Lab 7: Curvy data

Entomology 5126 - Spring 2026

Jeremy Hemberger and Brian Aukema

February 23, 2026

Table of contents

---

1 Introduction

Today, we'll work on regression techniques for curvy data. Sometimes, when you first graph the data, you can see instantly that there is an asymptote, or a gentle curve, or a large squiggle, or…. whatever it is, a straight line will not be a good model. And let's face it, there are very few patterns in ecology that are straight lines! Today, we will focus on two simple techniques:

- Transforming an independent (x) variable. We fit a straight line to data, that, when back-transformed, becomes curved!
- Polynomial regression. This technique is a special type of multiple regression. Of course, you are good at multiple regression now.

2 Fitting curves

2.1 Transforming the x variable

Decay functions and asymptotes are frequently seen in dose-response curves where increasing doses have decreasing biological effects, or perhaps an effect decays through time. One of the easiest things to do is transform the x variable, fit a regular regression line, and then back-transform.

reciprocal \$x\$ transformation	logarithmic \$x\$ transformation
$y_i = \beta_0 + \beta_1(1/x_i) + \epsilon_i$	$y_i = \beta_0 + \beta_1 \log(x_i) + \epsilon_i$

2.2 Polynomial regression

The basic formula for polynomial regression is

$$y_i = \beta_0 + \beta_1 x + \beta_2 x^2 + ... + \beta_k x^k + \epsilon_i$$

The β_0 is the intercept, or where the line crosses the y-axis. The β_1 is the slope of the first covariate, the "linear" term, or the increase in y per unit change in x *holding all other covariates constant*. The β_2 is the slope of the second covariate, the "quadratic" term. The ϵ_i refers to the "errors" about the line, and we assume that they are normally distributed with a constant variance (i.e., $\epsilon_i \sim N(0, \sigma^2)$).
An important note when reporting a polynomial regression line: by convention, we always report "lower order" terms, even if they are not significant. So, if you have a quadratic equation, for example, where only the quadratic (i.e., squared) covariate is significant, we would still include the intercept and the linear term when reporting the equation of the line, even if they are not significantly different from zero.

2.3 Checking assumptions

Both techniques have the same linear model structure. As such, you are familiar with the assumptions:

1. The model is correct
2. The observations are independent
3. The variances are equal
4. The errors are normally distributed

As you might expect, you can check the assumptions of equal variances and normally distributed errors in your residual plot, as well as detecting if your model is perhaps misspecified.

2.4 Transforming a covariate in `R`

Both of these techniques involve transforming a covariate. For example, in polynomial regression, you need to square a covariate to get the quadratic term. When doing this in `R`, you need to use the `identity` specifier, `I()`. That's just a fancy way of telling the program to transform and protect the covariate. For example:

```
> fm1 <- lm(y ~ log(x), ...) # standard function, no identity protection needed
> fm2 <- lm(y ~ I(1/x), ...) # add data argument as before
> fm3 <- lm(y ~ x + I(x^2), ...) # quadratic function
```

2.5 Other techniques (FYI)

There are *many* other techniques for fitting curves: loess fitting (also known as *locally weighted polynomial regression* or *locally weighted regression scatterplot smoothing*), smoothing splines, and *nonlinear* models.

For example, the asymptotic decay function below

could also be fit using a nonlinear model like this:

$y(x) = \phi_1 + (\phi_2 - \phi_1)\text{exp}[-\text{exp}(\phi_3)x]$

I am not going to cover nonlinear methods in this course. More often than not, the practitioner fits a nonlinear model because it looks good and the software offered the option (maybe the button was shiny?). Keeping with a theme in the course, **DON'T do something unless you can explain what you did and why you did it**. From the example above, one may arrive at a nice line, but if you ask what the estimate of ϕ_3 represents (is it an intercept? a slope?), the practitioner may not have a clue. That's not the way you want to conduct your analyses. (In the above example, it's actually the logarithm of the rate constant. A half-life could be estimated $t_{0.5} = \log 2/\exp(\phi_3)$).

The nice thing about using a "regular" regression line (even with a transformed x variable) is that the form of the equation is still very straightforward.

3 Today's data

Today's data set consists of the normal mean minimum temperatures in January (based on 30 years of data from 1931-1960) for 56 cities across the continental United States. It can be found in Hand, D.J, et al. (1984) *A Handbook of Small Data Sets*, London: Chapman Hall, 208-210.

Variable Description

City Yeah, that State State postal abbreviation JanTemp Average minimum January temperature in degrees Farenheit Lat Latitude; degrees north of the equator Lon Longitude; degrees west of the prime meridian

4 Today's Assignment

1. First, let's analyze some mock data. To get it into `R`, you can put it into an Excel spreadsheet for export, or enter it directly.

   ```
   > x <- 1:10 # or, x <- c(1,2,3,4,5,6,7,8,9,10) is same thing
   > y <- c(10,5,3.5,1.5,1,1,0.9,0.9,0.5,0.4)
   > mydata <- data.frame(x=x, y=y) # make dataframe with these
   >                                #  numbers, use same titles
   > mydata
   ...
   ```

   Because it is second nature, I do not need to remind you to plot the data (y vs. x) and think about what sort of model to fit. Because the only regression line you have learned how to fit up to this point is a straight line, go ahead and do so now (use the `lm()` function as before).

   Report your results. Do *not* simply reproduce the `summary()` table: tell me whether the model is significant (F, P-values?) and comment on the R^2 value. If you had not initially graphed the data, would you be satisfied with this result?

2. Examine and comment on the residual plot, particularly on any analytical assumptions that might be violated.

3. : This is where you exercise the creative, right half of your brain after a perhaps stressful midterm. Connect the dots in your residual plot and draw something that will make Audrey and I smile when marking the labs. You may add a few sentences of explanation if helpful.

4. Now fit a reciprocal transformation of the x variable (a decay fit). Examine and comment on the residual plot from the reciprocal transformation and the analytical results. Is this any better than the straight line fit?

5. Let's try a different sort of nonlinear curve. Create a new response variable by subtracting all the values from 10 as follows:

   ```
   > mydata$new_y <- 10 - mydata$y
   ```

   Plot the data, and fit a straight regression line (you do not need to include this in your assignment). You will notice that the residual plot indicates a poor model fit.
   Fit a logarithmic transformation of x instead. Does the R^2 value improve? Have we captured more of the variation than using a simple straight line?

   (Actually, the residual plot still suggests that we have not fully captured the curve, but, for teaching and learning purposes, set this fact aside for now.)

6. Ok, as we have covered, one of the nice things about the previous transformations is that we are still using a simple, intuitive regression line

   $y_i \sim \beta_0 + \beta_1 x_i + \epsilon_i$, where $\epsilon_i \sim N(0, \sigma^2)$

   The covariate may be transformed, but the form of the equation is still pretty straightforward.

   That doesn't, however, prevent some of us from being easily confused: what *does* the line of best fit look like, and how do we plot it?

   You have an x data range of 1 to 10. Let's say we want the predicted value ($\hat{y}$) at $x = 5.5$. The natural log of 5.5 is 1.704748. If I multiply that number by the slope, and then add the intercept, I get my new y value to graph against $x = 5.5$. Simple.

   But, if I was really graphing this, I wouldn't want to do all these calculations by hand. There *has* to be an easier way to do this, right?

   Of course there is. In `R`, you can extract the fitted value for each value of x by using `fitted.values(model)`. For example,

   ```
   > plot(mydata$new_y ~ mydata$x) # plot data
   > fitted.values(fm2) # model with log(x) from previous question
   > lines(fitted.values(fm2)) # overplot y-hat
   ```

That method is useful for predicting values at all of the existing values of your x variable in the dataset. Right now, those values are 1 to 10. If I was going to add a line of best fit in my graphing software (SigmaPlot, Excel, whatever), I might specify a range from 0 to 10, by increments of 0.1 (or even finer, so it is a nice smooth curve). A more complex method, but useful for getting your estimated $\hat{y}$ for *any* value of x (instead of at only the existing x values in your data set), is to use the `predict()` command. First, create a data frame of what x values you want to use for predictions. It can be as long or as short as you want (continued next page).

```
> new.data <- data.frame(x=1:100/10) # what does this do?
> new.data                           # oh! cool!
```

It is **critical** to name the x variable the *same name as the x* covariate in your model. Otherwise, the prediction command, referring to the model fit, won't have a bloomin' clue what covariate you are talking about. So, if you fit `y~temperature` in your model formula, you should be naming the column in the data frame above `temperature` and not `x`, ok? To continue the example,

```
> mypredictions <- predict(fm2, new.data, se.fit=T)
```

This will work for *any* regression line (as long as you specify the right covariates). If you are using multiple regression, specify multiple new columns in your data frame, corresponding to the desired names of your covariates.

The resulting object, `mypredictions`, yields four different internal objects of varying lengths as part of the output. For example, the predictions store information about how many degrees of freedom were in the error term of the original model and the residual scale. We are usually most interested in the `fit`, and the standard errors of the predicted y-values, the `se.fit`.

Let's place them in columns alongside the new x values:

```
> new.data$y.hats <- mypredictions$fit
> new.data$y.hat.SEs <- mypredictions$se.fit
> new.data
```

Provide the $\hat{y}$ and the associated prediction error for a new x value at $x = 3.5$. (Please use the line from question 5, the $\log(x)$ transform, and not the reciprocal transform from question 4).

7. Finally, we get to the real data for today; we turn our attention to polynomial fitting. Read in the data, and do a scatterplot matrix.
We are finally working with spatial data! First, what strikes you as strange in the scatterplot of `Lat` on the y axis and `Long` on the x axis? Why does this occur? (Not required, but can you find a quick fix?)

8. Find and report a good spatial regression equation explaining the minimum January temperatures. Explain, from both ecological and statistical points of view, why you chose the final terms that you did.