

DETECTING CURVILINEARITY (continued)
 CONDITIONAL EFFECTS PLOTS

More on use of X^2 terms to detect curvilinearity: As we have said, a quick way to detect curvilinearity in the relationship between a dependent variable and a predictor variable is to generate a new variable that is the square of the predictor and then use both the predictor and its square in a regression. The following illustrates the use of this approach in two cases using *simulated* data:

Y is related to e^X
 Y is related to X, X^2 and X^3

Case 1. $y_1 = \exp x + \text{err}$ where $\text{err} \sim N(0,2)$ $x = 1/12, 2/12, 3/12, \dots$

y_1 was generated as follows:

```
. * generate the set of x values
. set obs 100
. gen x= _n
. replace x = x/12

. * generate a normally distributed error term with mean=0, sd=1
. gen err = uniform()
. replace err = invnorm(err)

* transform err so that is has sd=2
. replace err = 2*err
. gen y1= exp(x) + err

. * generate the square of x
. gen xsq = x*x
```

I then regressed y_1 first on just x and then on x and x^2

```
. regress y1 x
```

Source	SS	df	MS	Number of obs =	100
Model	46943534.0	1	46943534.0	F(1, 98) =	118.66
Residual	38770847.9	98	395620.897	Prob > F =	0.0000
				R-squared =	0.5477
				Adj R-squared =	0.5431
Total	85714381.8	99	865801.837	Root MSE =	628.98

y1	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
x	284.8272	26.14768	10.893	0.000	232.938	336.7164
_cons	-678.589	126.7462	-5.354	0.000	-930.1128	-427.0653

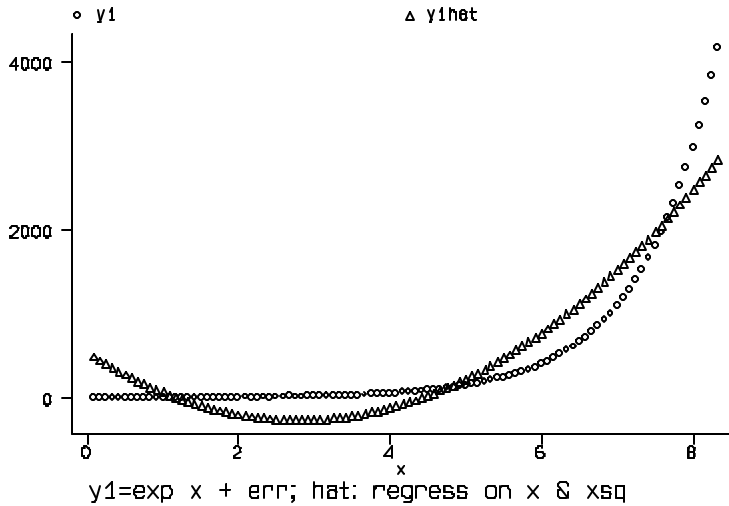
```
. regress y1 x xsq
```

Source	SS	df	MS	Number of obs =	100
Model	74610617.4	2	37305308.7	F(2, 97) =	326.89
Residual	11069765.5	97	114121.293	Prob > F =	0.0000
				R-squared =	0.8708
				Adj R-squared =	0.8681
Total	85680382.8	99	865458.413	Root MSE =	337.82

y	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
x	-570.8072	56.71165	-10.065	0.000	-683.3641 -458.2503
xsq	101.6573	6.528158	15.572	0.000	88.70068 114.6139
_cons	533.7508	103.4068	5.162	0.000	328.517 738.9846

```
. predict y1hat
```

```
. graph y1 y1hat x, connect(.1) xlabel ylabel title(y1=exp x + err; hat: regress on x xsq)
```



Clearly, the use of the x^2 term detects curvilinearity in the data. The method is not as good as transforming the data. Since $y1$ is basically e^x (plus an error term), we could have used $\log(y1)$.

Two things to try: ladder $y1$ you'll find that there is no transformation that fits well
 generate $\log y1 = \log(y1)$ and plot it against x it's quite linear

Conclusion: use of the x^2 term is a rough way of looking for curvilinearity.

Case 2. $y2$ is related to x , x^2 and x^3

```
. gen xcube = xsq*x
. gen y2=.7 +.3*x +.2*xsq+.1*xcube + err
. predict y2hat
```

. regress y2 x

Source	SS	df	MS	Number of obs =	100
Model	40411.0963	1	40411.0963	F(1, 98) =	624.70
Residual	6339.51396	98	64.688918	Prob > F =	0.0000
				R-squared =	0.8644
				Adj R-squared =	0.8630
				Root MSE =	8.0429
Total	46750.6103	99	472.228387		

y2	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
x	8.356875	.3343555	24.994	0.000	7.693357	9.020393
_cons	-13.88603	1.620729	-8.568	0.000	-17.10232	-10.66975

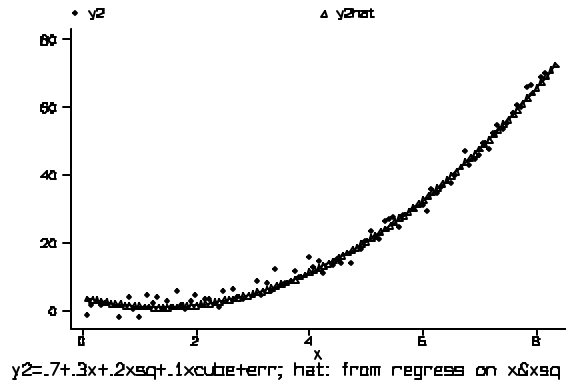
. regress y2 x xsq

Source	SS	df	MS	Number of obs =	100
Model	46251.9306	2	23125.9653	F(2, 97) =	4498.32
Residual	498.67963	97	5.14102711	Prob > F =	0.0000
				R-squared =	0.9893
				Adj R-squared =	0.9891
				Root MSE =	2.2674
Total	46750.6103	99	472.228387		

y2	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
x	-4.07352	.3806395	-10.702	0.000	-4.828984	-3.318056
xsq	1.476879	.043816	33.706	0.000	1.389916	1.563841
_cons	3.72369	.6940497	5.365	0.000	2.346194	5.101187

. predict y2hat

. graph y2 y2hat x, xlabel ylabel connect(.1) title(y2=.7+.3x+.2xsq+.1xcube+err; hat: from regress on x&xsq)



Here, once again, the use of the square term helped us detect the curvilinearity even before we graphed the

relationship.

Turning to real data:

```
. use nations
(Data on 109 countries)
. graph chldmort birth
. regress chldmort birth          (results not shown)
. gen birthsq = birth*birth

. regress chldmort birth birthsq
```

Source	SS	df	MS			
Model	9298.18113	2	4649.09057	Number of obs =	109	
Residual	4325.67208	106	40.8082271	F(2, 106) =	113.93	
Total	13623.8532	108	126.146789	Prob > F =	0.0000	
				R-squared =	0.6825	
				Adj R-squared =	0.6765	
				Root MSE =	6.3881	

chldmort	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
birth	-.7255742	.270928	-2.678	0.009	-1.262715	-.1884332
birthsq	.0221249	.0043273	5.113	0.000	.0135455	.0307042
_cons	5.892222	3.676548	1.603	0.112	-1.396892	13.18134

Another regression demonstrated that the relationship between chldmort and food was also curvilinear. Therefore both birth and food *and their squares* were included in the next regression:

```
. gen foodsq = food*food
. regress chldmort birth birthsq food foodsq
```

Source	SS	df	MS			
Model	10185.883	4	2546.47074	Number of obs =	108	
Residual	3387.99668	103	32.8931716	F(4, 103) =	77.42	
Total	13573.8796	107	126.858688	Prob > F =	0.0000	
				R-squared =	0.7504	
				Adj R-squared =	0.7407	
				Root MSE =	5.7353	

chldmort	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
birth	-.7179323	.2688836	-2.670	0.009	-1.2512	-.1846651
birthsq	.0186716	.0041746	4.473	0.000	.0103922	.0269509
food	-.0416276	.0106449	-3.911	0.000	-.0627391	-.020516
foodsq	6.48e-06	1.93e-06	3.354	0.001	2.65e-06	.0000103
_cons	72.67194	14.326	5.073	0.000	44.25971	101.0842

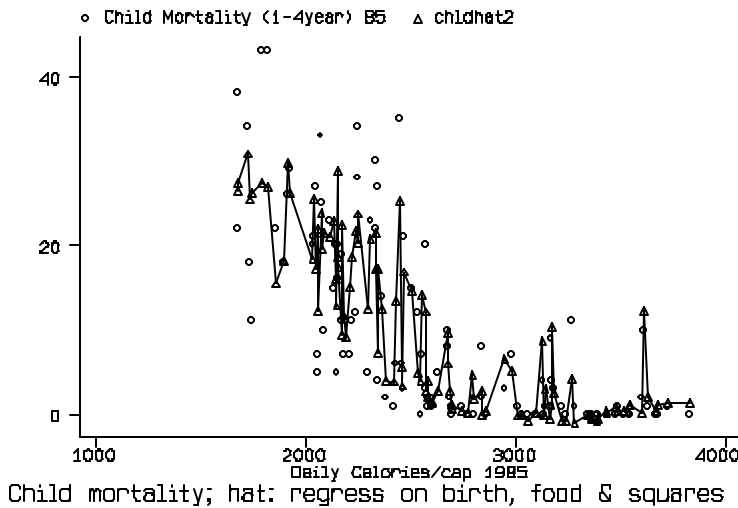
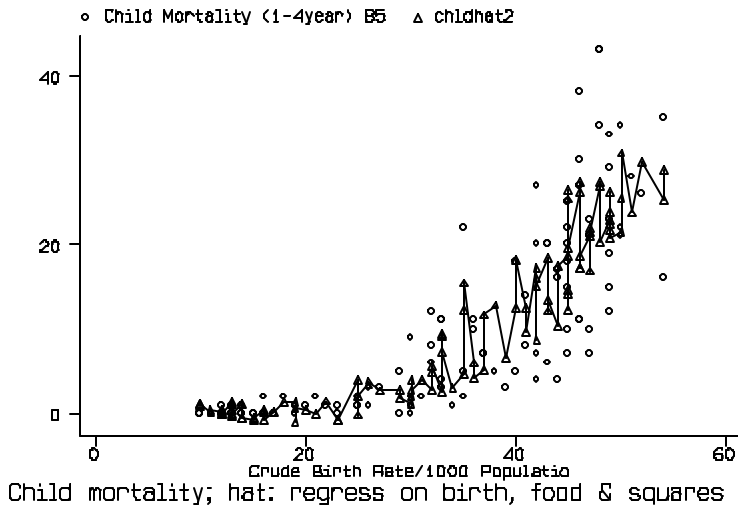
They are *all* significant.

```
. predict chldhat2
. sort birth
```

```
. graph chldmo chldhat2 birth, connect(.1) xlabel ylabel title(Child mortality; hat:
regress on birth, food & squares)
```

We can graph in the same way against food:

```
. sort food
. graph chldmo chldhat2 food, connect(.1) xlabel ylabel title(Child mortality; hat:
regress on birth, food & squares)
```



These connected lines - the predicted values - jump around because they are affected by the value of the predictor we didn't include -- in the first case, we graphed against birth and omitted food; in the second, we graphed against food and omitted birth.

CONDITIONAL EFFECTS PLOTS

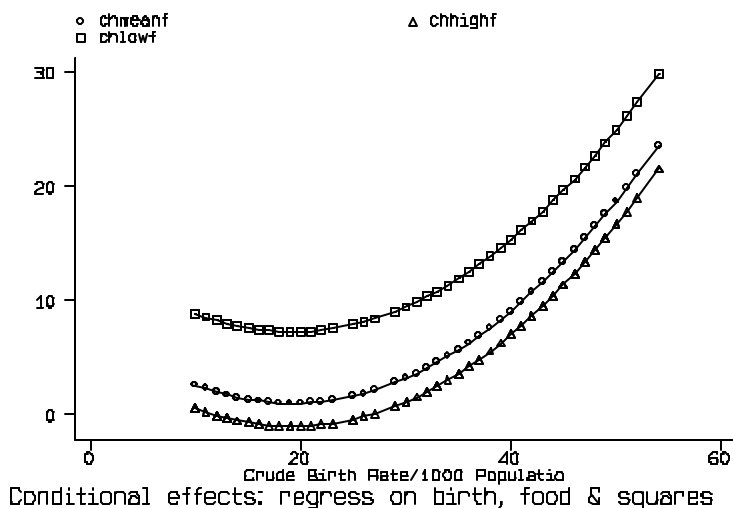
One way to see the predicted relationship between the dependent variable and a particular predictor is to make a set of predictions *when all the variables except that predictor are set to a constant value* - say their mean, or mean +1sd. For example, we can look at the predicted relationship between child mortality and birth when food is constant at approximately its mean, at approximately 1 sd above the mean, at approximately 1 sd below the mean.

```
. sum food birth
```

Variable	Obs	Mean	Std. Dev.	Min	Max
food	108	2648.648	569.4339	1678	3831
birth	109	32.78899	13.63416	10	54

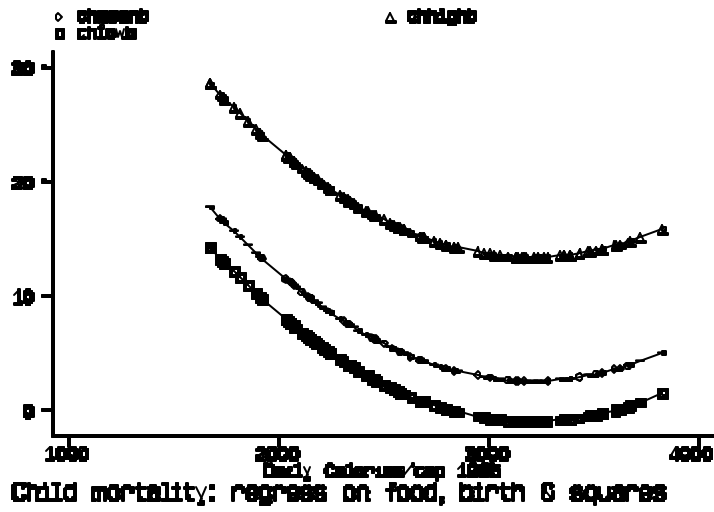
We'll look at the predicted values of child mortality in relation to birth when food = 2650 and 2650-570=2080 and 2650+570=3220. These will be **3 distinct curves**. STATA offers a way to "pick up" the coefficients found in the regression. Again, this procedure will use the coefficient from the *last* regress command that was issued:

```
. gen chmeanf = _b[_cons] + _b[birth]*birth + _b[birthsq] *birthsq + _b[food] * 2650 +
_b[foodsq] *2650 * 2650
. gen chlowf = _b[_cons] + _b[birth]*birth + _b[birthsq] *birthsq + _b[food] * 2080 +
_b[foodsq] *2080 * 2080
. gen chhighf = _b[_cons] + _b[birth]*birth + _b[birthsq] *birthsq + _b[food] * 3220 +
_b[foodsq] *3220 * 3220
```



The same approach can be taken plotting child mortality against food, holding birth constant at it's mean, 33, and mean+1 sd = 47, and mean - 1 sd = 19:

```
. gen chmeanb = _b[_cons] + _b[birth]*33 + _b[birthsq] *33*33 + _b[food] * food +
_b[foodsq] *foodsq
. gen chhighb = _b[_cons] + _b[birth]*47 + _b[birthsq] *47*47 + _b[food] * food +
_b[foodsq] *foodsq
. gen chlowb = _b[_cons] + _b[birth]*19 + _b[birthsq] *19*19 + _b[food] * food +
_b[foodsq] *foodsq
```



These plots are actually most interesting when there are interaction terms included, because then the shapes of the curves may be quite different.

Think about: how are these related to the curves we did for dummy variables?