Econometrics I - Non-Linear Relationships and Non-Linear Least Squares

Ryan T. Godwin

University of Manitoba

The models we've worked with so far have been linear in the parameters, and have been of the form:

$$\boldsymbol{y} = X\boldsymbol{\beta} + \epsilon$$

In this chapter we'll be interested in a more general model:

$$\boldsymbol{y} = f(\boldsymbol{\theta}; X) + \boldsymbol{\epsilon} \tag{1}$$

where $f()$ can be non-linear (the linear model is just a special case). Many relationships between variables are non-linear. Simple examples are diminishing marginal utility, or increasing returns to scale. If the data generating process specifies a non-linear relationship between the dependent and explanatory variables, LS may be biased and inconsistent.

Transforming a non-linear population model

In some situations where f in model 1 is non-linear, we may be able to transform the *variables* in order to linearize the relationship between y and X (for example taking logs of a Cobb-Douglas production function). Cobb-Douglas production function:

$$Y = AK^{\beta_2}L^{\beta_3}\varepsilon$$

By taking logs, the Cobb-Douglas production function can be rewritten as:

$$\log Y = \beta_1 + \beta_2 \log K + \beta_3 \log L + \log(\varepsilon)$$

This model now satisfies A.1 (linear in the parameters), however, it is not always advisable to estimate multiplicative models by LS! Silva and Tenreyro (2006)[1] if $\log(\varepsilon)$ is heteroskedastic[2] (it likely is), then X and $\log(\varepsilon)$ are *not* independent (violation of A.5).

[1]Silva and Tenreyro (2006). The Log of Gravity. *The Review of Economics and Statistics*.

[2]We cover heteroskedasticity in the next chapter.

Polynomial regression model

If model is non-linear in the parameters, can *approximate* the non-linear relationship. Add additional variables to X, that are just non-linear transformations of variables in the original X matrix. Including higher order polynomials of the x variables (i.e. squared and cubed terms), makes f in model 1 approximately linear. The validity of such an approach is based on the Taylor series approximation.

One way to characterize the non-linear relationship between y and x is to say that the marginal effect of x on y depends on the value of x itself. By just including powers of the regressors on the right-hand-side of the regression model (this is not a violation of A.2), we can allow the marginal effect of x on y to vary in a way that depends on x.

For example:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \cdots + \epsilon \tag{2}$$

Taking the derivative of y with respect to x gives:

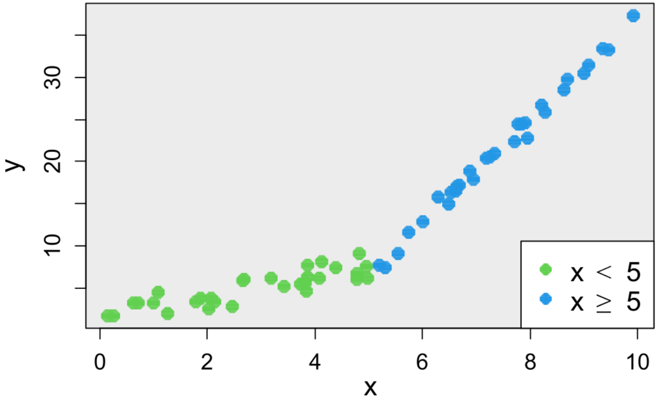
$$\frac{\partial y}{\partial x} = \beta_1 + 2x\beta_2 + 3x^2\beta_3 + \cdots$$

We can choose $\boldsymbol{\beta}$ to give an arbitrarily good approximation of any non-linear relationship $y = f(x)$. The appropriate order of the polynomial may be determined through a series of t-tests, where if the highest order term is found to be insignificant, it is dropped and the model re-estimated.

The drawback of polynomial regression models are that the parameters become difficult to interpret, and the model may not generalize well outside of the data. Often we are interested in a causal effect that is represented by parameters in a model, and so we might want to estimate the non-linear model directly.

Splines

Figure: What model would you specify for this data?



There may be a “break” in the model so that it is “piecewise” linear (see Figure 1 where there is a break at $x = 5$). From section on dummies we know that we can use a dummy variable to fully interact with all of the variables in the model, allowing the intercept and slope coefficients to differ by the value of the dummy. That is, we could define a dummy variable:

$$\begin{aligned} D &= 0 \quad \text{if } x < 5; \\ D &= 1 \quad \text{if } x \geq 5 \end{aligned}$$

and then estimate a model for the data in Figure 1 as:

$$y = \beta_1 + \beta_2 x + \beta_3 D + \beta_4 D x + \epsilon \tag{3}$$

Figure: A dummy variable allows for two separate regression lines to be estimated, on either side of the "break" point. Notice that the two lines do not connect.

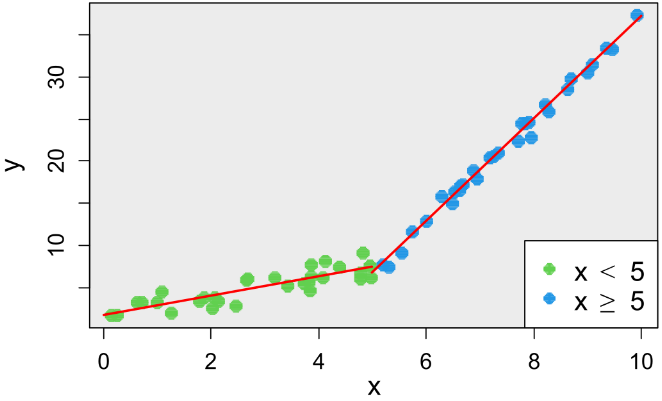
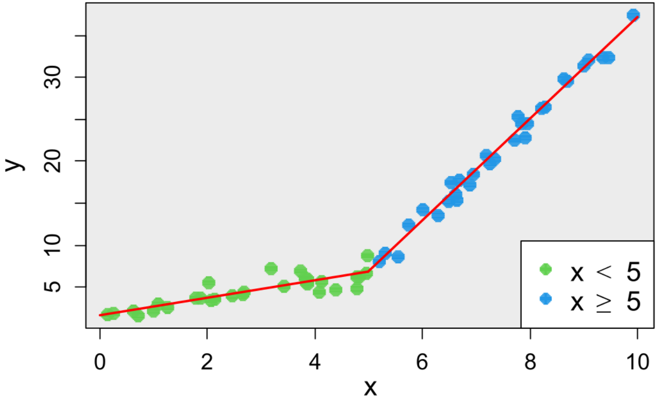


Figure: Restricting the parameter values in a model where a dummy variable fully interacts with every variable can force the piecewise linear function to join at knots. This is known as spline regression.



In the RLS section, we imposed *restrictions* on the parameters in the model. If we want to estimate a *piecewise* linear regression function, then we can *force* the two lines to join at the break-point (called a "knot"). We can do so by imposing a restriction on the model: the y value at the end of the first line should be equal to the y value at the beginning of the second line. That is, the equations for the two separate lines should equal when $x = 5$. Imposing such a restriction gives us the model:

$$y = \beta_1 + \beta_2 x + \beta_4 D(x - 5) + \epsilon \tag{4}$$

Estimating the restricted model 4 is called *spline* regression. We can have multiple "knots" (break points) in the spline regression, where the piecewise linear functions are joined at the knots through a set of restrictions on the parameter values. This model can be extended to allow an arbitrary number of knots, the locations of which can be determined by the data. This extension is known as non-parametric kernel regression (not covered in these notes).

Non-linear least squares

In yet other cases, when the model is inherently non-linear in the parameters, an approximation may be inadequate. In addition, we may desire to estimate the economic model in its natural form, to have a more direct interpretation of the parameter estimates. Then, we must use a *different* estimation methodology, such as non-linear least squares (NLS). An alternative option could be *maximum likelihood.* Take for example a CES production function:

$$Y_i = \gamma \left[\delta K_i^{-\rho} + (1-\delta) L_i^{-\rho}\right]^{-\upsilon/\rho} \exp\left(\epsilon_i\right)$$

This is a function where there is no known transformation to make it linear in the parameters. In general, suppose we have a single non-linear equation:

$$\begin{aligned} y_i &= f\left(x_{i1}, x_{i2}, \ldots, x_{ik}; \theta_1, \theta_2, \ldots, \theta_p\right) + \epsilon_i \\ \boldsymbol{y} &= f(X; \boldsymbol{\theta}) + \boldsymbol{\epsilon} \end{aligned} \tag{5}$$

We can still consider a "least squares" approach to estimate model 5 (which can be motivated by the method of moments). The Non-Linear Least Squares estimator is the vector, $\hat{\boldsymbol{\theta}}$, that minimizes the quantity:

$$S(X, \boldsymbol{\theta}) = \sum_i \left[y_i - f_i(X, \widehat{\boldsymbol{\theta}}) \right]^2$$

(Note that the usual LS estimator is a special case of this). To obtain the estimator, we differentiate S with respect to each element of $\hat{\boldsymbol{\theta}}$; set up the "$p$" first-order conditions, and solve.
We won't derive the results here, but the properties of the NLS estimator are as follows:

- consistent
- asymptotically efficient
- asymptotically Normal

The NLS estimator has good asymptotic properties, however, there is a serious computational difficulty. Usually there is no *exact* solution for $\hat{\boldsymbol{\theta}}$! The first-order conditions are themselves non-linear in the unknown parameters. This is true for the vast majority of estimators in non-linear models, including maximum likelihood. There is (generally) no exact, closed-form solution. That is, we can't write down an explicit formula for the estimators of the parameters.

No closed form solution.

Take the model:

$$y_i = \theta_1 + \theta_2 x_{i2} + \theta_3 x_{i3} + (\theta_2\theta_3)\, x_{i4} + \epsilon_i$$

The sum of squared errors are:

$$S = \sum_i \left[y_i - \theta_1 - \theta_2 x_{i2} - \theta_3 x_{i3} - (\theta_2\theta_3)\, x_{i4}\right]^2$$

and the first-order conditions are:

$$\frac{\partial S}{\partial \theta_1} = -2\sum_i \left[y_i - \theta_1 - \theta_2 x_{i2} - \theta_3 x_{i3} - (\theta_2\theta_3)\, x_{i4}\right]$$
$$\frac{\partial S}{\partial \theta_2} = -2\sum_i \left[(\theta_3 x_{i4} + x_{i2})\,(y_i - \theta_1 - \theta_2 x_{i2} - \theta_3 x_{i3} - \theta_2\theta_3 x_{i4})\right]$$
$$\frac{\partial S}{\partial \theta_3} = -2\sum_i \left[(\theta_2 x_{i4} + x_{i3})\,(y_i - \theta_1 - \theta_2 x_{i2} - \theta_3 x_{i3} - \theta_2\theta_3 x_{i4})\right]$$

Setting these 3 equations to zero, we can't solve analytically for the estimators of the three parameters.

In situations such as above, and the majority of non-linear models, we need to use a *numerical algorithm* to obtain a solution to the first-order conditions. There are lots of methods for doing this, most of which are based on *Newton*'s algorithm.

Taylor series approximation

A Taylor series is an expansion of a function $f(x)$ about a point $x = a$, and is given by:

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \cdots$$

where $f'(a)$ and $f''(a)$ are the first and second derivatives of $f(x)$ evaluated at the point a, for example.
Taylor's theorem says that certain functions can be expressed as a Taylor series. The right-hand-side of the Taylor series becomes an *approximation* when we truncate the infinite sum, for example:

$$f(x) \approx f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2$$

Newton-Raphson algorithm

Suppose we want to minimize some function, $f(\boldsymbol{\theta})$. We can approximate the function using a Taylor's series expansion about $\widetilde{\boldsymbol{\theta}}$, the vector value that minimizes $f(\boldsymbol{\theta})$:

$$f(\boldsymbol{\theta}) \approx f(\widetilde{\boldsymbol{\theta}}) + (\boldsymbol{\theta} - \widetilde{\boldsymbol{\theta}})' \left(\frac{\partial f}{\partial \boldsymbol{\theta}}\right)_{\widetilde{\boldsymbol{\theta}}} + \frac{1}{2!}(\boldsymbol{\theta} - \widetilde{\boldsymbol{\theta}})' \left[\frac{\partial^2 f}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}'}\right]_{\widetilde{\boldsymbol{\theta}}} (\boldsymbol{\theta} - \widetilde{\boldsymbol{\theta}})$$

or:

$$f(\boldsymbol{\theta}) \approx f(\widetilde{\boldsymbol{\theta}}) + (\boldsymbol{\theta} - \widetilde{\boldsymbol{\theta}})' g(\widetilde{\boldsymbol{\theta}}) + \frac{1}{2!}(\boldsymbol{\theta} - \widetilde{\boldsymbol{\theta}})' H(\widetilde{\boldsymbol{\theta}})(\boldsymbol{\theta} - \widetilde{\boldsymbol{\theta}})$$

Now, the derivative of the function $f(\boldsymbol{\theta})$ that we want to minimize is:

$$\frac{\partial f(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \approx 0 + (\boldsymbol{\theta} - \widetilde{\boldsymbol{\theta}})' g(\widetilde{\boldsymbol{\theta}}) + \frac{1}{2!} 2H(\widetilde{\boldsymbol{\theta}})(\boldsymbol{\theta} - \widetilde{\boldsymbol{\theta}})$$

However, the gradient at the minimum located by $\widetilde{\boldsymbol{\theta}}$ is zero $(g(\widetilde{\boldsymbol{\theta}}) = 0)$, so:

$$(\boldsymbol{\theta} - \widetilde{\boldsymbol{\theta}}) \approx H^{-1}(\widetilde{\boldsymbol{\theta}}) \left(\frac{\partial f(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right)$$

or

Approximate location of the minimum

$$\widetilde{\boldsymbol{\theta}} \approx \boldsymbol{\theta} - H^{-1}(\widetilde{\boldsymbol{\theta}}) g(\boldsymbol{\theta})$$

The above equation suggests a numerical algorithm: set $\boldsymbol{\theta} = \boldsymbol{\theta}_0$ to begin, and then iterate:

$$\begin{gathered}
\boldsymbol{\theta}_1 = \boldsymbol{\theta}_0 - H^{-1}\left(\boldsymbol{\theta}_1\right) g\left(\boldsymbol{\theta}_0\right) \\
\boldsymbol{\theta}_2 = \boldsymbol{\theta}_1 - H^{-1}\left(\boldsymbol{\theta}_2\right) g\left(\boldsymbol{\theta}_1\right) \\
\vdots \\
\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n - H^{-1}\left(\boldsymbol{\theta}_{n+1}\right) g\left(\boldsymbol{\theta}_n\right)
\end{gathered}$$

or, *approximately*:

NR algorithm

$$\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n - H^{-1}\left(\boldsymbol{\theta}_n\right) g\left(\boldsymbol{\theta}_n\right) \tag{6}$$

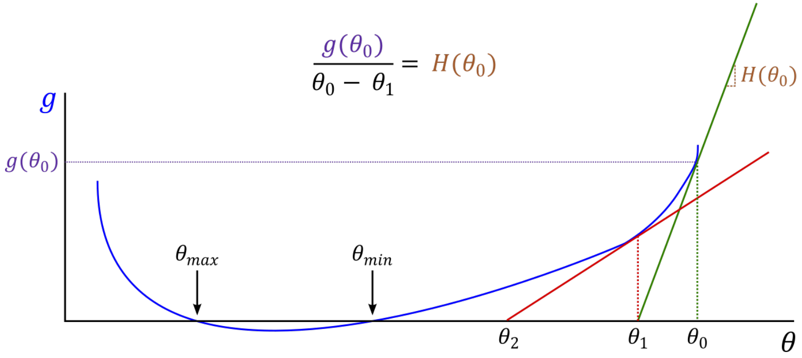
The algorithm stops at *convergence* - when the difference between iterations is within a certain *tolerance*, that is, if:

$$\left| \frac{\left(\theta_{n+1}^{(i)} - \theta_n^{(i)}\right)}{\theta_n^{(i)}} \right| < \mathcal{E}^{(i)}; \quad i = 1, 2, \ldots, p$$

Note:

- The algorithm fails if H ever becomes *singular* at any iteration.
- The algorithm will achieve a minimum if H is positive definite.
- The algorithm may locate only a local minimum.
- The algorithm may oscillate.

Figure: Illustration of Newton-Raphson algorithm for scalar θ.



Example of Newton's method

Consider locating the minimum of the following function:

$$f(\theta) = 3\theta^4 - 4\theta^3 + 1$$

We can actually solve for the minimum analytically (the solution is 1):

$$\begin{aligned} g(\theta) &= 12\theta^3 - 12\theta^2 = 12\theta^2(\theta - 1) \\ H(\theta) &= 36\theta^2 - 24\theta = 12\theta(3\theta - 2) \end{aligned}$$

Let's instead use the algorithm. Choose an initial value $\theta_0 = 2$ for example. Then:

$$\begin{aligned} \theta_1 &= 2 - \left(\frac{48}{96}\right) = 1.5 \\ \theta_2 &= 1.5 - \left(\frac{13.5}{45}\right) = 1.2 \\ \theta_3 &= 1.2 - \left(\frac{3.456}{23.040}\right) = 1.05 \end{aligned}$$

We can see the solution converging to 1. What if we try $\theta_0 = -2$?

NLS in R

In this example, we generate data according to the following non-linear model:

$$y = \beta_1 x^{\beta_2} + \epsilon$$

Set some parameter values and generate some data:

```r
set.seed(1)
n <- 50
x <- rnorm(n, 5, 2)
beta1 <- 3
beta2 <- 2.5
y <- beta1 * x ^ beta2 + rnorm(n, 0, 40)
```

Next, estimate the model using NLS:

```r
mod <- nls(y ~ b1 * x ^ b2, start = list(b1 = 0, b2 = 0))
summary(mod)
```

```
Formula: y ~ b1 * x^b2

Parameters:
   Estimate Std. Error t value Pr(>|t|)
b1   2.6193     0.6124   4.277 8.97e-05 ***
b2   2.5618     0.1230  20.824  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' '
     1

Residual standard error: 36.46 on 48 degrees of freedom

Number of iterations to convergence: 9
Achieved convergence tolerance: 1.443e-07
```

Notice that we need to choose starting values for the parameters, and that the estimation output refers to iterations, convergence, and tolerance.

The Log of Gravity

Beginning in 1962, many theories of trade, even when based on different foundations, all predicted a gravity relationship for trade flows (analogous to Newton's law of gravitation):

$$T_{ij} = \alpha_0 Y_i^{\alpha_1} Y_j^{\alpha_2} D_{ij}^{\alpha_3} \eta_{ij} \tag{7}$$

The gravity model in equation 7 suggests that trade flow from i to j is directly proportional to the GDP of i and j (Y_i and Y_j), and inversely proportional to their (broadly-defined) distance (D_{ij}). $\alpha_0, \alpha_1, \alpha_2, \alpha_3$ are the parameters to be estimated. η_{ij} is an error term with $E\left(\eta_{ij} \mid Y_i, Y_j, D_{ij}\right) = 1$, which is independent from the regressors.

For decades, equation 7 was log-linearized (as we have done with the Cobb-Douglas model) so that LS can be used to estimate the unknown parameters:

$$\begin{aligned} \ln T_{ij} &= \ln \alpha_0 + \alpha_1 \ln Y_i + \alpha_2 \ln Y_j + \alpha_3 \ln D_{ij} + \ln \eta_{ij} \\ &= \beta_0 + \beta_1 \ln Y_i + \beta_2 \ln Y_j + \beta_3 \ln D_{ij} + \epsilon_{ij} \end{aligned} \tag{8}$$

An immediate issue with log-linearizing the gravity equation is that trade flows of zero ($T_{ij} = 0$) are not allowed, since $\ln 0$ is undefined. The practice was to drop observations with 0 trade flows from the data set, even though in many cases there were lots of 0s in the data.

Further to the problem of 0s in the data, in a famous paper called "The log of gravity"[3], Silva and Tenreyo (2006) showed that the LS estimator of equation 8 is biased and inconsistent in the fairly common situation where the error term η_{ij} is heteroskedastic (we cover heteroskedasticity in the next chapter). This was surprising; although η_{ij} may be independent from the regressors, $\ln \eta_{ij}$ is generally not independent. This is a violation of A.5, suggesting that LS should not be used, and that the gravity equation in 7 should be estimated in it's non-linear form (NLS can be used, but Silva and Tenreyo actually propose a more efficient estimator in their paper).

[3]Silva, J. S., & Tenreyro, S. (2006). The log of gravity. The Review of Economics and Statistics, 88(4), 641-658.

Figure: A table of estimates from Silva and Tenreyo (2006). I have edited the table, removing three columns.

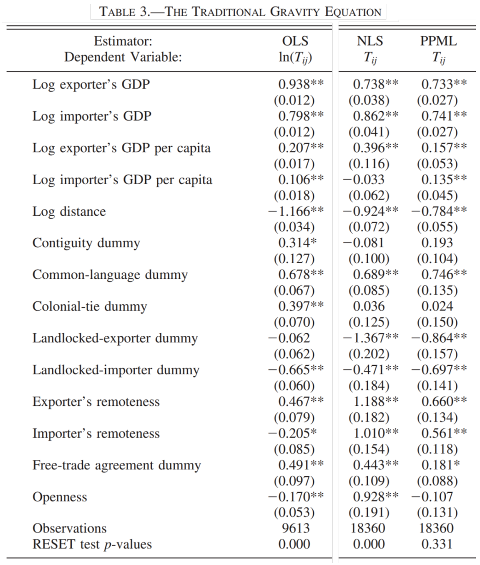


TABLE 3.—THE TRADITIONAL GRAVITY EQUATION

Estimator: Dependent Variable:	OLS $\ln(T_{ij})$	NLS T_{ij}	PPML T_{ij}
Log exporter's GDP	0.938**	0.738**	0.733**
	(0.012)	(0.038)	(0.027)
Log importer's GDP	0.798**	0.862**	0.741**
	(0.012)	(0.041)	(0.027)
Log exporter's GDP per capita	0.207**	0.396**	0.157**
	(0.017)	(0.116)	(0.053)
Log importer's GDP per capita	0.106**	−0.033	0.135**
	(0.018)	(0.062)	(0.045)
Log distance	−1.166**	−0.924**	−0.784**
	(0.034)	(0.072)	(0.055)
Contiguity dummy	0.314*	−0.081	0.193
	(0.127)	(0.100)	(0.104)
Common-language dummy	0.678**	0.689**	0.746**
	(0.067)	(0.085)	(0.135)
Colonial-tie dummy	0.397**	0.036	0.024
	(0.070)	(0.125)	(0.150)
Landlocked-exporter dummy	−0.062	−1.367**	−0.864**
	(0.062)	(0.202)	(0.157)
Landlocked-importer dummy	−0.665**	−0.471**	−0.697**
	(0.060)	(0.184)	(0.141)
Exporter's remoteness	0.467**	1.188**	0.660**
	(0.079)	(0.182)	(0.134)
Importer's remoteness	−0.205*	1.010**	0.561**
	(0.085)	(0.154)	(0.118)
Free-trade agreement dummy	0.491**	0.443**	0.181*
	(0.097)	(0.109)	(0.088)
Openness	−0.170**	0.928**	−0.107
	(0.053)	(0.191)	(0.131)
Observations	9613	18360	18360
RESET test *p*-values	0.000	0.000	0.331

Estimate gravity by LS

Let's reproduce the first column in the table above. Load the gravity data using:

```r
grav <- read.csv("https://rtgodwin.com/data/gravity.csv")
```

Drop all observations where trade is zero (that's almost half the sample!) and estimate the model by log-linearizing and using LS:

```r
grav.no.zeros <- subset(grav, grav$trade > 0)
m1 <- lm(log(trade) ~ lypex + lypim + lyex + lyim + ldist
      + border + comlang + colony + landl_ex + landl_im
      + lremot_ex + lremot_im + comfrt_wto + open_wto,
         data = grav.no.zeros)
summary(m1)
```

```
Coefficients:
               Estimate Std. Error t value Pr(>|t|)    
(Intercept) -28.49202    1.08804 -26.187  < 2e-16 ***
lypex         0.93782    0.01163  80.624  < 2e-16 ***
lypim         0.79779    0.01110  71.881  < 2e-16 ***
lyex          0.20731    0.01664  12.462  < 2e-16 ***
lyim          0.10613    0.01670   6.355 2.18e-10 ***
ldist        -1.16601    0.03391 -34.390  < 2e-16 ***
border        0.31400    0.14252   2.203 0.027603 *  
comlang       0.67804    0.06398  10.597  < 2e-16 ***
colony        0.39680    0.06810   5.827 5.84e-09 ***
landl_ex     -0.06197    0.06459  -0.959 0.337363    
landl_im     -0.66452    0.06313 -10.526  < 2e-16 ***
lremot_ex     0.46707    0.07776   6.007 1.96e-09 ***
lremot_im    -0.20496    0.08077  -2.538 0.011177 *  
comfrt_wto    0.49082    0.10533   4.660 3.20e-06 ***
open_wto     -0.16964    0.04902  -3.460 0.000542 ***
---
Signif.codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.9 on 9598 degrees of freedom
Multiple R-squared:  0.6623,  Adjusted R-squared:  0.6618
F-statistic:  1345 on 14 and 9598 DF,  p-value: < 2.2e-16
```

Estimate gravity by NLS

Taking logs and using LS is a bad idea (bias and inconsistency), so we should *undo* the logs:

$$\begin{aligned} T_{ij} &= \alpha_0 Y_i^{\alpha_1} Y_j^{\alpha_2} D_{ij}^{\alpha_3} \eta_{ij} \\ \ln T_{ij} &= \beta_0 + \beta_1 \ln Y_i + \beta_2 \ln Y_j + \beta_3 \ln D_{ij} + \ln \eta_{ij} \\ \exp(\ln T_{ij}) &= \exp(\beta_0 + \beta_1 \ln Y_i + \beta_2 \ln Y_j + \beta_3 \ln D_{ij} + \ln \eta_{ij}) \\ \boldsymbol{T} &= \exp(X\boldsymbol{\beta}) + \boldsymbol{\epsilon} \end{aligned} \tag{9}$$

We write the gravity model in this way[4] so that we can see that NLS is applicable (it matches $\boldsymbol{y} = f(X; \boldsymbol{\beta}) + \boldsymbol{\epsilon}$ in equation 5), and so that we see that we can ask the computer to find $\boldsymbol{\beta}$ in $\exp(X\boldsymbol{\beta})$ instead of α in $\alpha_0 Y_i^{\alpha_1} Y_j^{\alpha_2} D_{ij}^{\alpha_3}$. Asking for $\boldsymbol{\beta}$ is much easier than the α (it is very common to make transformations in non-linear estimation).

[4]We have used $\eta_{ij} = 1 + \epsilon_{ij} / \exp(x_{ij}\beta)$ (see Silva and Tenreyro (2006)) to get to the last line.

To estimate the model by NLS, on the RHS we put the linear equation inside of the exponent: $\exp(X\boldsymbol{\beta})$.[5] In the R code below, notice that we need to choose starting values for each of the parameters:

```r
m2 <- nls(trade ~ exp(b0 + b1 * lypex + b2 * lypim
    + b3 * lyex + b4 * lyim + b5 * ldist
    + b6 * border + b7 * comlang + b8 * colony
    + b9 * landl_ex + b10 * landl_im + b11 * lremot_ex
    + b12 * lremot_im + b13 * comfrt_wto + b14 * open_wto),
  data = dat,
  start = list(b0 = 0, b1 = 0.5, b2 = 0.5, b3 = 0.5,
    b4 = 0.5, b5 = 0.5, b6 = 0.5, b7 = 0.5,
    b8 = 0.5, b9 = 0.5, b10 = 0.5, b11 = 0.5,
    b12 = 0.5, b13 = 0.5, b14 = 0.5))
summary(m2)
```

[5]This is a common “link” function to use when the LHS variable needs to be non-negative.

```
Parameters:
      Estimate Std. Error  t value Pr(>|t|)
b0  -45.098657   0.239150 -188.579  < 2e-16 ***
b1    0.737755   0.004358  169.282  < 2e-16 ***
b2    0.861871   0.004517  190.811  < 2e-16 ***
b3    0.395645   0.009664   40.940  < 2e-16 ***
b4   -0.032511   0.006660   -4.881 1.06e-06 ***
b5   -0.923709   0.008487 -108.844  < 2e-16 ***
b6   -0.081309   0.009858   -8.248  < 2e-16 ***
b7    0.689402   0.015900   43.358  < 2e-16 ***
b8    0.035797   0.017787    2.013   0.0442 *
b9   -1.367068   0.030514  -44.802  < 2e-16 ***
b10  -0.471462   0.022312  -21.130  < 2e-16 ***
b11   1.187801   0.018258   65.056  < 2e-16 ***
b12   1.009682   0.017882   56.462  < 2e-16 ***
b13   0.442547   0.013734   32.224  < 2e-16 ***
b14   0.928022   0.023823   38.955  < 2e-16 ***
---
Signif.codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 567000 on 18345 degrees of freedom

Number of iterations to convergence: 47
Achieved convergence tolerance: 6.762e-06
```