358483    -3.83   0.000    -1.944117   -.6276162
      cigs_2 |  -1.982424    .572109    -3.47   0.001    -3.103736   -.8611106
   nonsmoker |  -1.223749   .1119583   -10.93   0.000    -1.443183   -1.004315
         age |   .1194541   .0045818    26.07   0.000     .1104739    .1284343
       _cons |  -1.591489   .1052078   -15.13   0.000    -1.797693   -1.385286
------------------------------------------------------------------------------
```

Omission of the `zero` option would cause `fp` to halt with an error message because nonpositive covariate values (for example, values of `cigs`) are invalid unless the `scale` option is specified.

A closely related approach involves the `catzero` option. Here we no longer need to have `nonsmoker` in the model, because `fp` creates its own dummy variable `cigs_0` to indicate whether the individual does not smoke on that day.

```
. fp <cigs>, catzero replace: logit all10 <cigs> age
(fitting 44 models)
(....10%....20%....30%....40%....50%....60%....70%....80%....90%....100%)
Fractional polynomial comparisons:

         cigs    Test             Deviance
                   df   Deviance     diff.        P   Powers

      omitted       5   10175.75   231.047    0.000
       linear       3    9958.80    14.093    0.003   1
        m = 1       2    9946.60     1.895    0.388   0
        m = 2       0    9944.71     0.000       --   -1 -1

Note: Test df is degrees of freedom, and P = P > chi2 is sig. level
      for tests comparing models vs. model with m = 2 based on
      deviance difference, chi2.
Logistic regression                                 Number of obs =  17,260
                                                    LR chi2(4)    = 1029.03
                                                    Prob > chi2   =  0.0000
Log likelihood = -4972.3539                         Pseudo R2     =  0.0938

        all10   Coefficient  Std. err.      z    P>|z|     [95% conf. interval]

       cigs_0     -1.223749   .1119583  -10.93   0.000    -1.443183   -1.004315
       cigs_1     -1.285867   .3358483   -3.83   0.000    -1.944117   -.6276162
       cigs_2     -1.982424    .572109   -3.47   0.001    -3.103736   -.8611106
          age      .1194541   .0045818   26.07   0.000     .1104739    .1284343
        _cons     -1.591489   .1052078  -15.13   0.000    -1.797693   -1.385286
```

Under both approaches, the comparison table suggests that we can accept the FP1 model instead of the FP2 model. We estimate the parameters of the accepted model—that is, the one that uses the natural logarithm of `cigs`—with `fp`.

```
. fp <cigs>, catzero replace fp(0): logit all10 <cigs> age
-> logit all10 cigs_0 cigs_1 age
Logistic regression                                 Number of obs =  17,260
                                                    LR chi2(3)    = 1027.13
                                                    Prob > chi2   =  0.0000
Log likelihood = -4973.3016                         Pseudo R2     =  0.0936

        all10   Coefficient  Std. err.      z    P>|z|     [95% conf. interval]

       cigs_0      .1883732   .1553093    1.21   0.225    -.1160274    .4927738
       cigs_1      .3469842   .0543552    6.38   0.000     .2404499    .4535185
          age      .1194976   .0045818   26.08   0.000     .1105174    .1284778
        _cons     -3.003767   .1514909  -19.83   0.000    -3.300683    -2.70685
```

The high p-value for `cigs_0` in the output indicates that we cannot reject that there is no extra effect at zero for nonsmokers.

◁

Stored results

In addition to the results that *est_cmd* stores, `fp` stores the following in `e()`:

Scalars

`e(fp_dimension)`	degree of fractional polynomial
`e(fp_center_mean)`	value used for centering or .
`e(fp_scale_a)`	value used for scaling or .
`e(fp_scale_b)`	value used for scaling or .
`e(fp_compare_df2)`	denominator degree of freedom in F test

Macros

`e(fp_cmd)`	`fp, search():` or `fp, powers():`
`e(fp_cmdline)`	full `fp` command as typed
`e(fp_variable)`	fractional polynomial variable
`e(fp_terms)`	generated `fp` variables
`e(fp_gen_cmdline)`	`fp generate` command to re-create `e(fp_terms)` variables
`e(fp_catzero)`	`catzero`, if specified
`e(fp_zero)`	`zero`, if specified
`e(fp_compare_type)`	`F` or `chi2`

Matrices

`e(fp_fp)`	powers used in fractional polynomial
`e(fp_compare)`	results of model comparisons
`e(fp_compare_stat)`	F test statistics
`e(fp_compare_df1)`	`chi2` degrees of freedom or numerator degrees of freedom of F test
`e(fp_compare_fp)`	powers of comparison models
`e(fp_compare_length)`	encoded string for display of row titles
`e(fp_powers)`	powers that are searched

`fp generate` stores the following in `r()`:

Scalars

`r(fp_center_mean)`	value used for centering or .
`r(fp_scale_a)`	value used for scaling or .
`r(fp_scale_b)`	value used for scaling or .

Macros

`r(fp_cmdline)`	full `fp generate` command as typed
`r(fp_variable)`	fractional polynomial variable
`r(fp_terms)`	generated `fp` variables
`r(fp_catzero)`	`catzero`, if specified
`r(fp_zero)`	`zero`, if specified

Matrices

`r(fp_fp)`	powers used in fractional polynomial

Methods and formulas

The general definition of a fractional polynomial, accommodating possible repeated powers, may be written for functions $H_1(x), \ldots, H_m(x)$ of $x > 0$ as

$$\beta_0 + \sum_{j=1}^{m} \beta_j H_j(x)$$

where $H_1(x) = x^{(p_1)}$ and for $j = 2, \ldots, m$,

$$H_j(x) = \begin{cases} x^{(p_j)} & \text{if } p_j \neq p_{j-1} \\ H_{j-1}(x)\ln(x) & \text{if } p_j = p_{j-1} \end{cases}$$

For example, a fractional polynomial of degree 3 with powers $(1, 3, 3)$ has $H_1(x) = x$, $H_2(x) = x^3$, and $H_3(x) = x^3 \ln(x)$ and equals $\beta_0 + \beta_1 x + \beta_2 x^3 + \beta_3 x^3 \ln(x)$.

We can express a fractional polynomial in vector notation by using $\mathbf{H}(x) = [H_1(x), \ldots, H_d(x)]'$. We define $x^{(p_1,p_2,\ldots,p_m)} = [\mathbf{H}(x)', 1]'$. Under this notation, we can write

$$x^{(1,3,3)\prime}\boldsymbol{\beta} = \beta_0 + \beta_1 x + \beta_2 x^3 + \beta_3 x^3 \ln(x)$$

The fractional polynomial may be centered so that the intercept can be more easily interpreted. When centering the fractional polynomial of x at c, we subtract $c^{(p_1,p_2,\ldots,p_m)}$ from $x^{(p_1,p_2,\ldots,p_m)}$, where $c^{(p_1,p_2,\ldots,p_d)} = [\mathbf{H}(x)', 0]'$. The centered fractional polynomial is

$$\left(x^{(p_1,\ldots,p_d)} - c^{(p_1,\ldots,p_d)}\right)'\boldsymbol{\beta}$$

The definition may be extended to allow $x \leq 0$ values. For these values, the fractional polynomial is equal to the intercept β_0 or equal to a zero-offset term α_0 plus the intercept β_0.

The deviance D of a model is defined as -2 times its maximized log likelihood. For normal error models, we use the formula

$$D = n\left(1 - \bar{l} + \ln \frac{2\pi \text{RSS}}{n}\right)$$

where n is the sample size, $\bar{l}$ is the mean of the lognormalized weights ($\bar{l} = 0$ if the weights are all equal), and RSS is the residual sum of squares as fit by `regress`.

When `fp` is used to search for the best combination of powers, it reports a table comparing fractional polynomial models of degree $k < m$ with the degree m fractional polynomial model, which will have the lowest deviance. The comparison table also includes the linear model, in which <*term*> is not raised to a power, and the null model, in which <*term*> is omitted.

The `Test df` column of the model comparison table does not correspond to the model degrees of freedom for the individual models but rather to the degrees of freedom of a test comparing that model with the model with the lowest deviance. For normal error models, this is the numerator degrees of freedom of a partial F test; for other models, it is the degrees of freedom of the likelihood-ratio χ^2 test. When calculating the test degrees of freedom, the command accounts for the two types of parameters that are being estimated by `fp`: coefficients (β_j) and powers. Because the powers in a fractional polynomial are chosen from a finite set rather than from the entire real line, the degrees of freedom defined in this way are approximate and generally yield somewhat conservative tests (Royston and Altman 1994).

The p-values reported by `fp` are calculated differently for normal error models than for other models. Let D_k and D_m be the deviances of the models with degrees k and m, respectively. For normal error models, a variance ratio F is calculated as

$$F = \frac{d_2}{d_1}\left\{\exp\left(\frac{D_k - D_m}{n}\right) - 1\right\}$$

where d_1 is the numerator df, the number of additional parameters estimated by the degree m model over the degree k model. d_2 is the denominator degrees of freedom and equals the residual degrees of freedom of the degree m model minus the number of powers estimated, m. The p-value is obtained by referring F to an F distribution on (d_1, d_2) df.

For nonnormal models, the p-value is obtained by referring $D_k - D_m$ to a χ^2 distribution on d_1 degrees of freedom, with d_1 defined as above.

Acknowledgment

We thank Patrick Royston of the MRC Clinical Trials Unit, London, and coauthor of the Stata Press book *Flexible Parametric Survival Analysis Using Stata: Beyond the Cox Model* for writing `fracpoly` and `fracgen`, the commands on which `fp` and `fp generate` are based. We also thank Professor Royston for his advice on and review of the new `fp` commands.

References

Isaacs, D., D. G. Altman, C. E. Tidmarsh, H. B. Valman, and A. D. Webster. 1983. Serum immunoglobulin concentrations in preschool children measured by laser nephelometry: Reference ranges for IgG, IgA, IgM. *Journal of Clinical Pathology* 36: 1193–1196. https://doi.org/10.1136/jcp.36.10.1193.

Libois, F., and V. Verardi. 2013. Semiparametric fixed-effects estimator. *Stata Journal* 13: 329–336.

Royston, P., and D. G. Altman. 1994. Regression using fractional polynomials of continuous covariates: Parsimonious parametric modelling. *Applied Statistics* 43: 429–467. https://doi.org/10.2307/2986270.

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.

Smith, J. M., C. J. Dore, A. Charlett, and J. D. Lewis. 1992. A randomized trial of Biofilm dressing for venous leg ulcers. *Phlebology* 7: 108–113. https://doi.org/10.1177/026835559200700307.

Also see

[R] **fp postestimation** — Postestimation tools for fp

[R] **mfp** — Multivariable fractional polynomial models

[U] **20 Estimation and postestimation commands**