Nonlinearity in Regression

Paul E. Johnson[1] [2]

[1]Department of Political Science

[2]Center for Research Methods and Data Analysis, University of Kansas

Trimmed Down for 2015!

Outline

Outline

Are Relationships "Really" Linear?

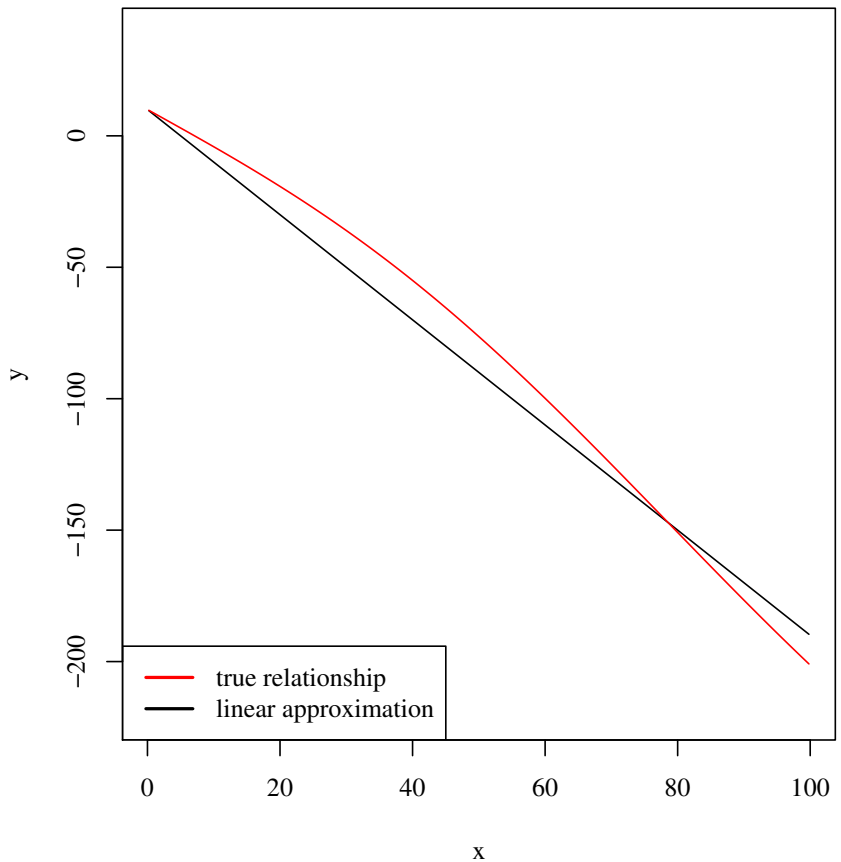
- Models (so far):

$$y_i = b_0 + b_1 x_i + e_i$$

- Maybe that's good enough
  - Occam's razor (use simplest model first)
- Mathematically (from Calculus, see Taylor's theorem) we may add complexity in stages, inserting x_i^2, x_i^3, to "bend" a line.

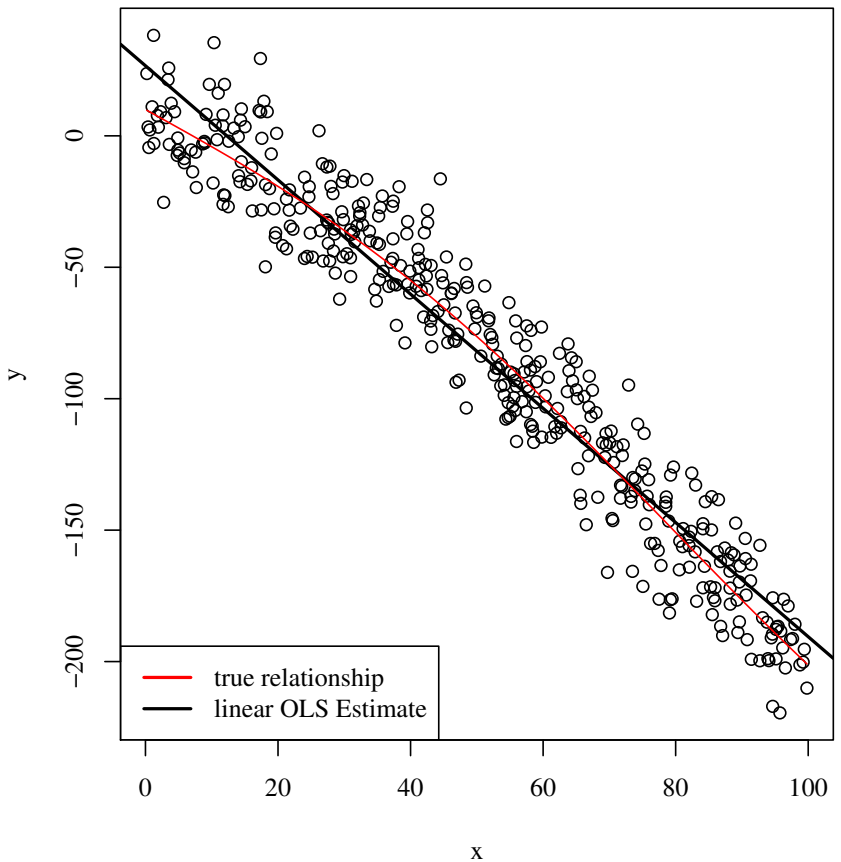
The "True" Model is Approximately Linear

- The relationship is almost linear
- Predictions won't be much different
- Slope of line in that region won't be much different



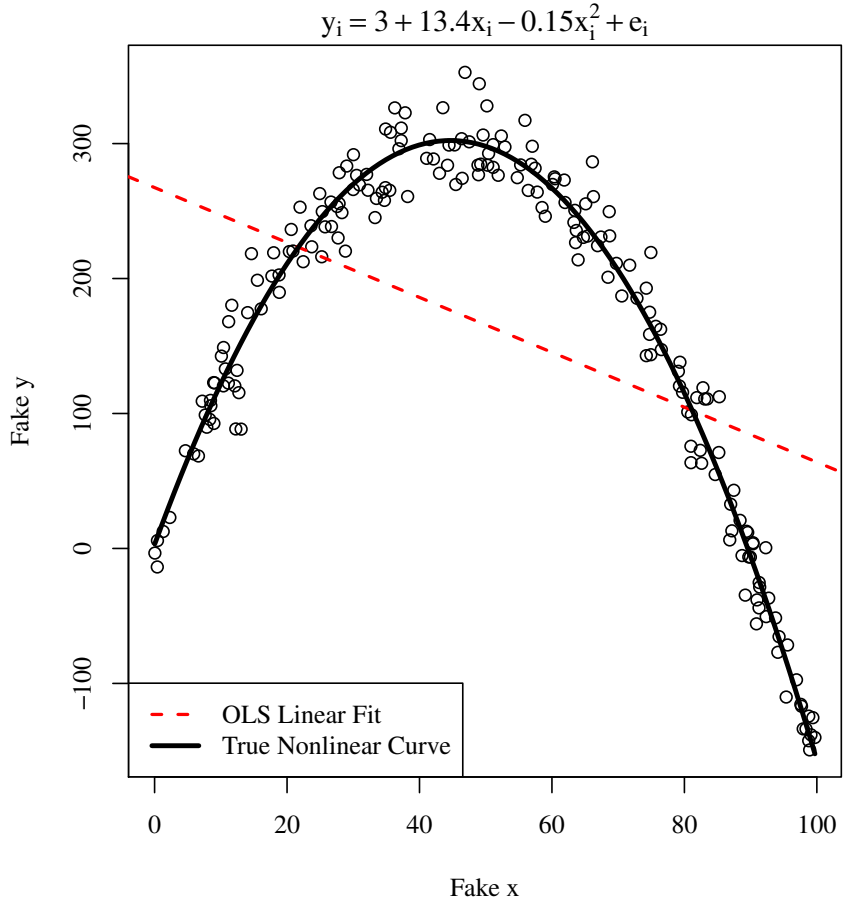
The "True" Model is Approximately Linear

- After error gets thrown into the model, its hard to see how there's much practical difference between the predictions of the "approximate" model and the true model.



The Straight Line Model Is Not Useful If...

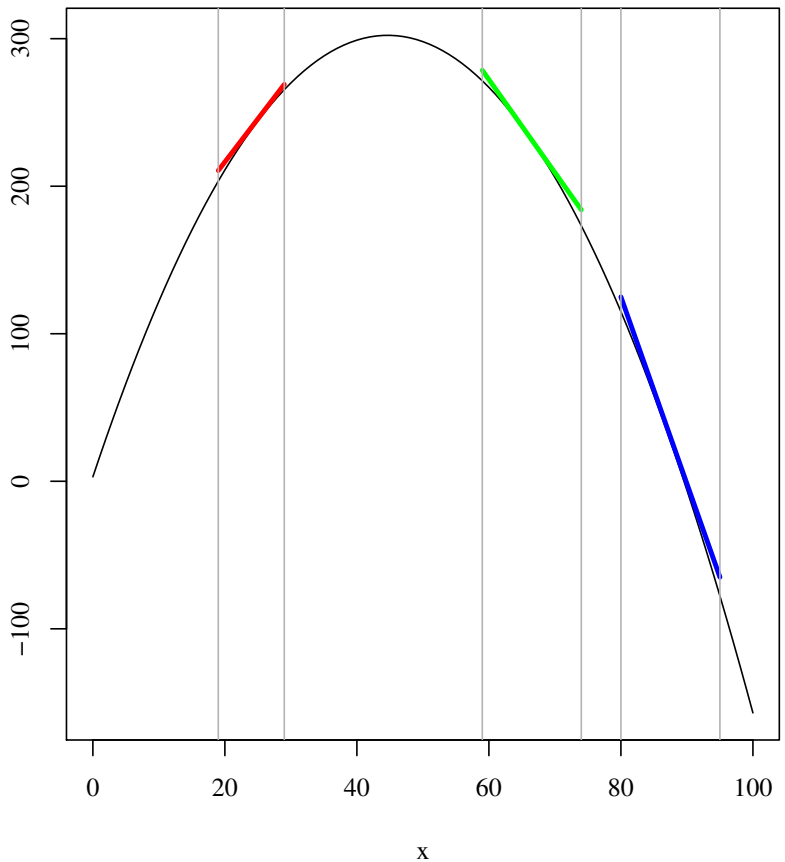
- Perhaps a picture really is worth 1000 words



Outline

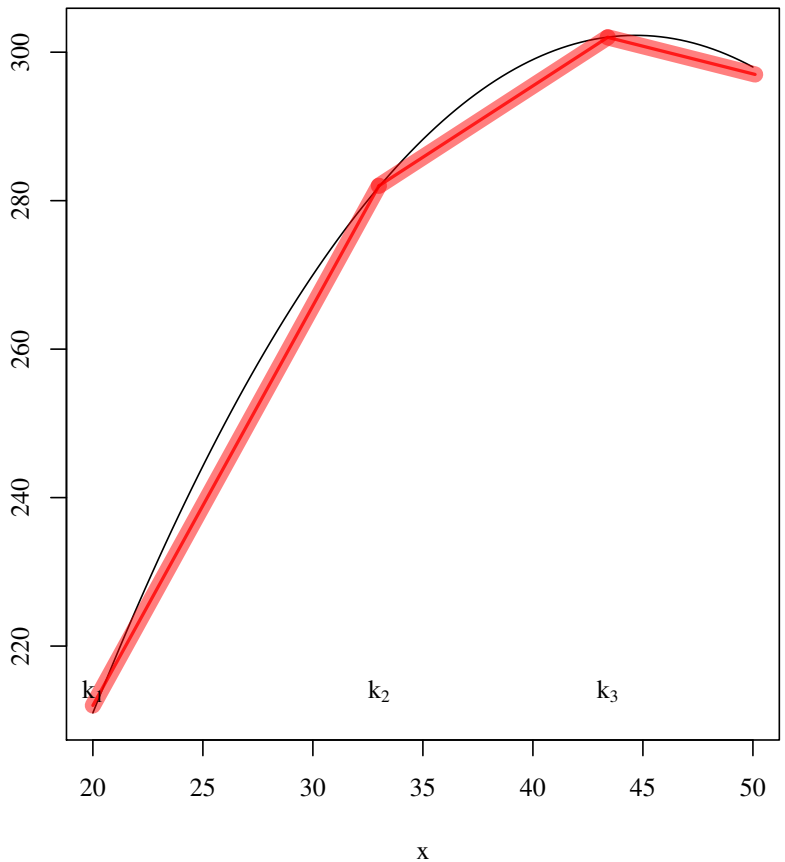
Segments

- A straight line will approximate any small part of the big curve
- If data is observed only in a narrow range, a straight line might be sufficient.



A Linear Spline Model

- Choose "knot" points, k_1, k_2, k_3
- G.E.P. Box, "Essentially, all models are wrong, but some are useful."



Estimating a Spline Model

- Write out the linear predictor, formula will look something like

$$b_0 + b_1 x_i + b_2(x_i - k_2) + b_3(x_i - k_3) \quad (1)$$
$$= b_0 + b_1 x_i + b_2 x2_i + b_3 x3_i \quad (2)$$

- Essentially, one manufactures "before & after dummy variables" and creates 3 columns for variants of x

x	x2	x3
13	0	0
14	0	0
15	1	0
16	2	0
17	3	1
18	4	2

- The b_2 and b_3 are "slope shifts: the effect of x_i "jumps"

Splines

- Handy!
  - Relative easy to code up
  - Will usually reveal "very bendy" relationships
- Problems
  - The "jagged" edges are theoretically bothersome (preferred "smooth" curves)
  - Assumes knots are at known points
    - Estimating the knots is a challenging problem.

Outline

Can Be Estimated With OLS

- Re-conceptualize "columns" of predictors.
- Might replace x_i with $\log(x_i)$

$$y_i = b_0 + b_1 log(x_i) + e_i \tag{3}$$

- "throw in" x squared (or one of many transformed x columns)

$$y_i = b_0 + b_1 x_i + b_2 x_i^2 + e_i \tag{4}$$

- Sometimes have used reciprocal or square root, but not so often.

Outline

The Quadratic Model: add $b_2 x_i^2$

- Replace the usual

$$b_0 + b_1 x_i \tag{5}$$

- With this

$$y = b_0 + b_1 x_i + b_2 x_i^2 \tag{6}$$

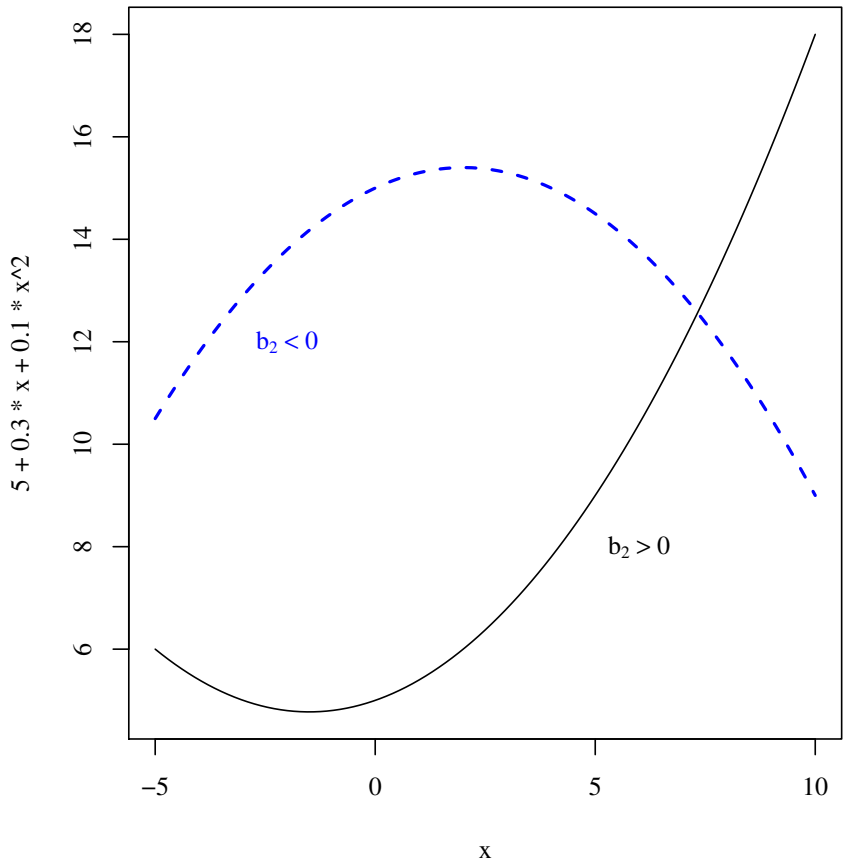
Graph the Quadratic

- Remember high school math. If
- $b_2 < 0$, then this is a "hill" shaped function
- $b_2 > 0$, then this is a "U" shaped function
- The "peak" or "bottom" occurs where

$$x_i = \frac{-b_1}{2b_2} \tag{7}$$

- From elementary calculus, recall the overall slope of y_i as a function of x_i is

$$\frac{dy}{dx_i} = b_1 + 2b_2 x_i \tag{8}$$



Reasons to believe that $y_i = b_0 + b_1 x_i + b_2 x_i^2$ is a Useful Model

- This is only one step more complicated than the linear model
- Including x^2 is like interacting x with itself

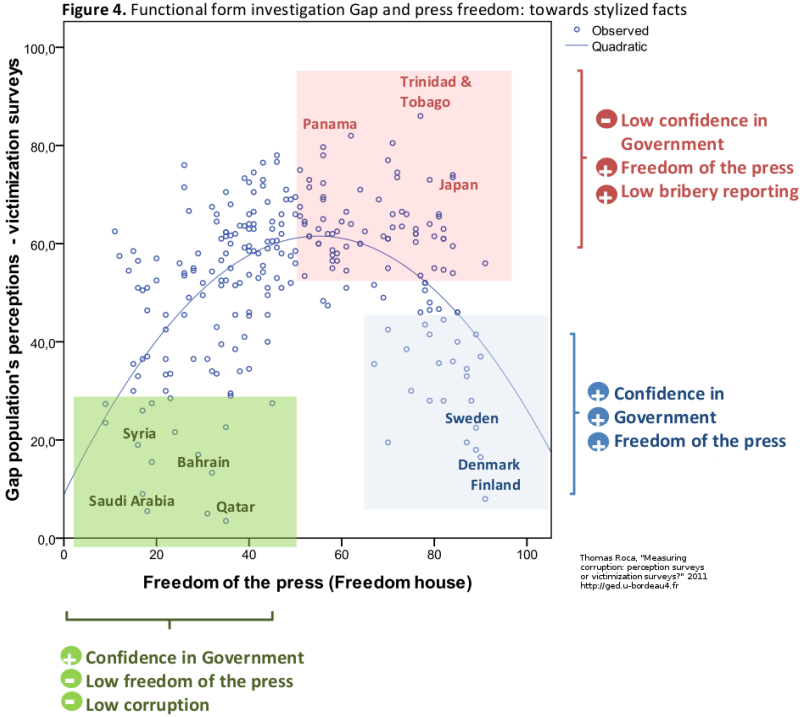
$$y_i = b_0 + (b_1 + b_2 x_i) x_i + e_i \tag{9}$$

  - The effect of x_i depends on where x_i "is": $b_1 + b_2 x_i$.
- If $b_2 > 0$, then the marginal effect of x_i is always getting bigger as x_i gets bigger

Quadratic Example #1

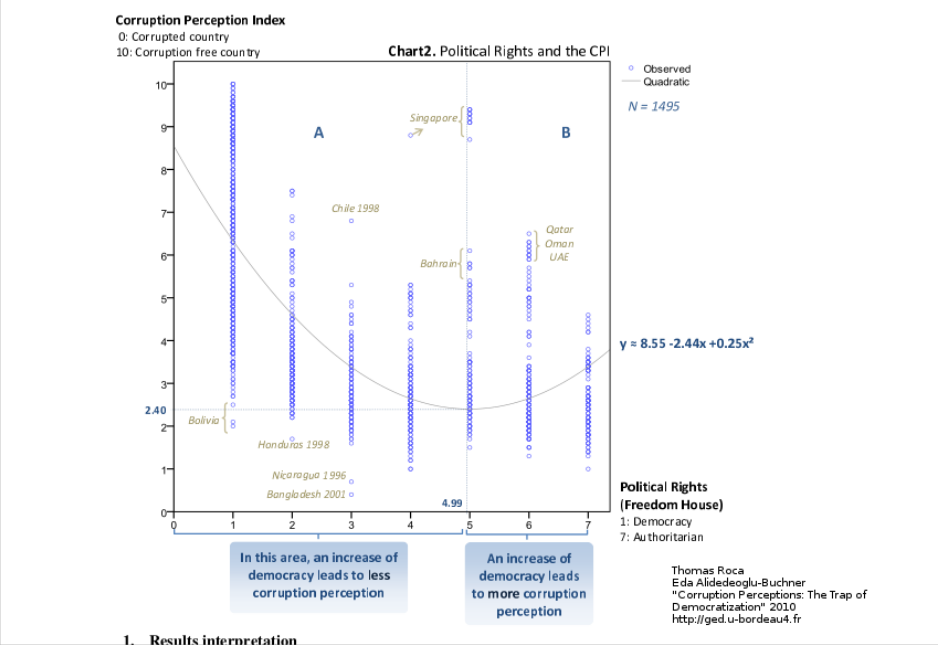
From these stylized facts, it is possible to construct the following pattern:

Figure 4. Functional form investigation Gap and press freedom: towards stylized facts



- From Thomas Roca. 2011. "Measuring Corruption: Perception Surveys or Victimization Surveys"
- Quatratic fits OK, but I wish I had some function with a sharper peak.

Quadratic In Use #2



- From Thomas Roca. 2010. "Corruption Perceptions: The Trap of Democratization"
- Quatratic fits OK, but I wish I had some function with a sharper peak.

Outline

Log, Square Root, Reciprocal

- Log on the right

$$y_i = b_0 + b_1 log(x_i) + e_i \tag{10}$$

- Square root on the right

$$y_i = b_0 + b_1 \sqrt{x_i} + e_i \tag{11}$$

- Reciprocal

$$y_i = b_0 + b_1 \frac{1}{x_i} + e_i \tag{12}$$

- In either case, we just calculate a new variable,

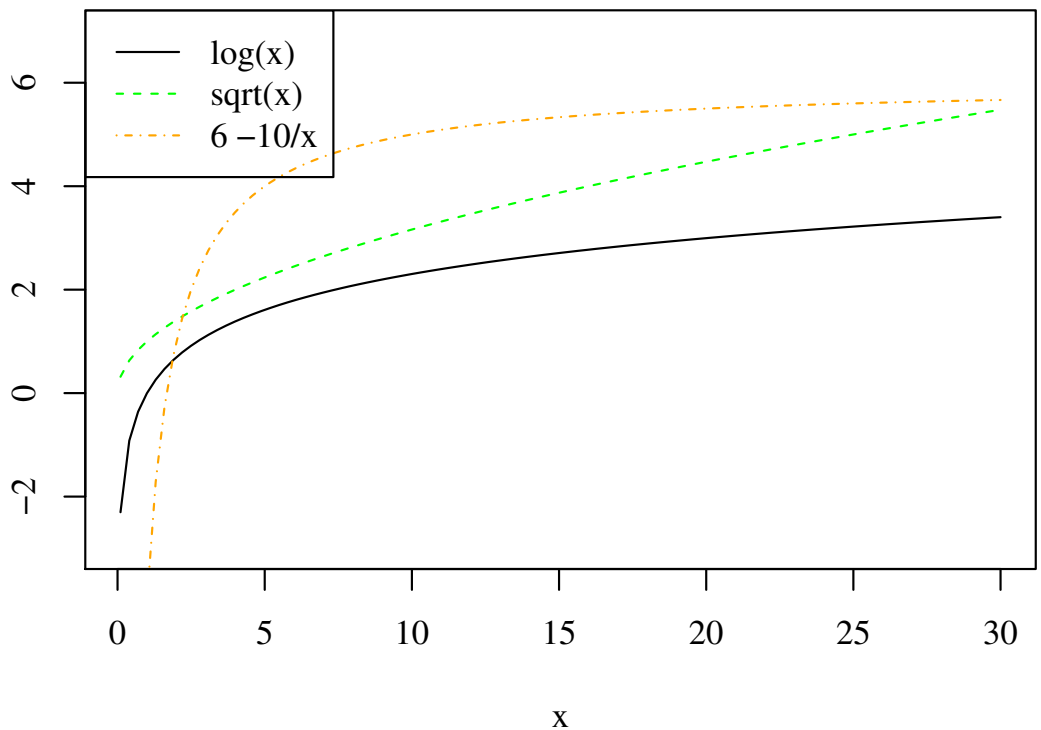
xlog = log(x)	xsqrt = sqrt(x)	xrec = 1/x

- And fit a regression:

lm(y ~ xlog)	lm(y ~ xsqrt)	lm(y ~ xrec)

Why These, Not Quadratic?

- The Quadratic goes up and down
- These are *monotonic* transformations
- log and sqrt are unbounded, reciprocal approaches a limit

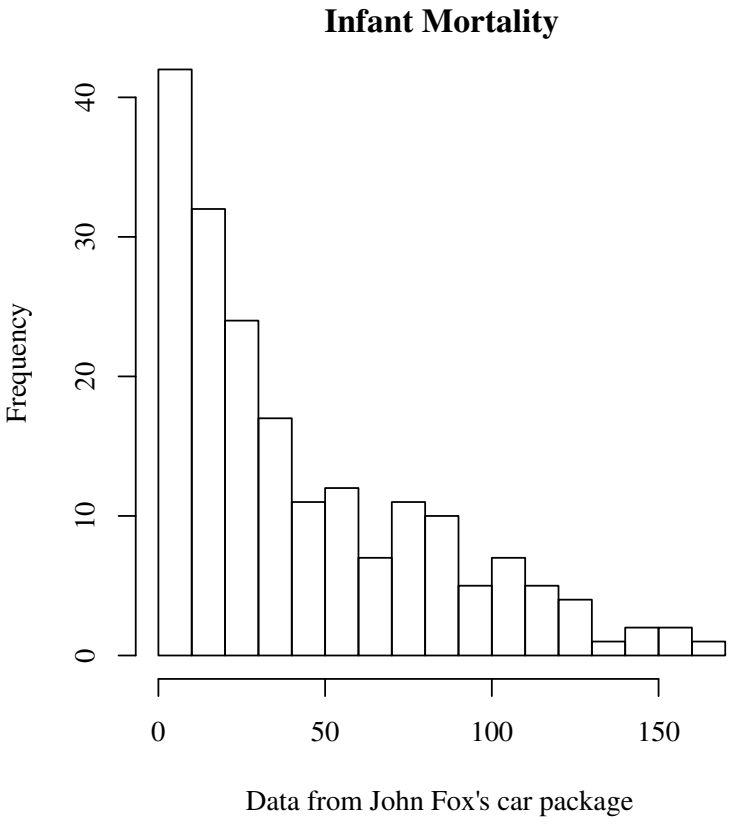


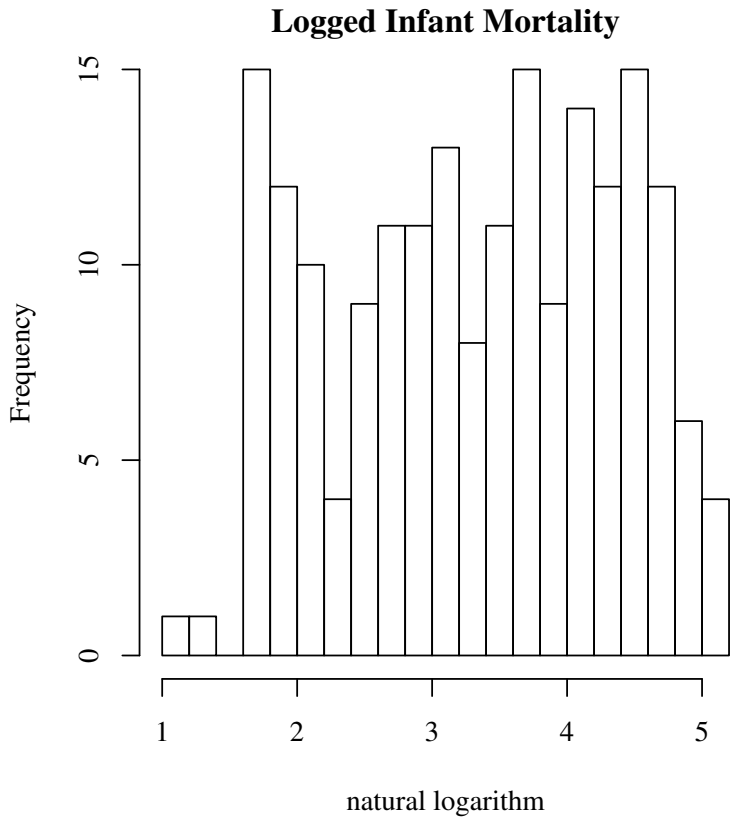
Outline

Suppose Distribution of y Variable is "clumpy", "skewed"

- When data appears to "clump" in the low ranges, replacing y by *logy* results in a regression that more closely fits the assumptions.
- log often applied to income or health data.
- Variations:
  - Use $log(\alpha + x)$ (Can assume α or estimate using tools in the R MASS package.)
  - The Box-Cox transformation has log as a special case

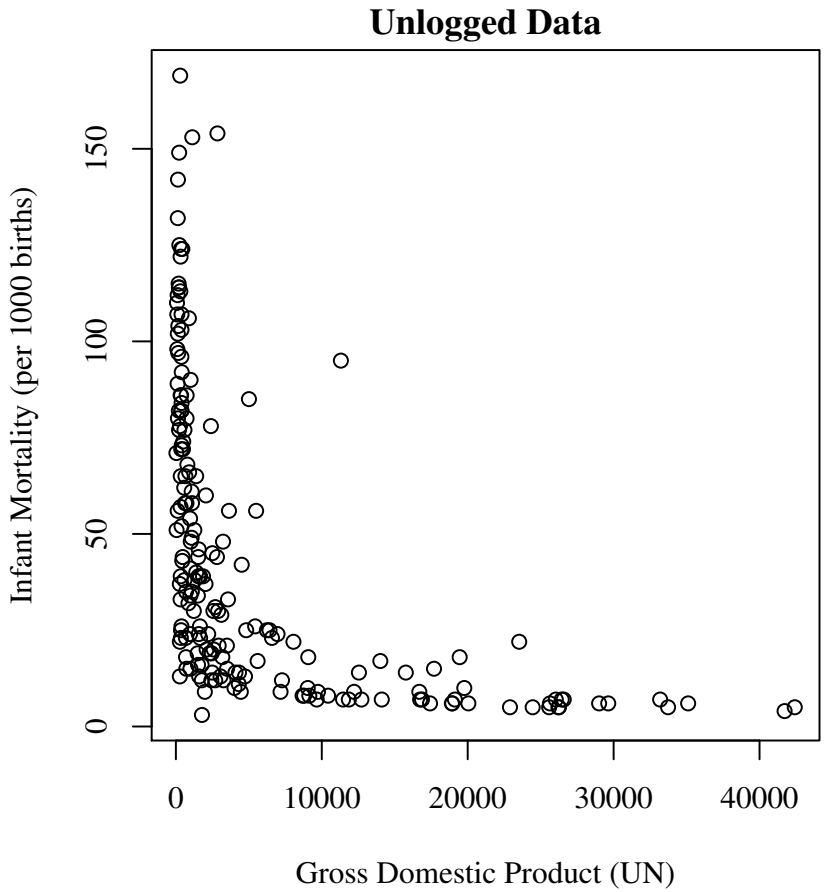
Examples: Infant Mortality

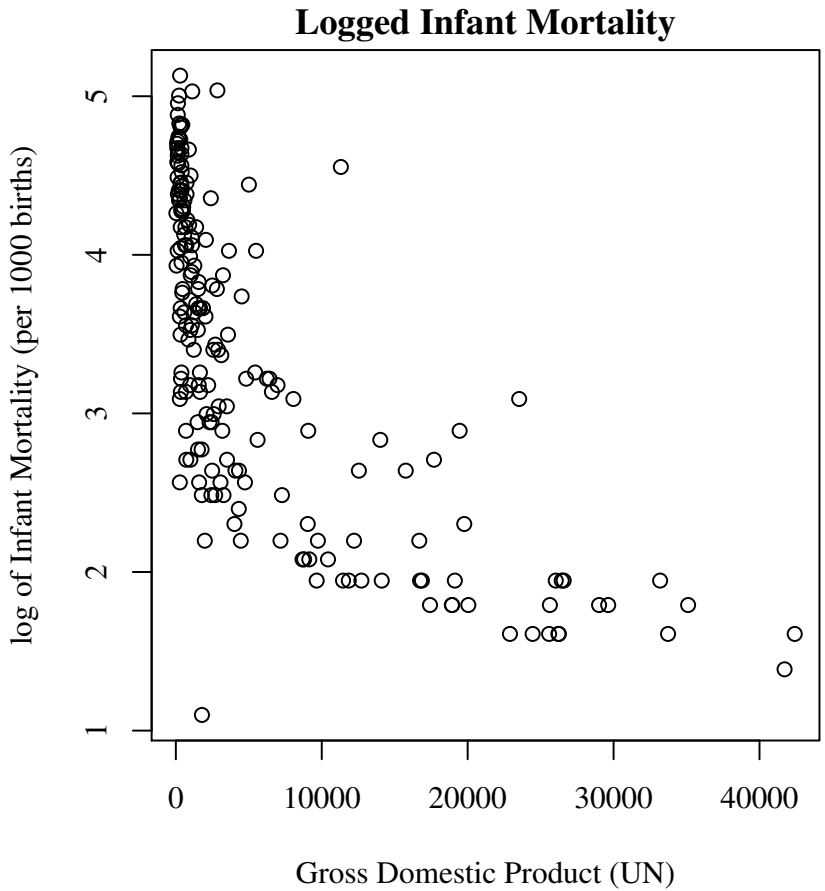




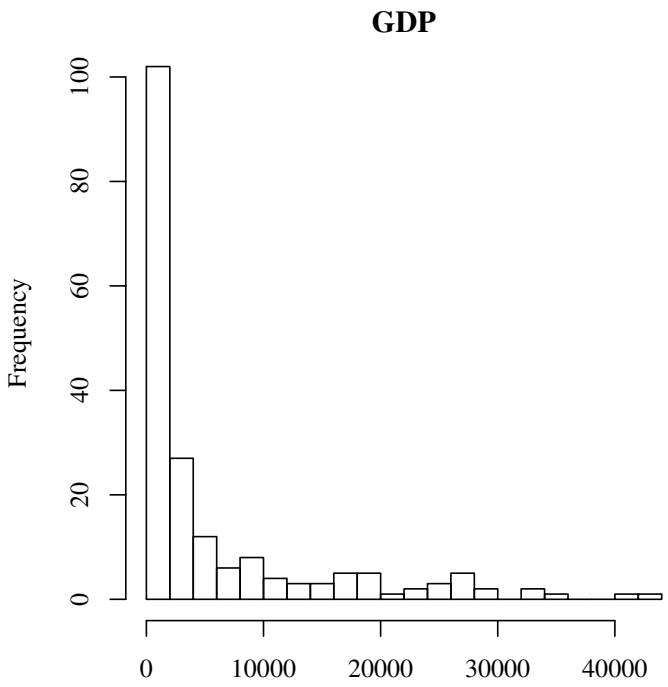
IMHO: not normal, certainly more symmetric

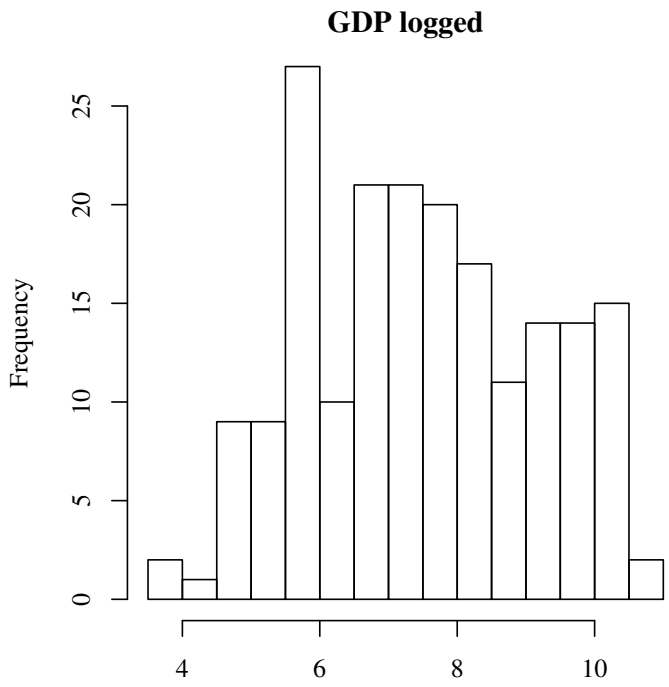
Scatter Still Bad, however





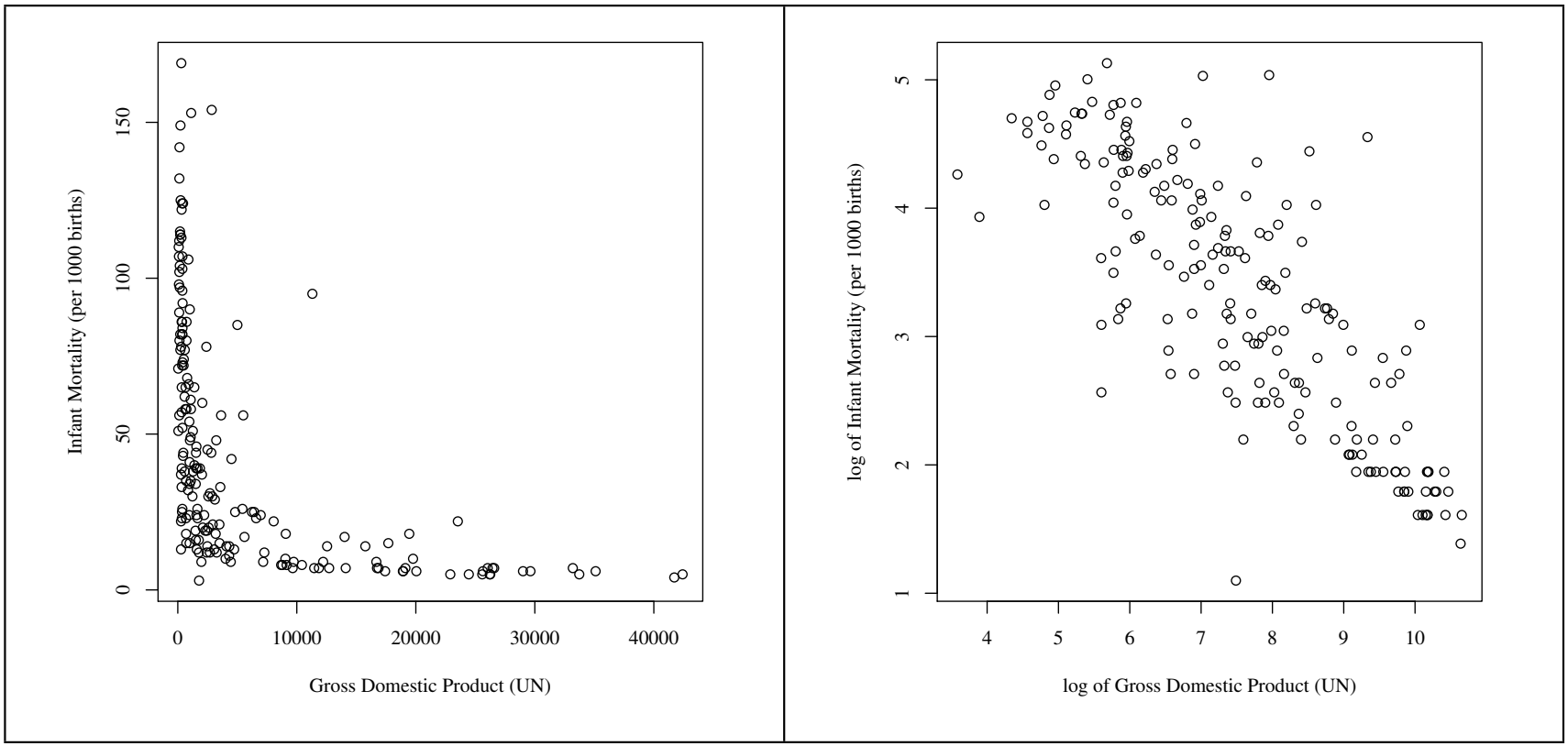
Look at the GDP variable





Log both DV and IV, some Inexplicable Magic Happens

United Nations infant mortality data from United Nations (Package "car" by John Fox)



Recall laws of logarithms

- $log(x^k) = k\, log(x)$
- $log(x * z) = log(x) + log(z)$
- $log(x/z) = log(x) - log(z)$

- The log and exp are inverse functions: each "reverses" the effect of the other.
- $log(exp(x)) = x$
- $exp(log(x)) = x$

Can Fit with OLS

- Step 1: We need to Log y (Recall the distribution is not compatible with regression unless we do)
- Step 2: In the Scatter, it appears as though x has a nonlinear effect

$$ylog_i = b_0 + b_1 xlog + \varepsilon \tag{13}$$

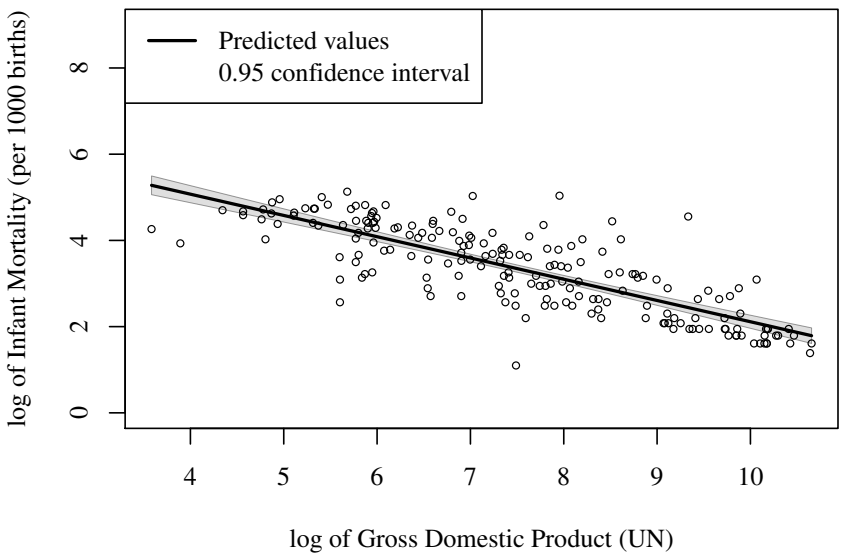
- It appears that the pleasant scatter of *ylog* on *xlog* is just a happy coincidence of all of this.

Can Fit with OLS

- I suggest

```
dat$ylog <- log(dat$y)
m1 <- lm(ylog ~ log(x), data = dat)
```

Application



	M1	
	Estimate	(S.E.)
(Intercept)	7.045***	(0.199)
gdplog	-0.493***	(0.026)
N	193	
RMSE	0.594	
R^2	0.656	

$*p \leq 0.05$** $p \leq 0.01$***$p \leq 0.001$

There's one Unsolved Problem (Embarrassing)

- The predicted values are on the scale of "log y".
- People often want to "re-transform" them to the y scale.
- Until 2014, I was teaching people "Its OK to anti-log $\widehat{ylog}$". However...
- In 2014, I learned that's wrong,
- 'Back-transforming' onto the y scale is conceptually difficult, goes beyond scope of this class.

Hence, think of ylog as your actual variable

- Call the DV ylog and don't try to talk about scores on the y scale at all
- If you insist on back-transforming, go read this:
  Naihua Duan. (1983). "Smearing Estimate: A Nonparametric Retransformation Method," *Journal of the American Statistical Association*, 78 (3838): 605-610.
- Related commentary on many websites:
  `http://healthcare-economist.com/2010/11/16/duans-smearing-estimator`

Outline

Why so much stress about squares?

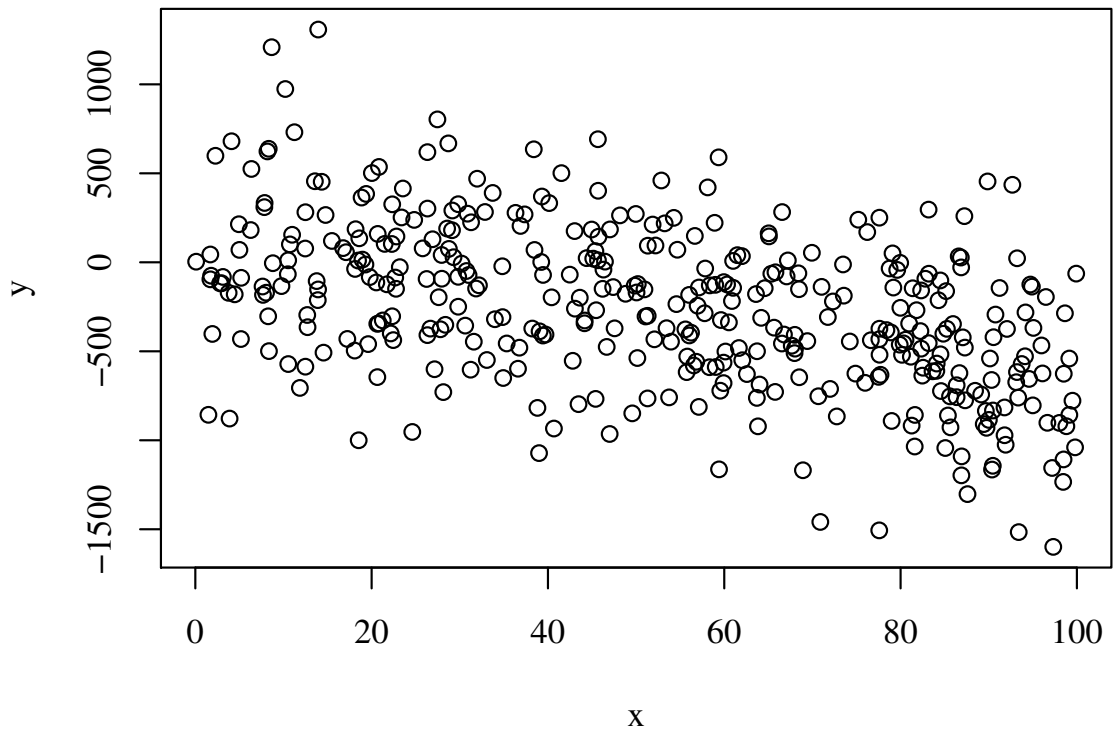
- When we use $\log(x)$, or $sqrt(x)$, or $1/x$ as a predictor, we just transform and go.
- If we want to add x^2 as well as x, then we have 2 "collinear columns" and there are some special concerns
  - and various suggestions (good and bad)

Ways to fit quadratic Regressions

Work through my website:
http://pj.freefaculty.org/R/WorkingExamples/regression-quadratic-1.R.
Here's some fake data:

```
STDE = 400; b0 = 3; b1 = .2; b2 = -0.075
x <- runif(400, 0, 100)
x <- x[order(x)]
dat <- data.frame(x)
dat$xsq <- x*x
dat$y <- b0 + b1 * dat$x + b2 * dat$xsq + rnorm(400, m=0, s = STDE)
rm(x)
xcorr <- cor(dat$x, dat$xsq)
```

Ways to fit quadratic Regressions ...



In the olden days of SAS, we'd manually create a new variable "xsquare".

```
dat$xsq <- dat$x^2
m1 <- lm(y ~ x + xsq, data = dat)
```

Ways to fit quadratic Regressions ...

- Disadvantage. Estimation routine does not know that x and xsq are "linked together", some follow-up calculations won't come out correct (consider the drop1() function in R, for example).

1. We fix that by asking R's formula-handler to create x-squared for us (R will remember the 2 variables are linked).

```
m2a <- lm(y ~ x + I(x*x), data = dat)
```

I() "protects" the contents, makes sure they are treated as a mathematical expression.

Danger, Will Robinson

The Major Concerns

1. Collinearity: x-squared is correlated with x. (Pearson R often .8 or higher). That causes unstable parameter estimates, not much "power" to detect effects.
   The Pearson R between *x* and *xsq* is 0.971
2. if x is a big number, say 1000, then squaring it will give 1,000,000. Computer "rounding" error kicks in when columns are on different scales. With modern matrix algebra for linear models, this is not such a bad problem as it used to be. But if we were doing Maximum Likelihood or Nonlinear Fitting, we would worry.

Fancier ways to create 2nd column.

1. Fancy 1: Orthogonal polynomial. Generate 2 new columns for us

```
m1 <- lm(y ~ poly(x, 2), data = dat)
```

poly(x,2) creates 2 columns, but they are not correlated with each other. To see that, run

```
xpoly <- poly(dat$x, 2)
head(xpoly)
```

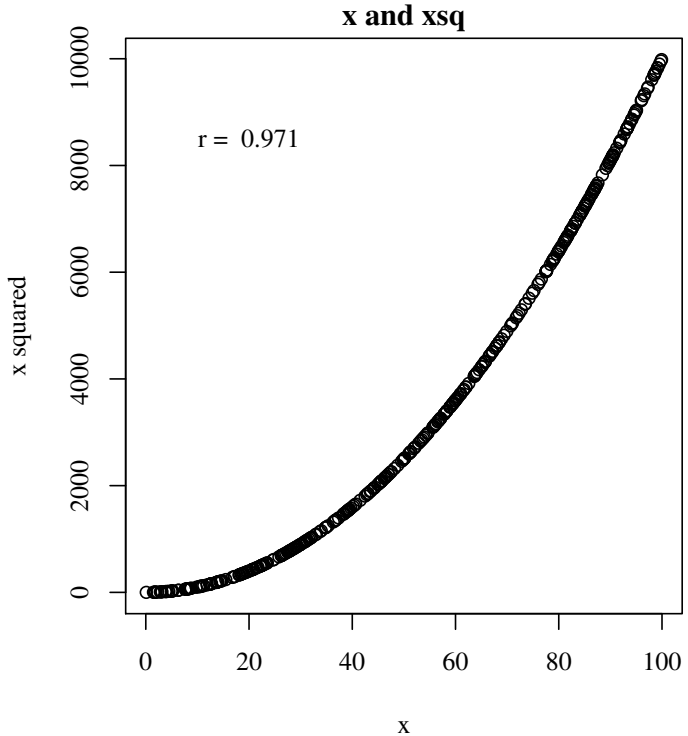
```
                1          2
[1,] -0.09043700 0.1234184
[2,] -0.08802246 0.1136961
[3,] -0.08764475 0.1121995
[4,] -0.08763258 0.1121514
[5,] -0.08749189 0.1115957
[6,] -0.08724543 0.1106244
```

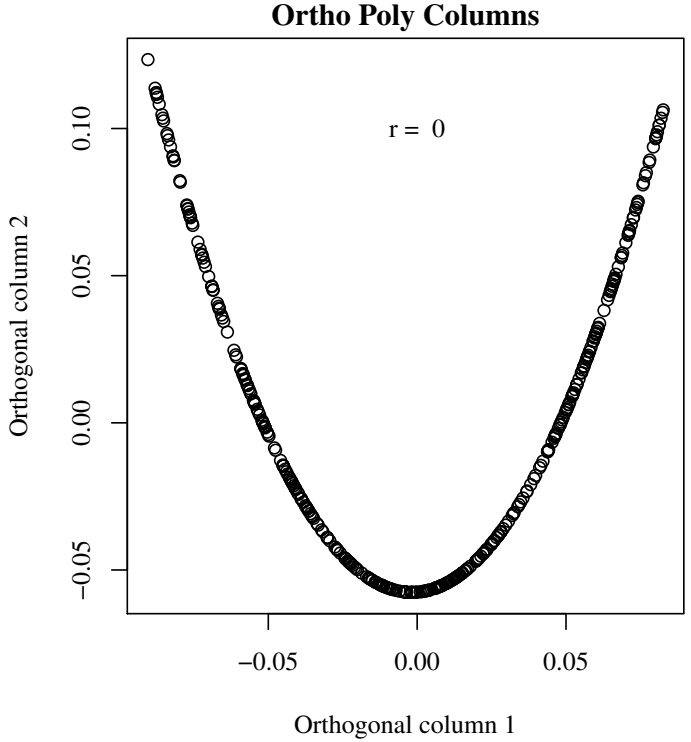
```
(pcorr <- cor(xpoly[, 1], xpoly[ ,2]))
```

```
[1] -1.599071e-16
```

Fancier ways to create 2nd column. ...

The poly() function essentially "extracts" the linear part–x–from the term representing x-squared. The two columns of xploy are orthogonal!
Compare the plots





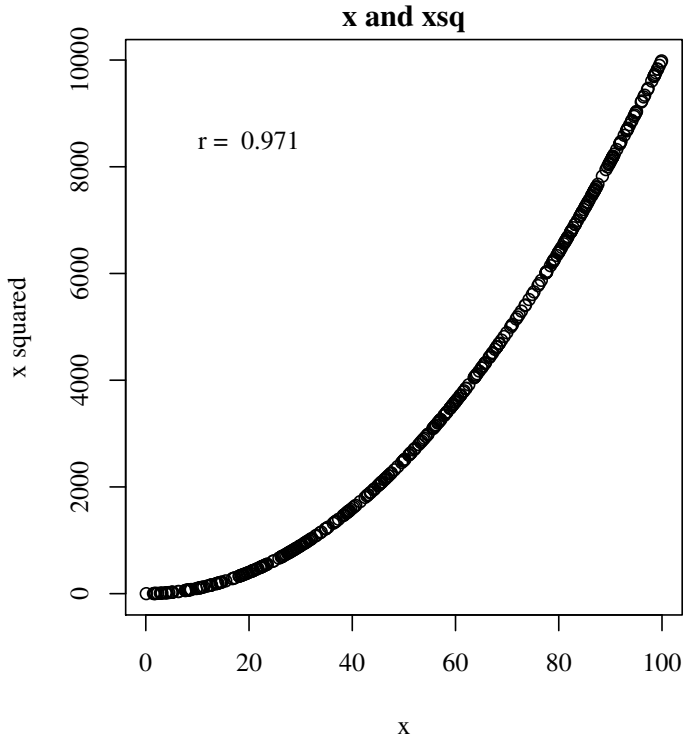
Fancier ways to create 2nd column. ...

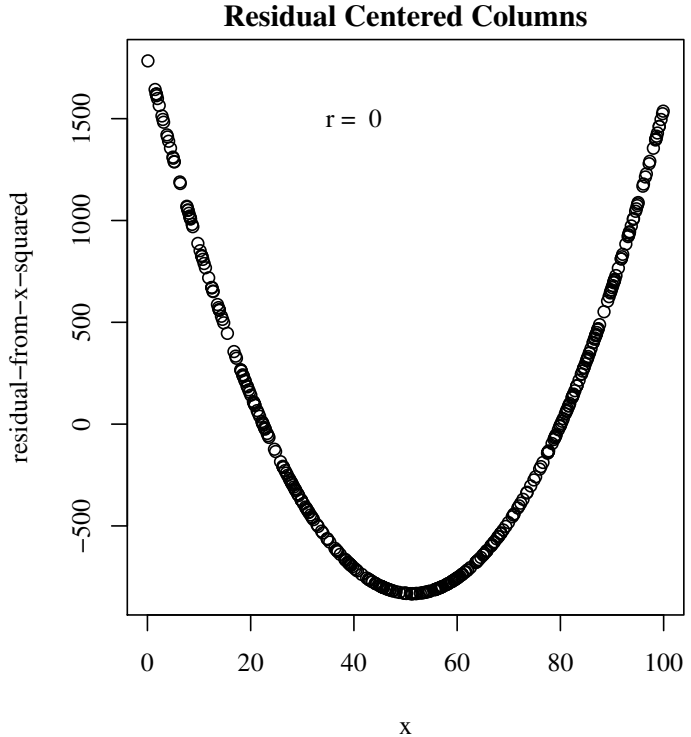
When we fit regressions with this variable, the estimates for the linear term (column 1) is stable. It is unaffected by the introduction of column 2. (Because it is ORTHOGONAL, silly)

2. Fancy 2: Residual-centering of x (rockchalk::residualCenter()). This creates "orthogonal" variables (like orthog poly), but they are not scaled similarly
   1. Fit $x^2 = b_0 + b_1 x_i + \varepsilon_i$
   2. Get the residuals, they are the part of x_i^2 that is separate from x_i
   3. Run a regression
      $y = b_0 + b_1 x_i + b_2 residual.from.previous + e_i$

Fancier ways to create 2nd column. ...

Compare the plots





Note difference on numerical scale, could cause numerical rounding error, but residual centering reduces much of the imbalance

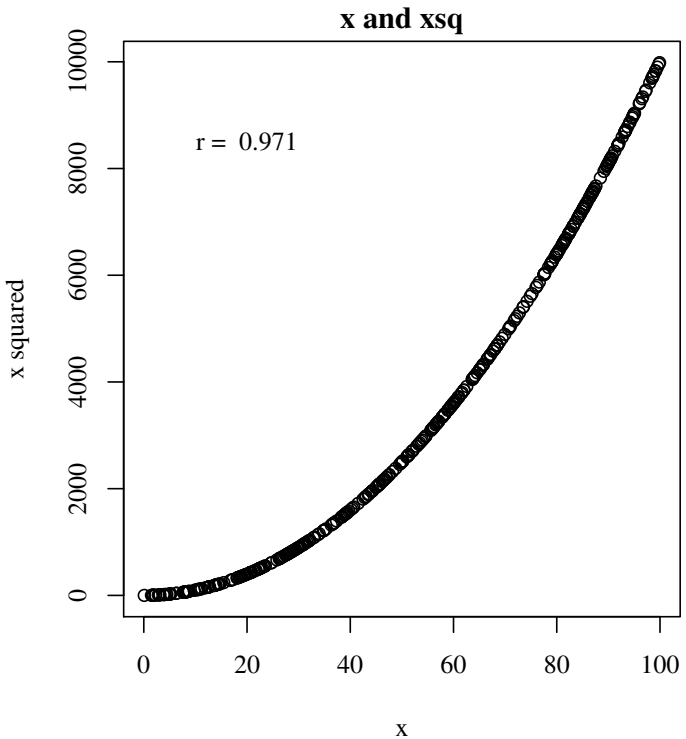
Fancier ways to create 2nd column. ...

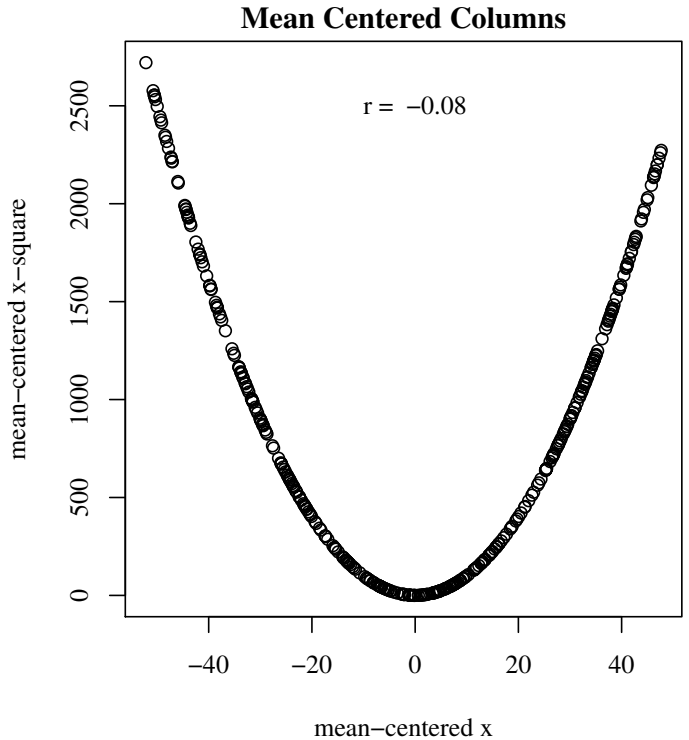
3. Fancy 3: Mean-center x (rockchalk::meanCenter())
   Create a centered variable, $xc = x_i - \bar{x}$, then fit
   $y = b_0 + b_1 xc + b_2 xc^2$

```
dat$xctr <- scale(dat$x, scale = FALSE)
m1 <- lm(y ~ xctr + I(xctr^2), data = dat)
```

Mean centering alters coefficients, but does not give predictor columns that are (strictly speaking) orthogonal.
Compare the plots

Fancier ways to create 2nd column. ...





What are you *Suppposed* to do?

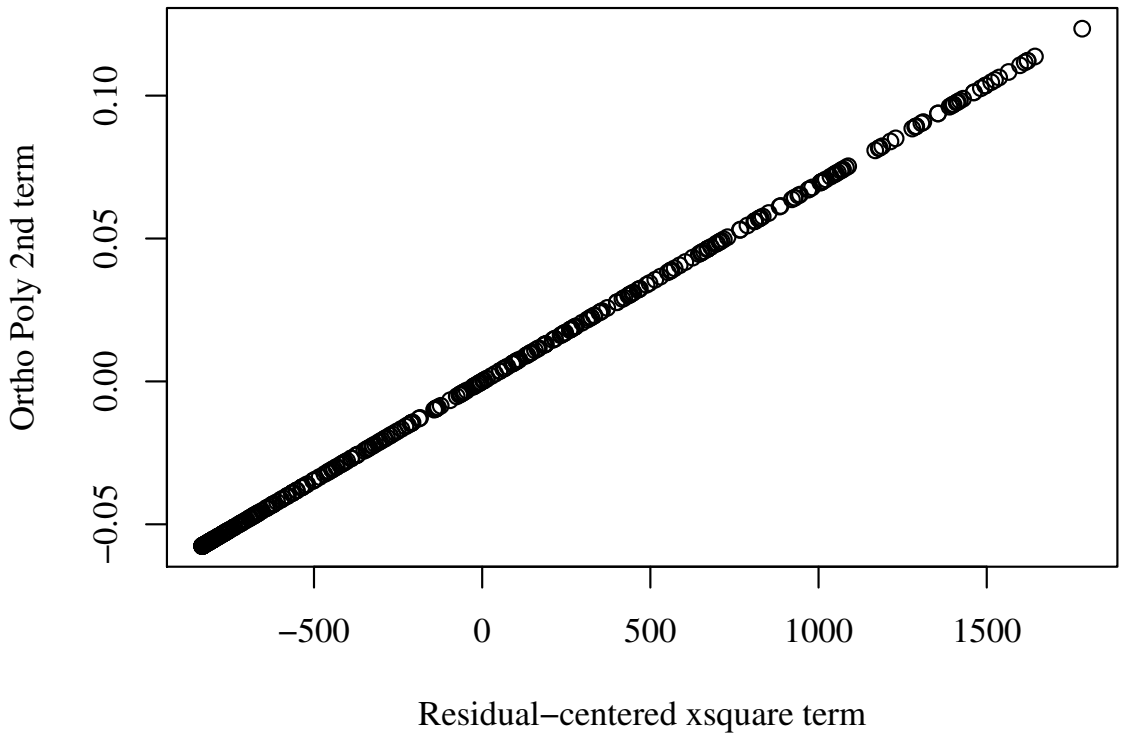
1. The Best thing, orthogonal polynomial, is also most difficult to understand (math!)
2. The "easiest" approach is mean-centering. However, it is useless, deceptive
3. Residual centering is easier to understand, creates orthogonal columns, and so is mostly as good as orthogonal polynomial. It is not *as good* because it does not give back variables on the same scale.

However
None of these change the predictive relationship we are estimating, so, actually, they are all the same (as we will see)

Where is the difference among these?

- Mean centering alters both terms. It Uses
  - centered X and
  - (centered-X)-squared.
- Residual Centering uses X (untransformed) and an orthogonal transformation representing the squared-effect.
- Orthogonal Polynomial columns are proportional to the Residual Centered columns, but they are scaled onto identical ranges.

Ortho Poly Equals Residual Centering (mostly)



CCWA recommend Centering Data

- Cohen, et al 2002 p. 204 "Centering also eliminates the extreme multicollinearity associated with using powers of predictors in a single equation. We therefore strongly recommend the use and reporting of centered polynomial equations."
- This claim is technically false.
  Echambadi, R., & Hess, J. D. (2007). Mean-Centering Does Not Alleviate Collinearity Problems in Moderated Multiple Regression Models. *Marketing Science*, 26(3), 438-445.
- Actually, as we shall see, none of these models "solve" multicollinearity in any real sense. They just alter our perception of it.

Your Intuition Should Have Told You

- Why? It is a simple re-scaling of a predictor!
  - $xc_i = x_i - \bar{x}$, the mean of xc_i is 0. How could this

$$y_i = b_0 + b_1 xc_i + b_2 xc_i^2 + \varepsilon_i \tag{14}$$

be meaningfully different from this:

$$y_i = b_0 + b_1 x_i + b_2 x_i^2 \tag{15}$$

They aren't different, the algebra is worked out in the *rockchalk* vignette.

- But it creates the deception of help! That re-positions the y axis to $\bar{x}$.
- There is a deeper lesson about multicollinearity in all of this.

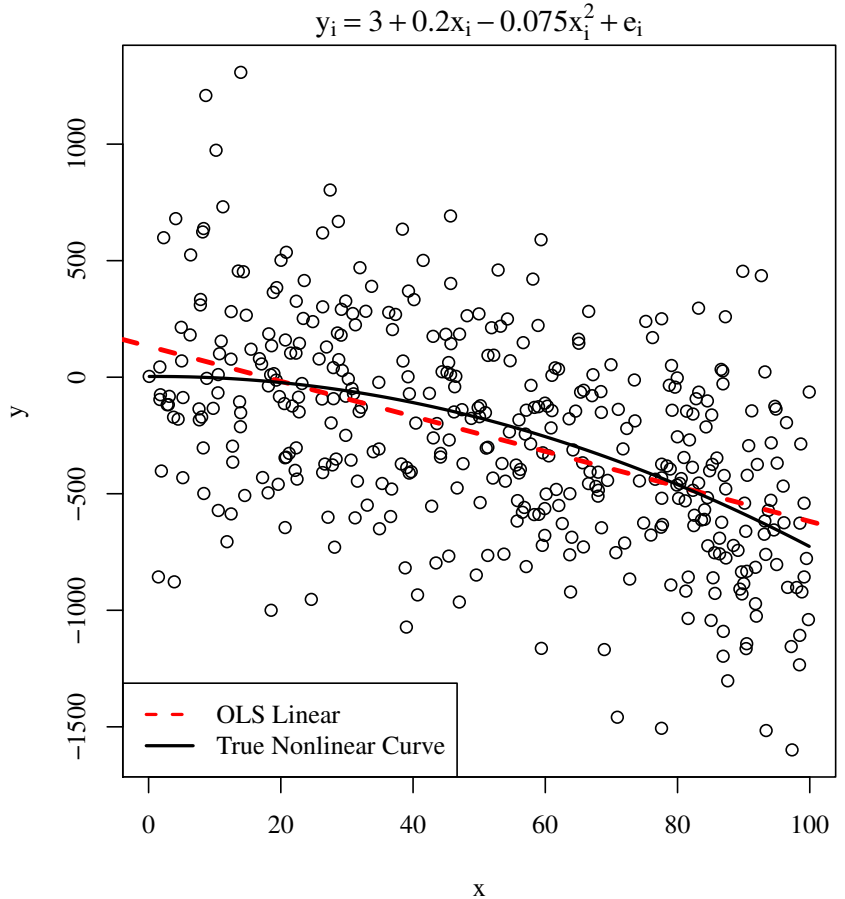
Estimation: without x-squared

Fit a "mis-specified" linear equation (ignoring the squared component).

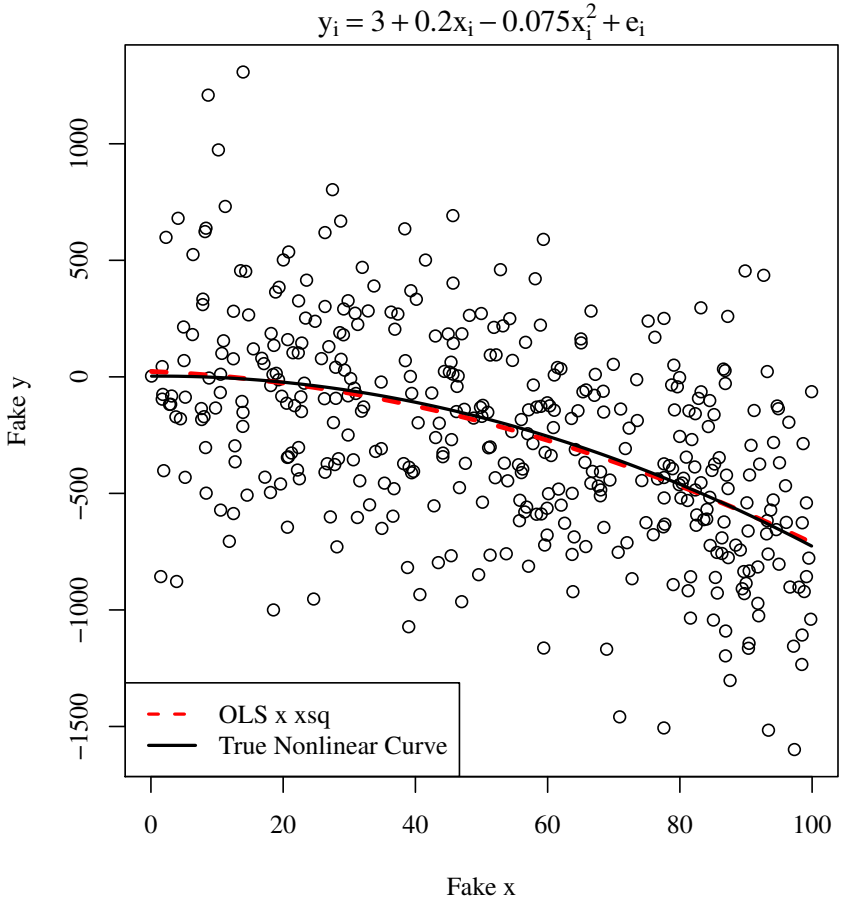
	M1	
	Estimate	(S.E.)
(Intercept)	132.450**	(41.097)
x	-7.510***	(0.689)
N	400	
RMSE	397.147	
R^2	0.230	

$*p \leq 0.05$** $p \leq 0.01$***$p \leq 0.001$

Hooray! Effect of x is statistically significant(ly different from 0)



Estimation: with x-squared



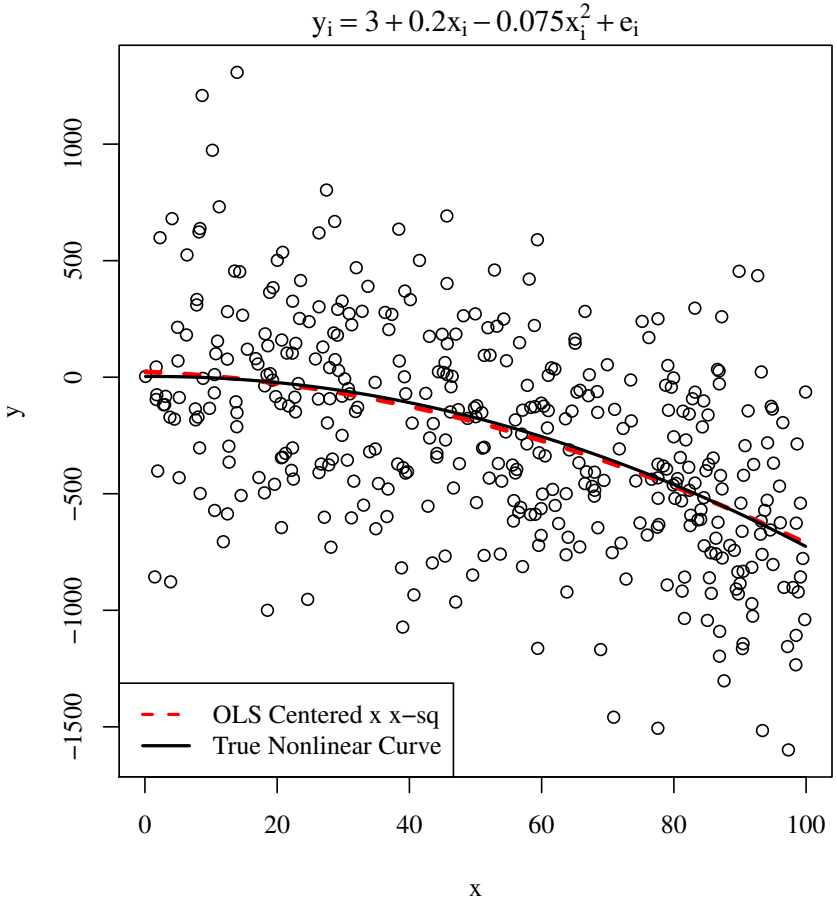
	M1	
	Estimate	(S.E.)
(Intercept)	23.142	(63.849)
x	-1.263	(2.885)
xsq	-0.061*	(0.027)
N	400	
RMSE	395.181	
R^2	0.240	
adj R^2	0.236	

$*p \leq 0.05$** $p \leq 0.01$***$p \leq 0.001$

Bone Crushing Defeat. x is "not significant" anymore.

Cohen et al claim this results from 'nonessential multicollinearity' between x and x^2.

Illustration with Mean Centered x



	M1	
	Estimate	(S.E.)
(Intercept)	-209.286***	(30.122)
xc	-7.633***	(0.687)
xcsquare	-0.061*	(0.027)
N	400	
RMSE	395.181	
R^2	0.240	
adj R^2	0.236	

$*p \leq 0.05$** $p \leq 0.01$***$p \leq 0.001$

Oh, Yeah! Effect of x is "significant" again

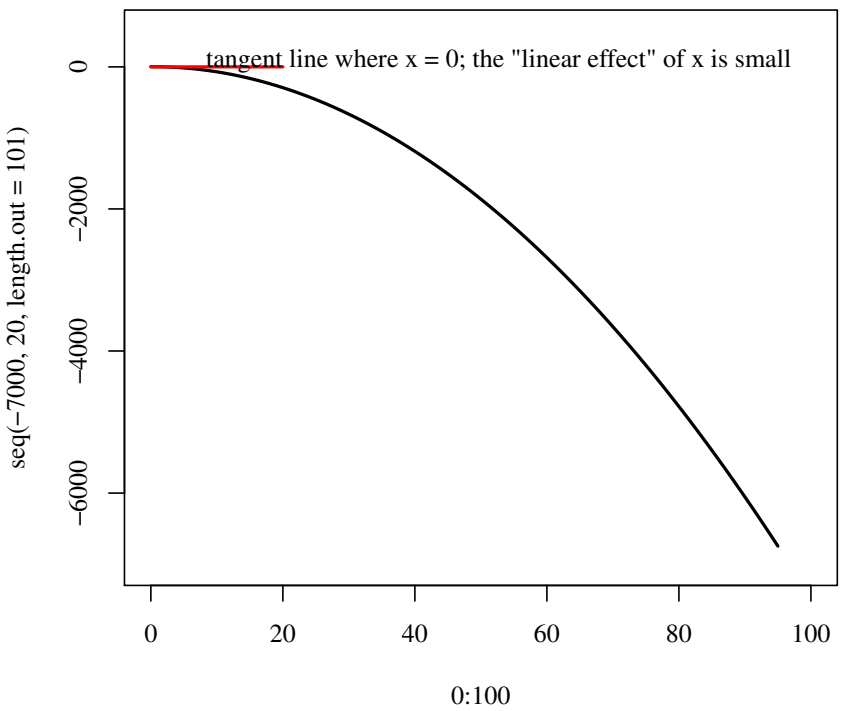
Visualize Mean-Centering

The regression reports on b_0 and b_1 are "snapshots", estimates based on one particular point on the curve.

- Consider the "true" relationship:

$$3 + 0.20x_i - 0.075x_i^2 \qquad (16)$$

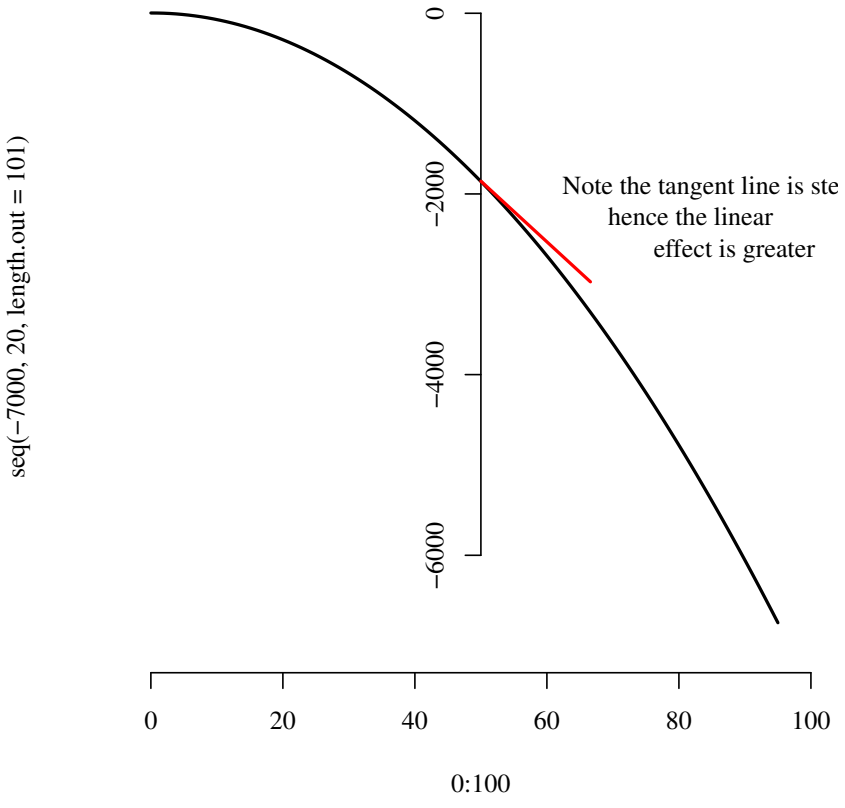
- Put the $y - axis$ where x $= 0$
- Linear effect is small there, right?



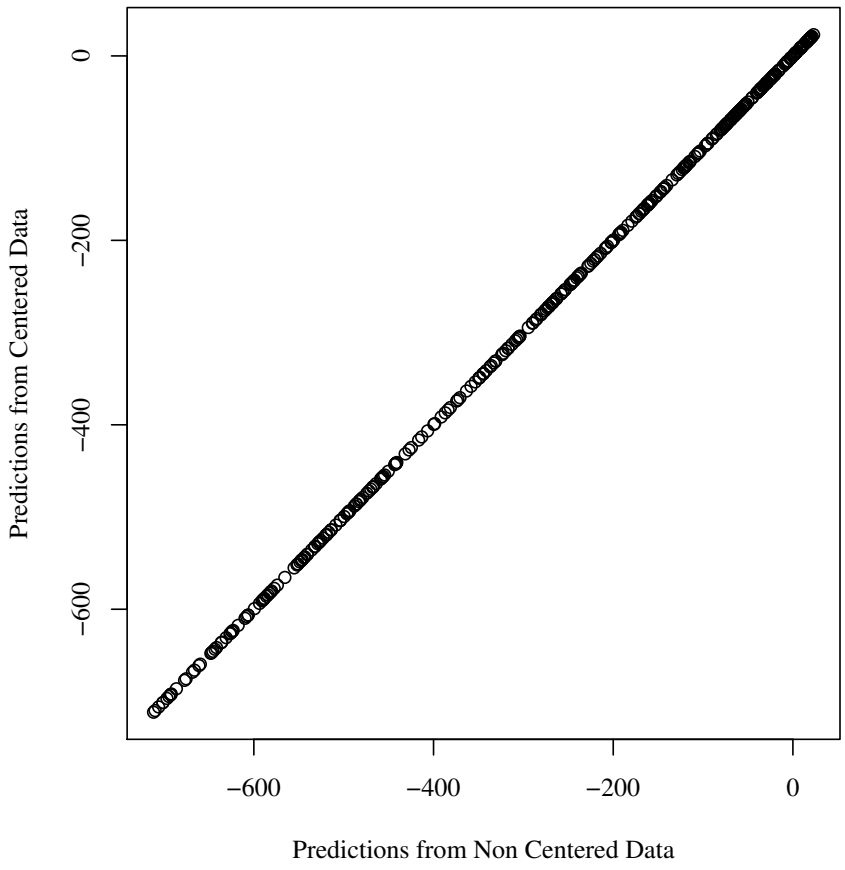
- The linear effect is the tangent line's slope, $b_1 + 2b_2x_i$. Plug in $x_i = 0$,we see the slope is b_1, a very small number.

Visualize Mean-Centering

- Put the $y - axis$ where x = 50
- Note the tangent line's slope is "very steep" there. That will lead to a large estimate of b_1.
- If 50 happens to be the observed mean, then mean-centering amounts to moving the y axis to that position.



The Predicted values of the non-centered and centered models are identical

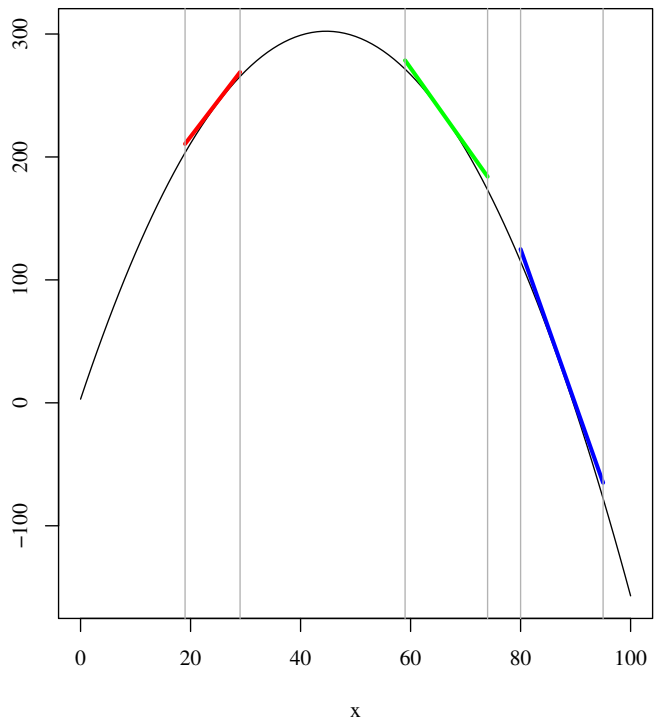


You Can't Get More Identical Than That!

- The rockchalk package was developed mainly because our class was in a constant stew of confusion about mean centering (which some faculty recommended), residual-centering (which others recommended), and no-centering (which I recommended).
- There are worked examples included with the package (look in the install folder) and there is a long discussion of it in the vignette.

Tempted to Mislead the Reader By Shifting the Y axis?

Remember this picture?



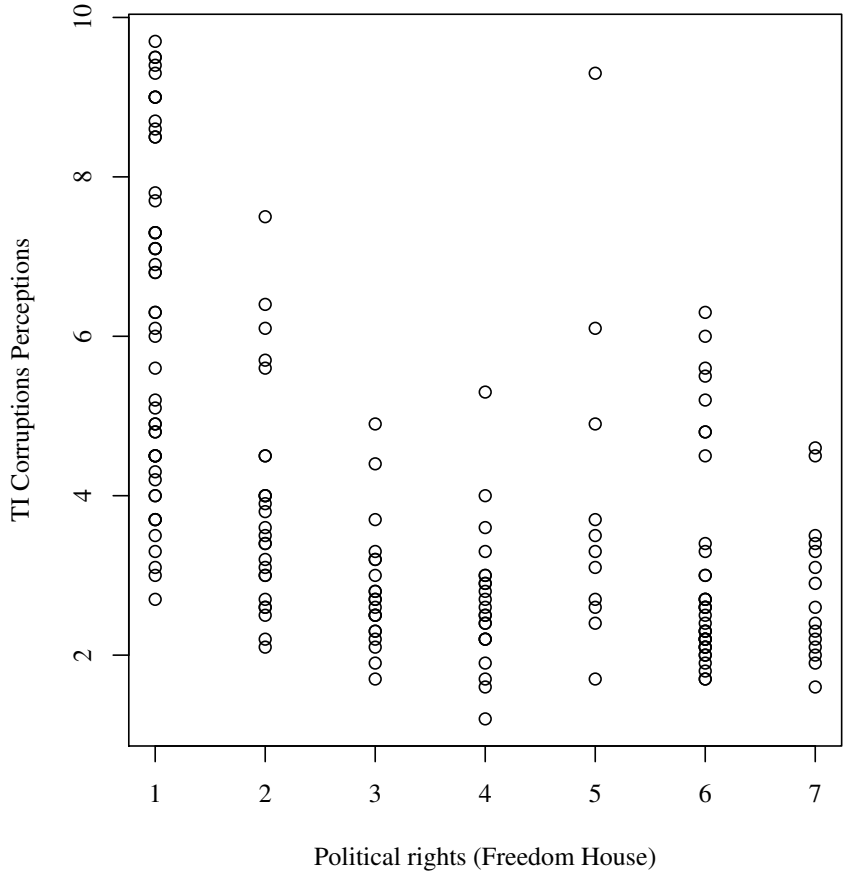
- By replacing x with $x -$ "*any constant you name*", we are implicitly re-positioning the $y -$ *axis* to select one of these colored lines!
- Maybe $x - mean(x)$ repositions us so that the regression summary seems "better" (more stars!)
- But it is really the exact same regression.

Conclusion: Centering Not Helpful, Not Harmful

- Although centering variables is eagerly recommended by many books, it is not actually having a substantial effect on the quality of the fitted model. The Predicted values are the same.
- Note that $\hat{b}_2$, the R^2 and the $RMSE$ are unchanged by centering.
- In the olden days, when the matrix $(X^T X)$ was inverted to calculate $\hat{b}$, mean centering may have helped with the problem of floating point approximation on digital computers. Today, regression estimates are calculated with orthogonal matrix decomposition methods (either QR or SVD), and thus we don't notice much benefit by mean-centering. (Other fitting methods may not use those more sophisticated algorithms, thus we might need to be more cautious).

The Corruption Example

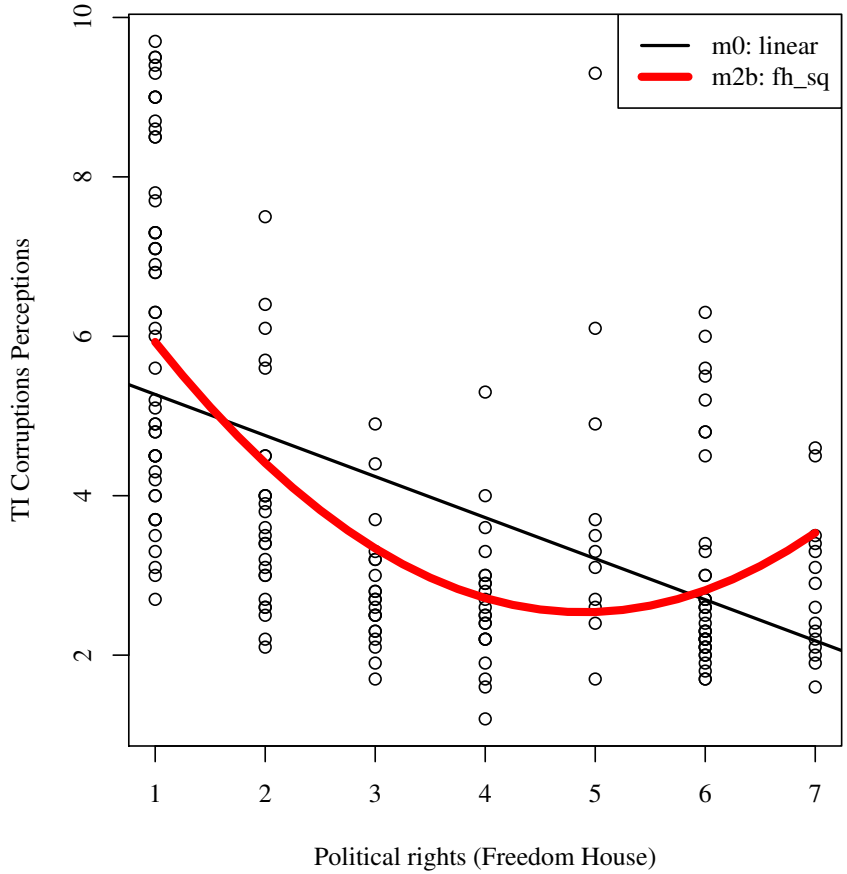
- Recall Roca's plot & regression
- "Quality of Government" data set
  Teorell, Jan, Nicholas Charron, Marcus Samanni, Sören Holmberg & Bo Rothstein. 2011. The Quality of Government Dataset, version 6Apr11. University of Gothenburg: The Quality of Government Institute, `http://www.qog.pol.gu.se`
- Scatter TI Corruption Perception Index against the Freedom House political rights scale



The Corruption Example

- The linear and quadratic fits

```
m0 <- lm(ti_cpi ~ fh_pr, data=dat)
m2b <- lm(ti_cpi ~ poly(fh_pr, 2, raw
    = TRUE), data=dat)
newdf <- data.frame(fh_pr = plotSeq(
    dat$fh_pr, length.out=25))
m2bpred <- predict(m2b, newdata =
    newdf)
```



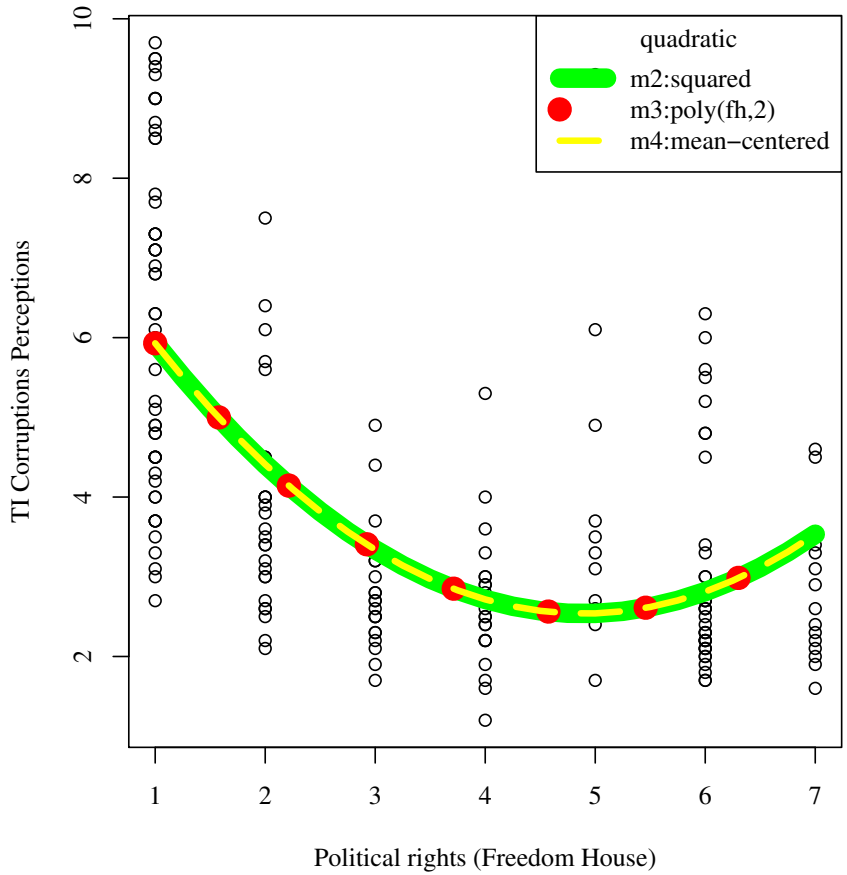
Equivalent Predictions From all Fits To The Corruption Example

- Insert mean-centered columns in the newdf object

```
dat$fh_pr_mc <- drop(scale(dat$fh_pr, scale = FALSE)
    )
newdf$fh_pr_mc <- plotSeq(dat$fh_pr_mc, 25)
```

```
m2 <- lm(ti_cpi ~ fh_pr + I(fh_pr^2), data=dat)
m2pred <- predict(m2, newdata = newdf)
m3 <- lm(ti_cpi ~ poly(fh_pr, 2), data=dat)
m3pred <- predict(m3, newdata=newdf)
```

```
m4 <- lm(ti_cpi ~ fh_pr_mc + I(fh_pr_mc^2), data=dat
    )
m4pred <- predict(m4, newdata=newdf)
```



Outline

Theory has to guide you on choosing a formula.

- The data will seldom give you a good reason to pick one model over another.
- MUST NOT compare R^2 across models that are fitted to different transformed values of y
- If the left hand side of the expression is the same, then *Adjusted* R^2can serve as a guide for picking the best fitting model.
- In more advanced models, information theoretic measures like AIC and BIC might be used.

Interpretation of results is the most important part.

- Consider the effort you make to interpret an OLS model. "Each unit increase in x causes a $\hat{b_1}$ increase in the expected value of y.
- You need to make a similar effort to interpret a nonlinear model, remembering that each one has unique mathematical properties.
- Usually these are things to look for:
  1. Can you understand the slope of the line representing the expected value?
  2. Does the function have a maximum value that is substantively important?
  3. Are there any "special" values of the parameters that you need to watch out for and give special interpretation.

Outline

B-C: Log transform on steriods

- The transformation $log(y)$ might make a variable that is more symmetric
- But is it as close to symmetry as possible?

Conditional Functional Relationship

Box & Cox proposed a flexible transformation: generates a "family" of possible models.

- The transformation depends on a parameter λ, which can take on values in a continuum.

$$y^{(\lambda)} = \begin{cases} \frac{y^{\lambda}-1}{\lambda} & \text{if } \lambda \neq 0 \\ log_e(y) & \text{if } \lambda = 0 \end{cases}$$

- Estimate λ iteratively. Guess $\hat{\lambda}$, transform, fit. Improve $\hat{\lambda}$, re-fit the model. Continue...

Check particular values of λ

λ	transformation
0	$ln(y)$
1	$y-1$
$\frac{1}{2}$	$2(\sqrt{y}-1)$

I have a separate handout on these models called BoxCoxRegression.

Outline

Take a Simple Example with logs

One version of the double-log model goes like this.

- theory: $y_i = x1_i^{b_1} exp(b_0 \varepsilon_i)$
- log both sides produces a model that OLS can estimate: $ylog_i = b_0 + b_1 x1log_i + \varepsilon_i$

Suppose "Nature" uses a complicated formula.

- You hypothesize some wild theoretical formula as your data generating process

$$y_i = 14 \cdot x1_i \cdot e^{5.0+3.7 \times log(x2_i)+\varepsilon_i} = 14x1_i exp(5.0 + 3.7log(x2_i) + \varepsilon_i)$$

$E(\varepsilon_i) = 0$, and e represents "Euler's constant", $e^x = exp(x)$

- That's too specific. Leave some parameters to estimate:

$$y_i = b_0 \cdot x1_i \cdot e^{b_1+b_2 \times log(x2_i)+\varepsilon_i}$$

- Look for an estimator, either "nonlinear least squares" or "maximum likelihood".
- Usually, you can't prove the estimators are unbiased!
- They may be
  - consistent
  - efficient
  - asymptotically normal

Most Understandable alternative: Nonlinear least squares

- Nonlinear least squares (NLS) . Basically, you write down a formula, assume the error is additive, and go:

$$y_i = f(X_i, b) + e_i \tag{17}$$

- The estimator "guesses" parameters $\hat{b}$ and from there is calculates predictions

$$\hat{y}_i = f(X_i, \hat{b}) \tag{18}$$

- Fit by making the sum of squared errors the smallest:

$$SS(\hat{b}) = \sum_{i=1}^{N} [y_i - f(X_i, \hat{b})]^2 \tag{19}$$

Use nls from R base on the UN problem

- Here's my theory

$$inf.mortality_i = b_0 + b_1 x_i^{b_2} + e_i \tag{20}$$

- Recall that x^{-b_2} equals $1/x^{b_2}$, so if $\hat{b}_2$ is negative, then this model is

$$\widehat{inf.mortality_i} = \hat{b}_0 + \hat{b}_1 \cdot \left(\frac{1}{x_i^{|\hat{b}2|}}\right)$$

- And that's really what I think. So I want a reciprocal model, but it is written down more generally.
- nls: adjust $\hat{b}_0$, $\hat{b}_1$, and $\hat{b}_2$ to make the squared prediction error as small as possible.
- Fitted model indicates that $\hat{b}_2 = -0.12$ (approximately -1/8) giving a very gradual curvature.

$$\widehat{inf.mortality_i} = -90.23 + 336.6 \cdot \left(\frac{1}{x_i^{\frac{1}{8}}}\right)$$

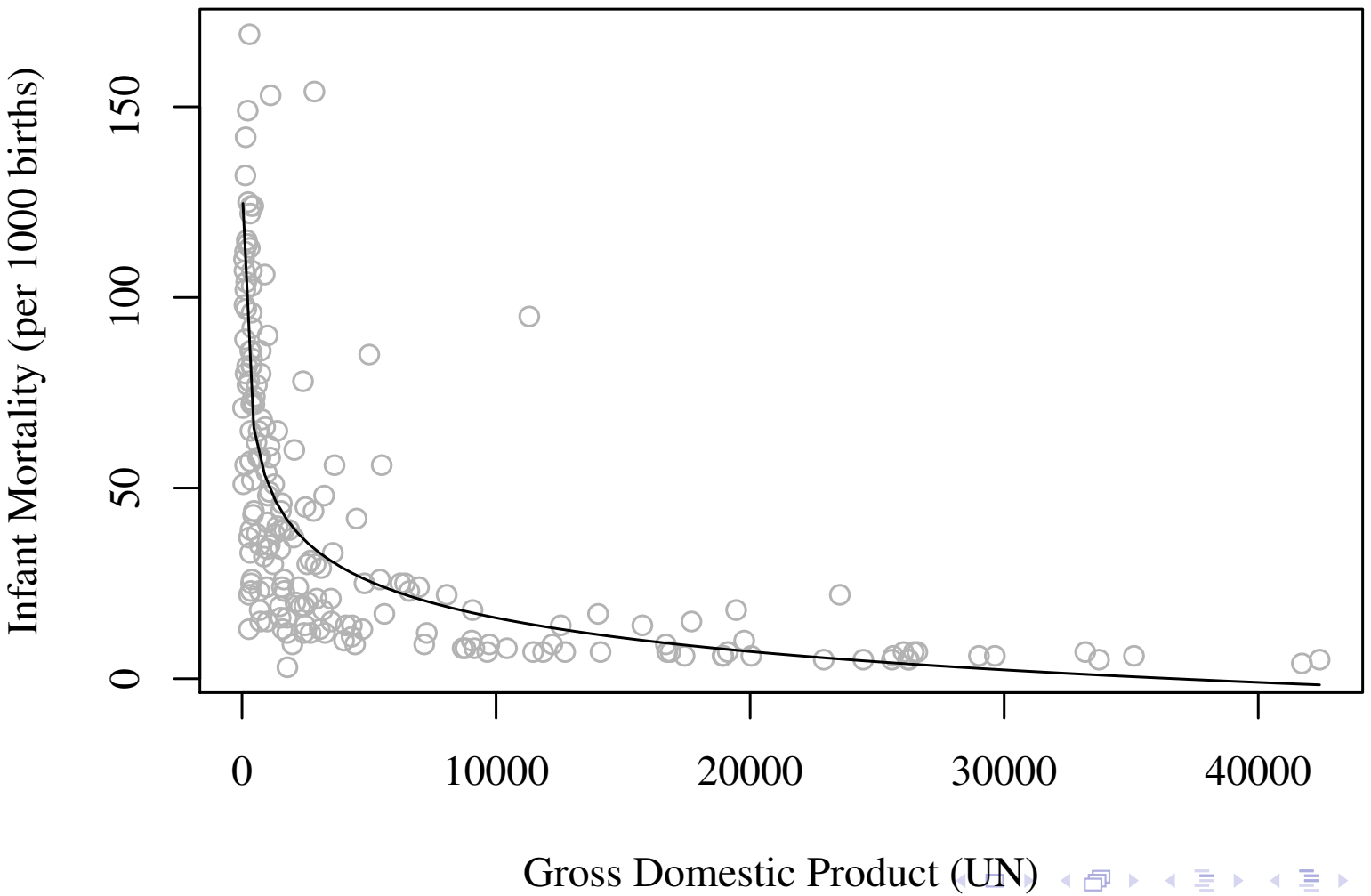
NLS Estimation Commands

```
nmod2 <- nls(infant.mortality ~ A + B*(gdp^C), data=UN, start=list(
    A=10,B=21,C=-1/10), control=nls.control(warnOnly=TRUE))
summary(nmod2)
```

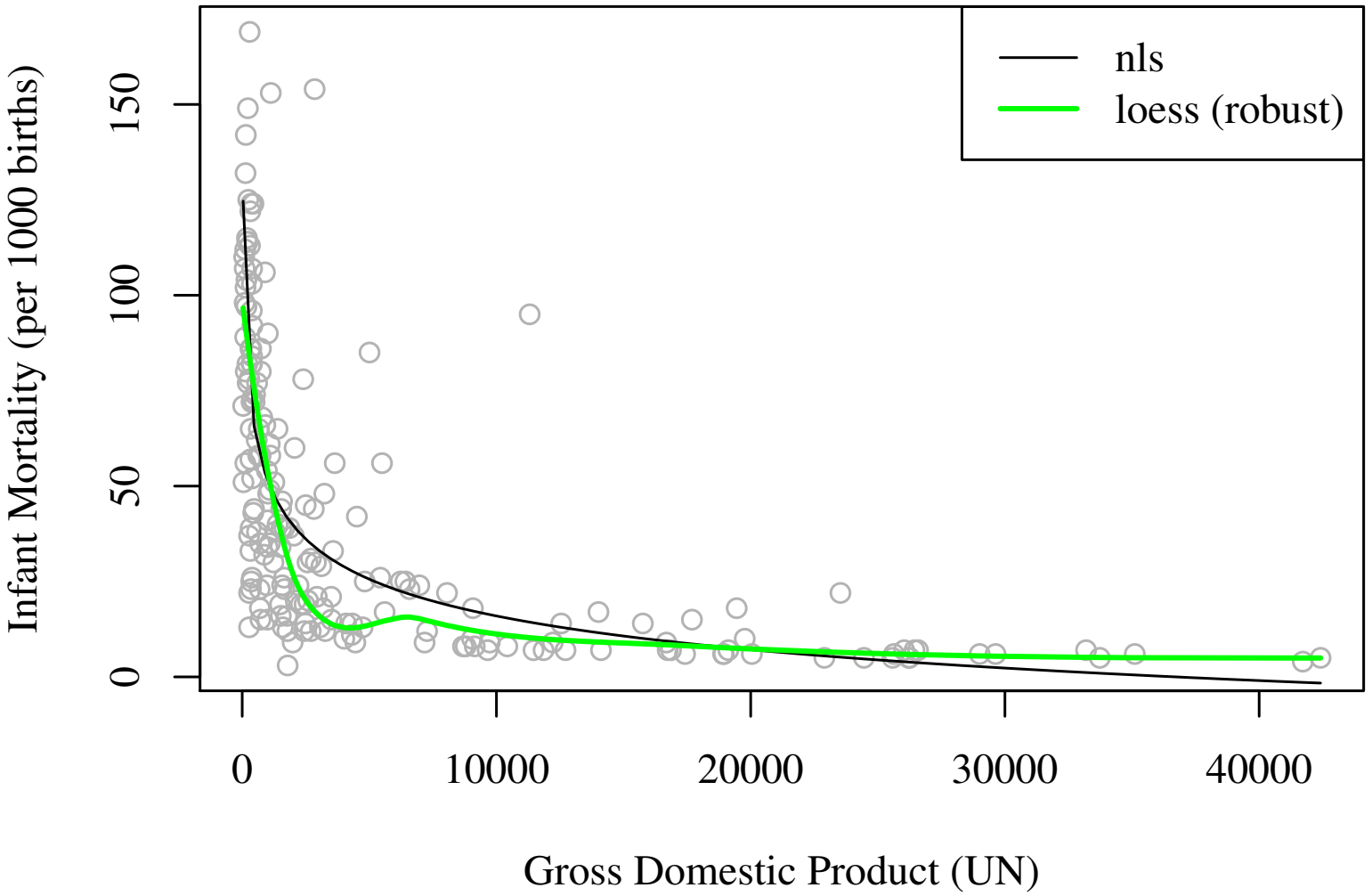
```
Formula: infant.mortality ~ A + B * (gdp^C)

Parameters:
   Estimate Std. Error t value Pr(>|t|)
A -90.23234   84.12664  -1.073    0.285
B 336.62030   31.21030  10.786   <2e-16 ***
C  -0.12525    0.07836  -1.598    0.112
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 26.94 on 190 degrees of freedom

Number of iterations to convergence: 9
Achieved convergence tolerance: 5.329e-06
```

Plot the Predicted Values from NLS

```
newdf$gdplog <- log(newdf$gdp)
```



Throw a loess on top



Outline

Problems

Regression modelers need to accumulate a 'toolbox' of functions that can be used to transform data. There are basics like logarithms, square roots, powers, and so forth. To force yourself to become familiar, consider these questions.

1. I'm looking at a scatterplot with some data that looks curved. I'm unsure if the formula I use should be $y_i = b_0 + b_1 log(x_i) + e_i$ or $y_i = b_0 + b_1(1/x_i) + e_i$. Sketch the predictive lines that those formulae imply and advise me which I should use. Remember that b_1 might be positive OR negative.
2. We have some substantive reasons to fit models that are quadratic: $y_i = b_0 + b_1 x_i + b_2 x_i^2 + e_i$. I need to know if you can articulate some reasons.
    1. That looks like you just threw an x_i^2 on the end for not good reason. Develop a substantive "story" (theory?) that might justify a formula like that. To make this plausible, you usually need to name x_i, so it represents something like "hours of job training" or "megabytes of hard disk storage" or whatnot.

Problems ...

2. When we develop a story to justify the addition of a squared term, we usually suppose that the coefficient b_2 is either positive or negative. Can you explain the significance of that sign?
3. The Cohen textbook mentions that the top of the peak (if $b_2 < 0$) or the bottom of the valley (if $b_2 > 0$) occurs where $x_i = \frac{-b_1}{2b_2}$. Sometimes that information can be very useful in evaluating fitted models. Choose values for the b's and check to see if the value of x_i that corresponds to your maximum (or minimum) is substantively important.

3. Generate some data that more-or-less fits a linear model, fit a regression (the "right regression"). Then create new variables, the log(y) and log(x). Estimate regression models in which you replace the correct variables with the logged versions. Fit one with log(y), one with log(x), one with both. Do the parameter estimates and model fit statistics give you any good information? Here's how I created my example data. You can fiddle parameters b0, b1, stde to suit your taste.

Problems ...

```
b0 <- 5
b1 <- -0.2
x <- rnorm(500, m=50, s=10)
stde <- 9
err <- stde * rnorm(500,0,1)
y <- b0 + b1 * x + err
dat <- data.frame(x=x, y=y)
plot(x,y)
summary(lm(y~x, data=dat))
```

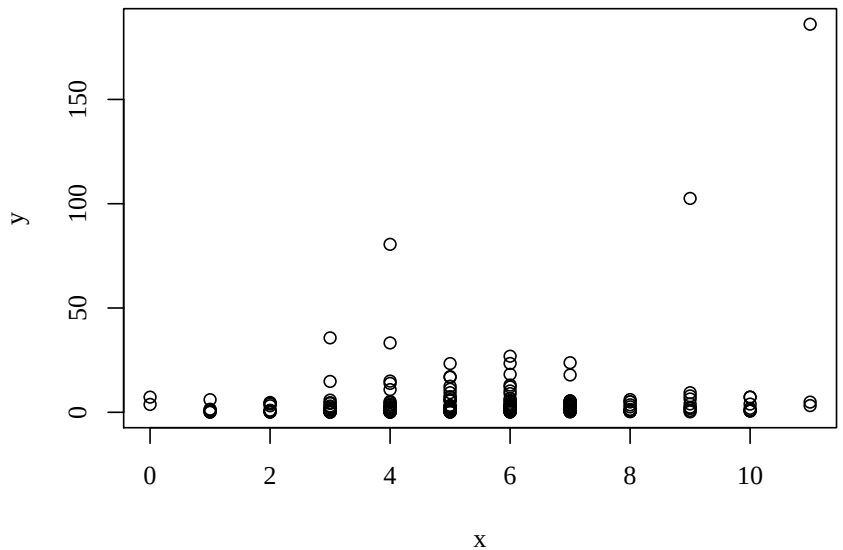
4. Let's make up some data. Then we pretend we don't remember how we created it, and we approach it in the usual way. Although this seems like a silly exercise, it is actually very instructive. Here's how I create my data:

Problems ...

```
b0 <- 0.005
b1 <- 0.1
x <- rpois(200, lambda=5)
stde <- 1.5
e1 <- stde*rnorm(200,m=0,sd=1)
y <- exp(b0 + b1*x + e1)
plot(x,y)
```

Here's what my plot of y on x looks like:

Problems ...



Wow. That is ugly.

Here are some things to try.

Problems ...

1. Fit a straight line OLS model predicting y from x. Estimate the regression, draw the line through the x-y scatter. Pretend you did not know what formula produced the data and you are forced to interpret the linear estimate as if you believed it were correct. As you go through the motions, think about this: What if your estimate of the slope is "statistically significant"? What would we usually say about it?

```
mod <- lm(y ~ x)
summary(mod)
plot(x,y,main="Linear Regression Predicting
     y from x")
abline(mod)
```

2. Create a set of the diagnostic regression plots for that data. Do the usual checks for linearity, homogeneity of variance, and outliers.

Problems ...

3. Throw in a squared x as a predictor. Carry out the regression analysis. Do your best to interpret the coefficients. Try to carry that out without looking at the scatterplot. Imagine what you would write about it, if you were forced to interpret this for your boss. Then look at the scatter and plot the predicted curve from your model.
How does this result shed light on the age old practice, which holds that if x^2 is statistically significant, then the true relationship is nonlinear and you have found it.

```
xsq <- x*x
mod <- lm(y ~ x + xsq)
summary(mod)
xsorted <- sort(unique(x))
predsq <- predict(mod, newdata=data.frame(x
    =xsorted, xsq=xsorted*xsorted))
plot(x,y,main="Scatter for y and x")
lines(xsorted, predsq)
```

Problems ...

4. I guess we better find out if "centering x" makes a difference in the quadratic equation you just fit. So subtract the mean of x from x, recalculate xsq, and fit the model. Is it easier to interpret?
5. You know how the data was created, so it will be easy for you to estimate the "right model's" coefficients with OLS, after you have properly transformed the variables. Do that. I'd like to ask you lots of questions about this, but I only have time for a few (sorry).
    1. First, how well does the transformed OLS model recover the parameters you set when the data was collected? Why not draw another random sample and see if the estimates stay in the same vicinity.
    2. Second, when you create the scatterplot of the properly transformed variables, and draw in the regression line, are there any obvious remaining signs of trouble?
    3. Third, the OLS model fitted to the transformed data can be used to make predictions on the scale of the original variables. How do those predictions differ from the linear and quadratic models discussed in the previous sections? Can you draw all of them on the same graph?

Problems ...

4. How's your R^2 on the correctly fitted "correct" OLS model? What would you say about it, especially in light of the fact that you know how the data is actually created?
   Here's some code I used to test this out

```
logy <- log(y)
mod3 <- lm(logy ~ x)
summary(mod3)
xsorted <- seq(range(x)[1], range(x)[2],
    length.out=20)
predsq <- predict(mod3, newdata=data.frame(
    x=xsorted))
par(mfrow=c(1,2))
plot(x, logy, main="Scatter for logy and x"
    )
lines(xsorted, predsq)
```

Problems ...

```
plot(x, y, main="y predicted from log(y)~x"
    )
lines(xsorted, exp(predsq))
par(mfrow=c(1,1))
```

5. I will keep thinking on this. I wish I had another small dataset like the United Nations example. That's a beautiful example of a horrible looking x-y plot that turns beautiful as logx-logy. If you have example data like that, why not pass it over?