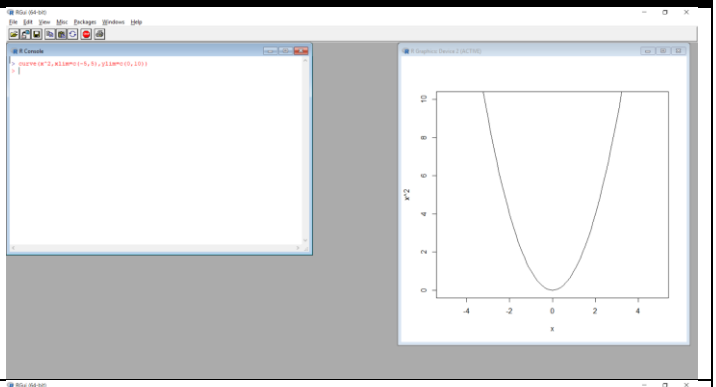
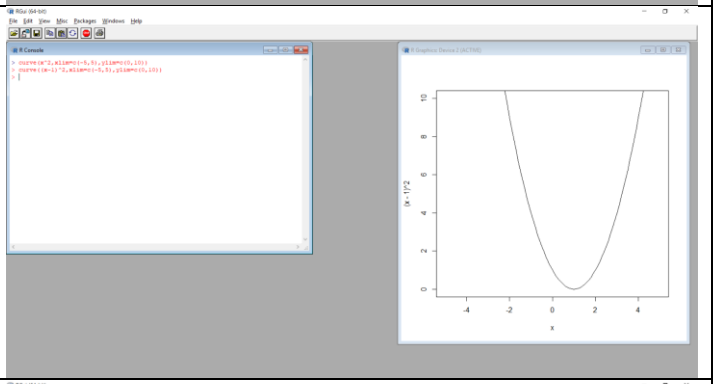
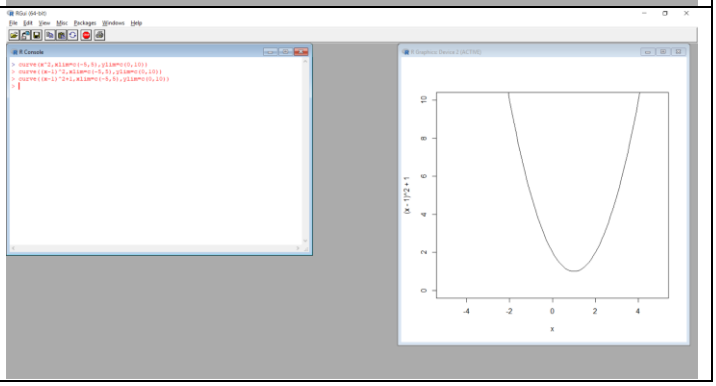
R-Programming Fundamentals for Business Students — Multiple Non-Linear Regressions

Nick V. Flor, University of New Mexico (nickflor@unm.edu)

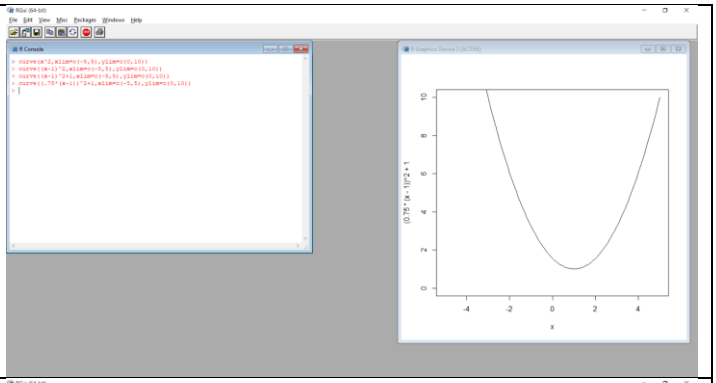
Assumptions. This tutorial assumes (1) that you have an Excel worksheet with the following columns: *pressure*, *performance*;(2) that you saved that file as *bcurve.csv* and, finally, (3) that you ran R and did a *File > Change dir…* to the folder containing *bcurve.csv*

ACTION	REACTION
INTERACTIVE INTUITION BEHIND CURVE FITTING	
• Type **`curve(x^2,xlim=c(-5,5),ylim=c(0,10))`** Explanation: Take any 2-D curve, y=f(x). In our case $y=x^2$ (denoted as **x^2** in R) which you probably recognize from algebra as the equation for a parabola. The **curve** command plots the curve. The parameters **xlim** & **ylim** set the dimensions for the x & y axes.	
• Type **`curve((x-1)^2, xlim=c(-5, 5), ylim=c(0,10))`** Explanation: y=f(x+lrshift). *Any curve can be shifted left or right by adding/subtracting some value to x.* Note: It's counter-intuitive, but to shift right, you subtract from x, to shift left (not shown), you add to x. Also keep in mind that subtracting is the same as adding a negative number. IMPORTANT: NOTE THE PARENTHESES.	
• Type **`curve((x-1)^2+1,xlim=c(-5,5),ylim=c(0,10))`** Explanation: y=f(x+lrshift)+udshift. *Any curve can be shifted up or down by adding/subtracting some value at the end of the function.*	

- Type **`curve((.75*(x-1))^2+1,xlim=c(-5,5),ylim=c(0,10))`**

Explanation: y=f(fat*(x+lrshift))+udshift. *Any curve can be made "fatter" or "skinnier" by multiplying the x (+shift) by some value.*

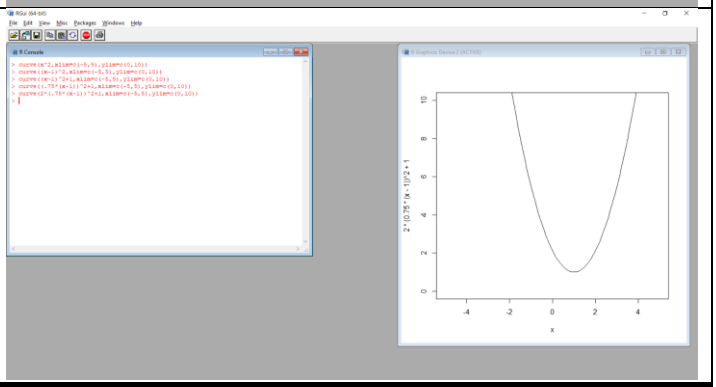
Note: This value is counter-intuitive. Multiply by a number <1 makes the curve "fatter", >1 "skinnier".



- Type **`curve(2*(.75*(x-1))^2+1,xlim=c(-5,5),ylim=c(0,10))`**
-

Explanation: y=tall*f(fat*(x+lrshift))+udshift. *Any curve can be made "taller" or "shorter" by multiplying the function by some value.*

Note: This is harder to see for some functions like parabolas—it looks skinnier w/a parabola, whereas with a bell curve you can see it controls "tallness".



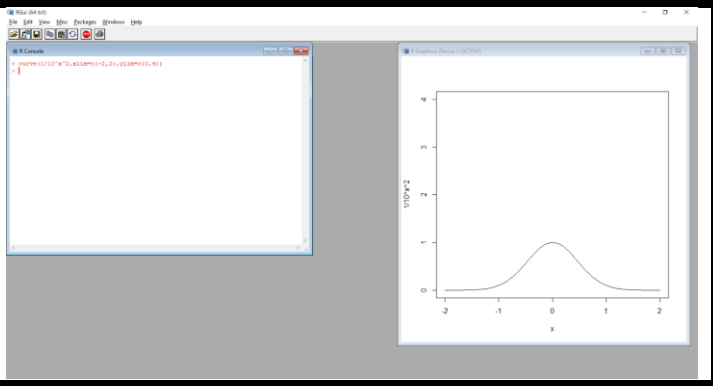
APPLICATION, MODELING A BELL CURVE DATASET — STEP 0: KNOW YOUR CURVES

- **`curve(1/10^x^2, xlim=c(-2,2), ylim=c(0,4))`**

Explanation: There are many equations for a bell-shaped curve. I "invented" the following:

$$y = \frac{1}{10^{x^2}}$$

… or $y = 10^{-x^2}$. In R, you can type this as **`1/10^x^2`**.
Warning: R does exponent precedence from right to left, so typing this in Excel will not work without placing parentheses around (x^2).

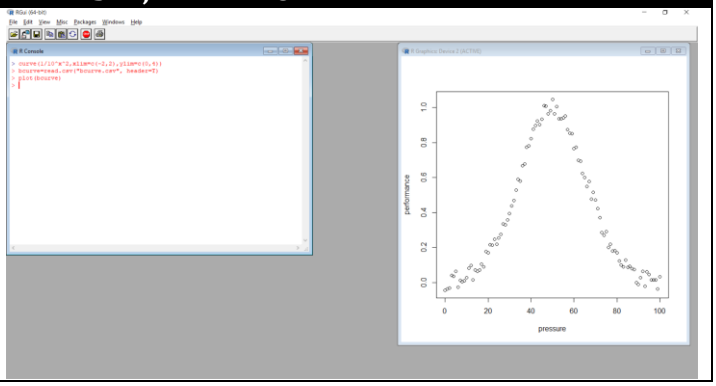


STEPS 1 & 2: READ IN YOUR DATASET, THEN PLOT IT

- **`bcurve=read.csv("bcurve.csv", header=T)`**
- **`plot(bcurve)`**

Explanation: The first command reads the csv file into the variable **bcurve**. The second plots the dataset.

VERY IMPORTANT NOTE: In the plot, the x-variable is named *pressure*, the y-variable *performance*.



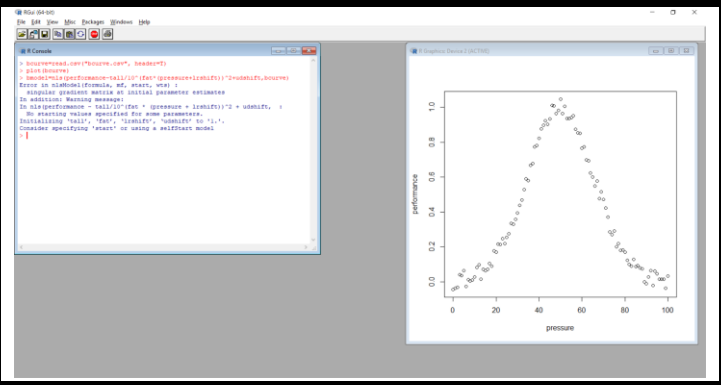
STEP 3. SELECT CURVE EQUATION & MODEL CURVE USING NLS

- `bmodel=nls(performance~tall/10^(fat * (pressure + lrshift))^2+udshift,bcurve)`

Explanation: Curve equation is the bell-curve formula. **bmodel** is just a variable. See Interactive Intuition section for **tall**, **fat**, **lrshift**, **udshift**;

bcurve is the dataset you read in Step 1.

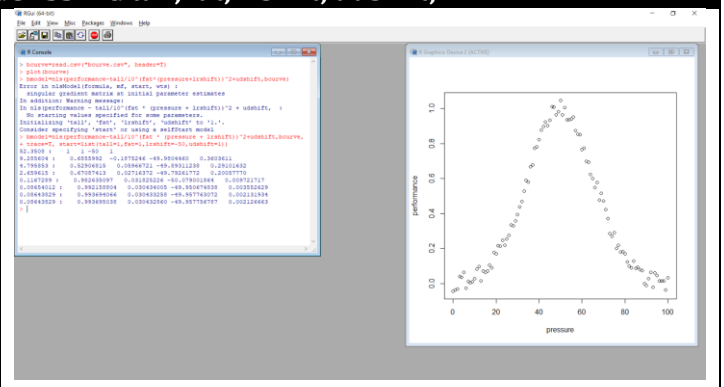
Note: If you do NOT get an *Error* message about *singular gradient,* this is a good thing. Go to step 4.



STEP 3A (IF ERROR) HELP R BY GUESSING tall, fat, lrshift, udshift,

- `bmodel=nls(performance~tall/10^(fat * (pressure + lrshift))^2+udshift,bcurve, trace=T, start=list(tall=1, fat=1, lrshift=-50, udshift=1))`

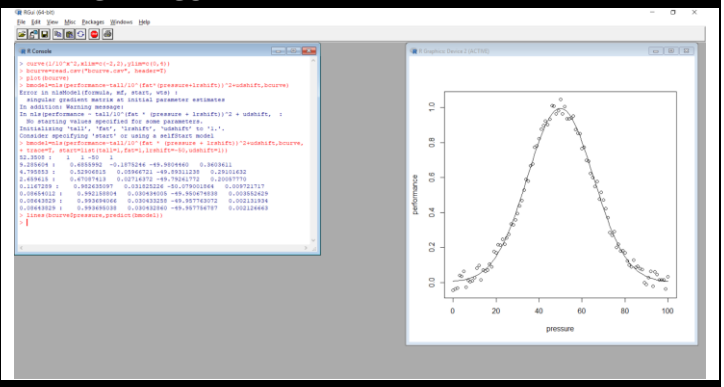
Explanation: **trace** adds debug information; **start** tells R to use your starting values instead of 1's for everything. You must declare starting values for all variables **tall**, **fat**, **lrshift**, **udshift**. BUT USUALLY lrshift is the culprit, so I keep everything at 1, and change **lrshift** based on an examination of the original curve versus the dataset curve. Remember a right shift is a negative value, e.g. **-50**



STEP 4: DRAW THE PREDICTED CURVE

- `lines(bcurve$pressure,predict(bmodel))`

Explanation: **lines** draws over the existing plot. You want to use the dataset x-values, **bcurve$pressure**, and the model's y-values, **predict(bmodel)**.

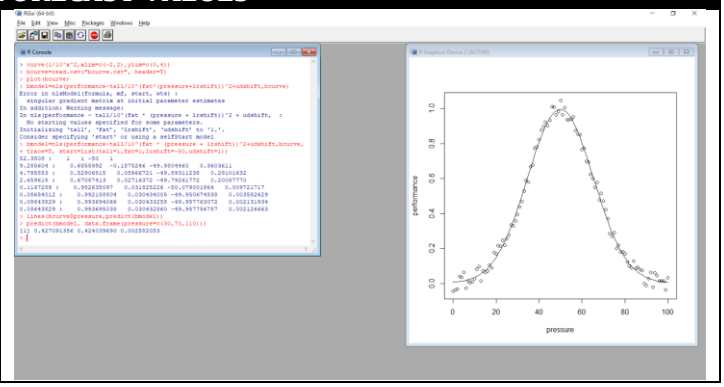


STEP 5: PREDICT/FORECAST VALUES

- `predict(bmodel, data.frame(pressure=c(30, 70, 110)))`

Explanation: Similar to lm, with an nls you use the **predict** function on the model, **bmodel**, specifying the variable, **pressure**, and values in a column, **c=(30, 70, 110).**

In our example, pressures of 30, 70, and 110 result in performances of 42.7%, 42.4%, and .25%



STEP 6 & 7 (OPTIONAL): GET MODEL COEFFICIENTS & DETERMINE EQUATION

- `summary(bmodel)`

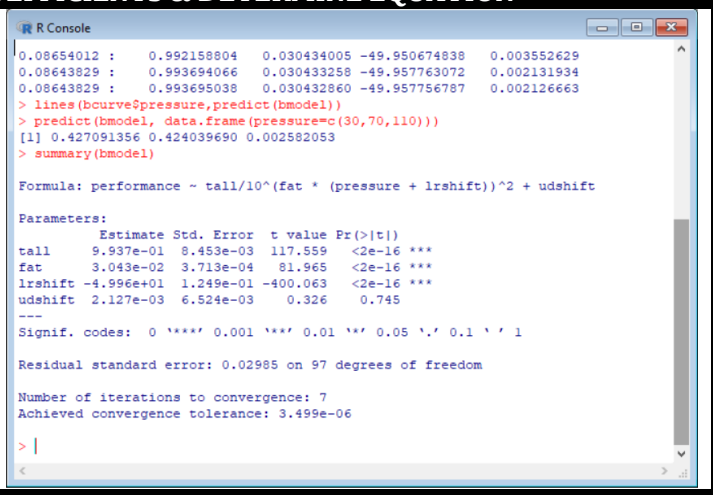
Explanation: the summary function gives you the coefficients for your equation and their p-values. Anything less than .05 is significant.

The equation is

$$performance = \frac{.994}{10^{(.030(pressure-49.96))^2}} + .002$$

Or rounded simply:

$$performance = \frac{1}{10^{(.030(pressure-50))^2}}$$

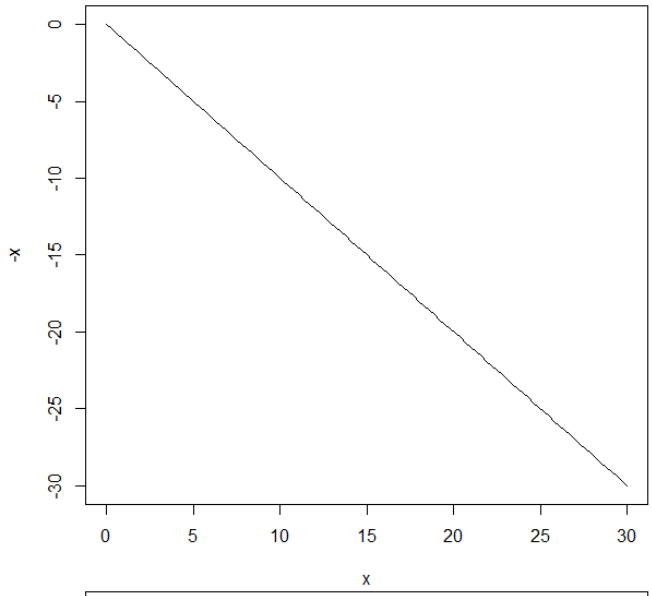


OTHER USEFUL CURVES (THE GREAT CURVES OF MANAGEMENT)

Downward Linear

$$y = -x$$

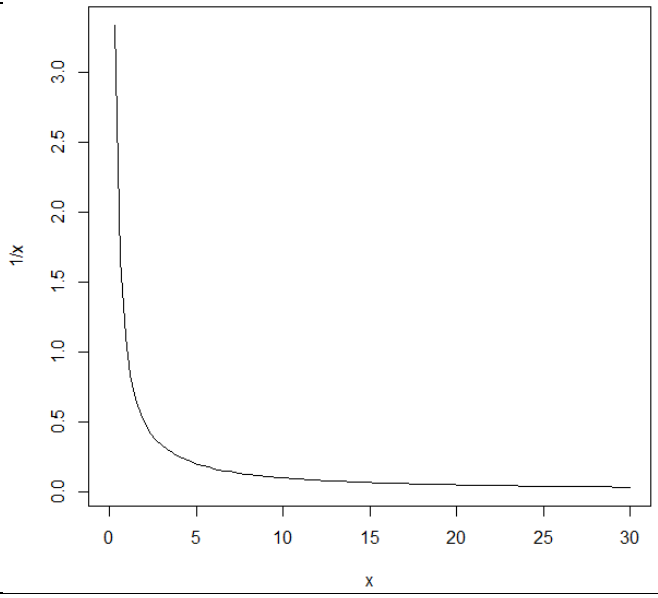
R Example: `curve(-x,0,30)`



Inverse Power Curve

$$y = \frac{1}{x}$$

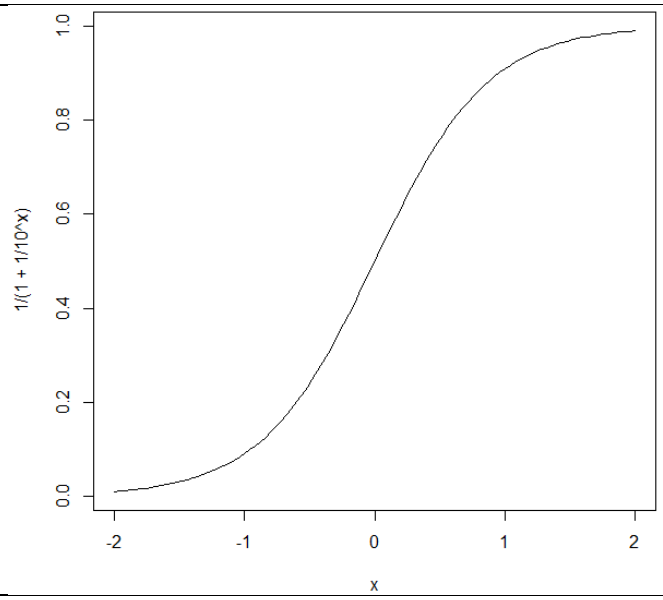
R Example: `curve(1/x,0,30)`



S-Curve (Sigmoid)

$$y = \frac{1}{1 + \frac{1}{10^x}}$$

R Example: **`curve(1/(1+1/10^x),-2,2)`**



Log Normal (my personal favorite for modeling viral spikes on social media)

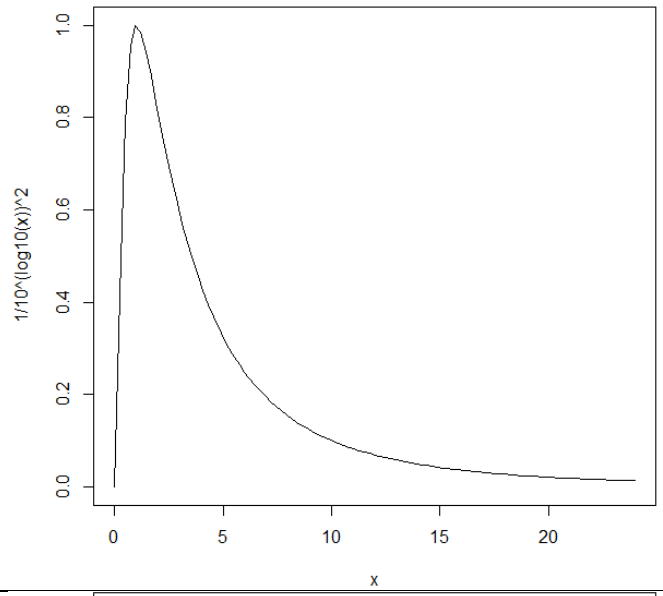
$$y = \frac{1}{10^{(\log_{10} x)^2}}$$

R Example: **`curve(1/10^(log10(x))^2,0,24)`**

Note: This curve is the exception to the rule, I model it as:

$$y = \frac{tall}{10^{(\log_{fat} \frac{x}{shift})^2}}$$

R Implementation: **`nls(y~tall/10^((log(x/shift) / log(b))^2,ncurve)`**



Log Normal (cont)