Regression Modeling Strategies

Dave Lorenz

February 10, 2015

These examples demonstrate how to use functions with the `smwrBase` package that transform explanatory variables to help model response-explanatory variable relations commonly found in hydrologic data. These example use a single explanatory variable with synthetic data to illustrate how to model the relations. As with most hydrologic relations, the data are assumed to be log-log related.

These examples assume that the user is familiar with general knowledge about linear regression and how to build models in R. The examples simply illustrate some functions in the `smwrBase` package without any recommendation of application.

```
> # Load the smwrBase package
> library(smwrBase)
> # Construct the synthetic data
> DF <- structure(list(Date = structure(c(14885, 14922, 14960,
+     14998, 15036, 15074, 15112, 15150, 15188, 15226, 15263,
+     15301, 15339, 15377, 15415, 15453, 15491, 15529, 15567,
+     15605), class = "Date"),
+   Flow = c(409, 509, 221, 2180, 4010, 2380, 6650, 3960, 3860,
+     558, 776, 1130, 1390, 1310, 2280, 2820, 1900, 1070, 483, 192),
+   Y1 = c(0.55, 0.65, 0.55, 1.18, 1.56, 1.17, 1.97, 1.38, 1.64,
+     0.74, 0.7, 0.88, 0.85, 0.96, 1.03, 1.26, 1.09, 0.8, 0.71, 0.52),
+   Y2 = c(0.52, 0.5, 0.49, 2.15, 2.5, 1.59, 2.34, 1.64, 1.9, 0.4,
+     0.77, 0.88, 1.81, 1.11, 1.69, 1.52, 1.28, 0.74, 0.55, 0.45),
+   Y3 = c(0.56, 0.65, 2.4, 2.54, 3.34, 0.75, 0.78, 0.35, 0.5, 0.31,
+     0.51, 2.23, 2.03, 2.41, 1.77, 1.19, 0.53, 0.35, 0.56, 0.33)),
+   .Names = c("Date", "Flow", "Y1", "Y2", "Y3"),
+   row.names = c(NA, -20L), class = "data.frame")
> # Print the first few rows
> head(DF)

        Date Flow   Y1   Y2   Y3
1 2010-10-03  409 0.55 0.52 0.56
2 2010-11-09  509 0.65 0.50 0.65
3 2010-12-17  221 0.55 0.49 2.40
4 2011-01-24 2180 1.18 2.15 2.54
5 2011-03-03 4010 1.56 2.50 3.34
6 2011-04-10 2380 1.17 1.59 0.75
```

1 Modeling Nonlinear Log-log Relations

Many hydrologic relations are approximately linear and homoscedastic when both the response and the explanatory variables are log transformed (either common or natural logarithms can be used). However, the relation will occasionally have some nonlinearity that must be resolved as shown in figure 1.

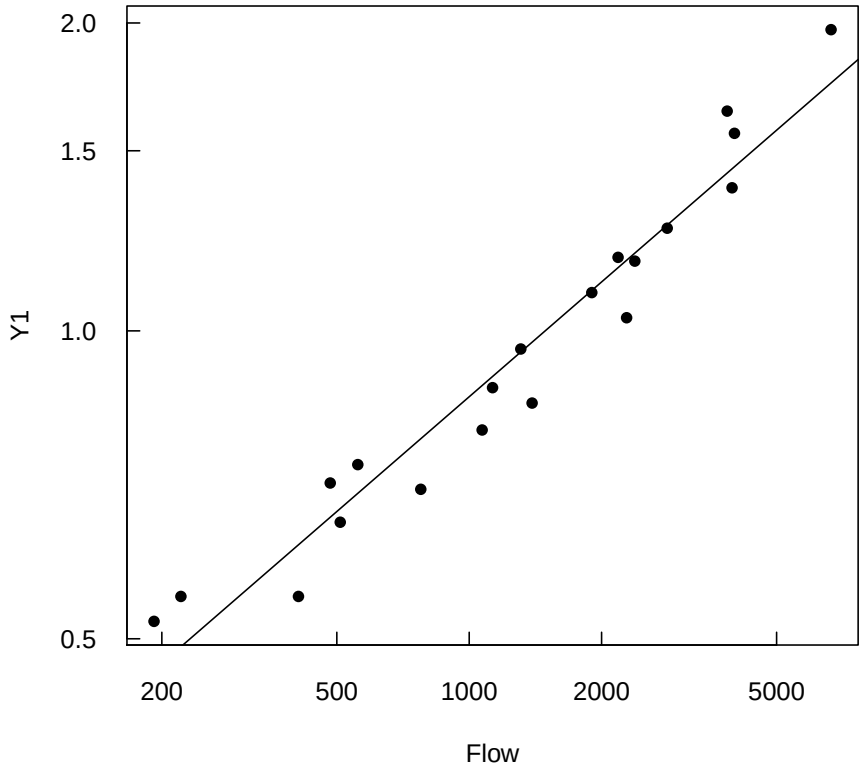

Figure 1. A curved log-log relation.

This example demonstrates three approaches for modeling the relation—adding a second order polynomial term, a ladder-of-powers transformation, and a hyperbolic transformation. No attempt is made to evaluate other linear regression model assumptions, but the residual sum of squares is minimized for the ladder-of-powers and hyperbolic transformations.

A very common and easy to use approach to modeling the nonlinearity shown in figure 1 is to add second or higher-order polynomial terms for the explanatory variable. The `poly` in `stats` is distributed in base R, but it has options only for orthonormal terms, which cannot be directly interpreted, or raw terms, which can produce very large variance inflation factors. The function `quadratic` in `smwrBase` overcomes both of the limitations, but only produces the first- and second-order terms. Its use is straightforward as there is no need to specify separate first- and second-order terms. Use as shown in the code below. The fitted line is shown in figure 2.

```
> # The model with a second-order polynomial term
> lm(log10(Y1) ~ quadratic(log10(Flow)), data=DF)

Call:
lm(formula = log10(Y1) ~ quadratic(log10(Flow)), data = DF)

Coefficients:
                      (Intercept)  quadratic(log10(Flow))(3.03898)1
                         -0.07486                           0.37284
quadratic(log10(Flow))(3.03898)2
                          0.13653
```

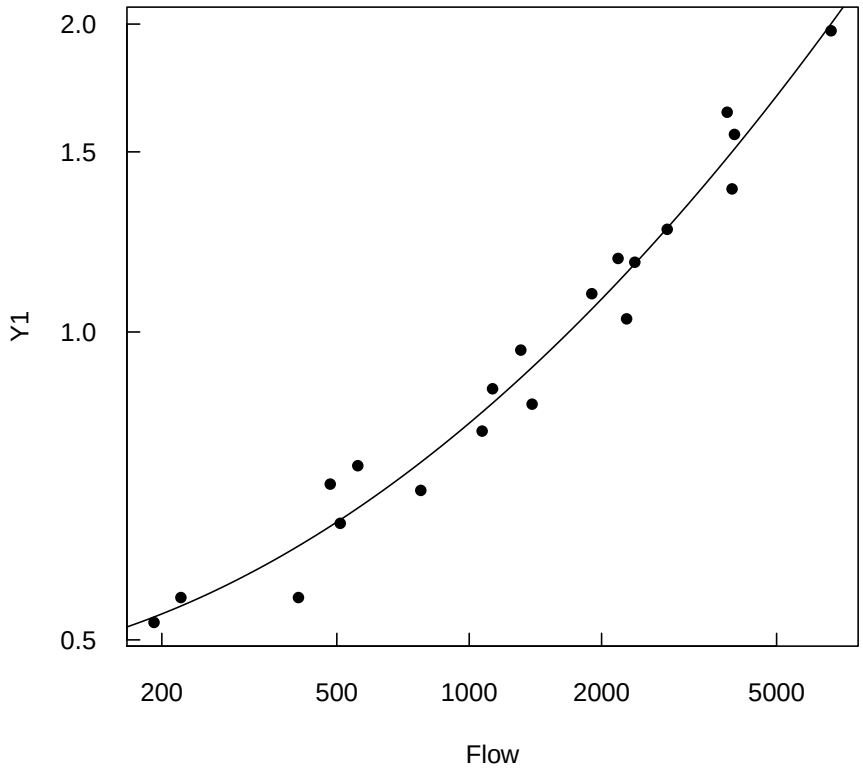

Figure 2. A curved log-log relation with a second-order polynomial fit.

The names of the coefficients can be cumbersome, but they are derived from the name of the term in the formula, quadratic(log10(Flow)), followed by the centering value in parenthesis and the order, 1 for the first-order term and 2 for the second-order term. The centering value is subtracted from the values and produces orthogonal first- and second-order terms by default.

The coefficients can be directly interpreted. For the quadratic function, the slope at the centering value is equal to the the coefficient for the first-order term, 0.37284 for these data. Because the sign of the second-order term is positive, the slope increases for increasing flow and decreases for

decreasing flow. It is sometimes useful to know where the slope is 0. The equation to find where the slope is zero is -2 divided by the coefficient for the second-order term times the coefficient for the first-order term plus the centering value. For this model: -2 / 0.13653 * 0.37284 + 3.03898 = -2.4227 in log units or about 0.004 when converted back to flow units.

Second-order fits can occasionally produce hydrologically undesirable results, for example reversals in the response relation on either side of the zero-slope value, or extraordinarily large response values of extrapolated outside of the calibration data. Helsel and Hirsch (2002) discuss the ladder-of-powers transformation, which can sometimes be used to reduce the likelihood of the reversals. The `boxCox` function in `smwrBase` can be used to construct several candidate models, selecting the "best" fit for the power.

For this example, the minimum sum of squared residuals will be used as the criterion for "best" fit. The `deviance` function in `stats` computes the model deviance, which for a linear regression model is the sum of the squared residuals. In general, the minimum deviance model is not necessarily desirable, but one needs to find a good, broadly applicable model (Helsel and Hirsch, 2002). This example will look at increments of 0.25 between 0 (equivalent to a log transform) and 1 (equivalent to linear) in the script below. The direction and range of the transforms to try can be inferred from Helsel and Hirsch (2002).

```
> # Try models with ladder of powers between 0 and 1
> deviance(lm(log10(Y1) ~ boxCox(Flow, 0.), data=DF))

[1] 0.02845197

> deviance(lm(log10(Y1) ~ boxCox(Flow, 0.25), data=DF))

[1] 0.01636857

> deviance(lm(log10(Y1) ~ boxCox(Flow, 0.50), data=DF))

[1] 0.02017803

> deviance(lm(log10(Y1) ~ boxCox(Flow, 0.75), data=DF))

[1] 0.0378018

> deviance(lm(log10(Y1) ~ boxCox(Flow, 1.), data=DF))

[1] 0.0653619
```

The minimum sum of squared residuals was for the fourth-root transform (0.25). The printed results of the regression model are shown in the script below, and the fitted line is shown in figure 3.

```
> # The model with fourth-root transform
> lm(log10(Y1) ~ boxCox(Flow, 0.25), data=DF)
```

```
Call:
lm(formula = log10(Y1) ~ boxCox(Flow, 0.25), data = DF)

Coefficients:
       (Intercept)  boxCox(Flow, 0.25)
        -0.5956165           0.0001308
```

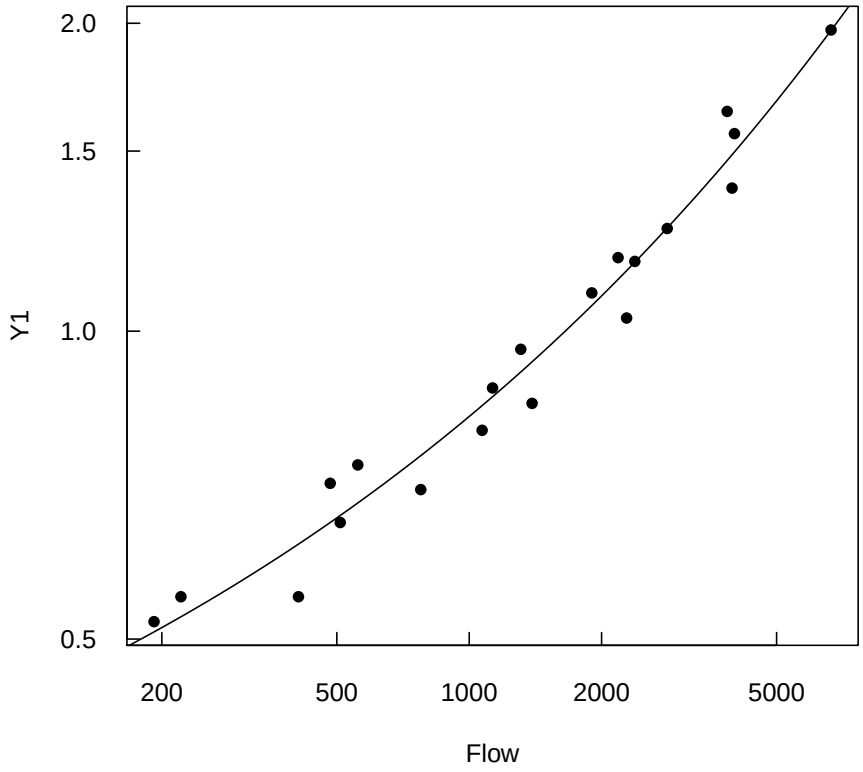

Figure 3. A log-log plot with a curved ladder-of-powers fit.

Another option for fitting data like this is the hyperbolic transform advocated by Johnson and others (1969). The `hyperbolic` function in `smwrBase` can be used for this transform. The hyperbolic transform can be thought of as a way to model mixing of different concentrations at high and low flows. The data represented by Y1 can be thought of as having a lower bound but increasing as flow increases. These are modeled by values for the `factor` argument less than 0. If the data can be thought of as having an upper bound at high flows and dilution at lower flows, then the values for `factor` argument should be greater than 0. In general, there is no reason for the value of `factor` to be greater than 3 or less than -3.

As with the ladder-of-powers example, the "best" model will be selected as shown in the scripts below. It will be done in two sessions, the first to set the limits for the second.

```
> # Try models with hyperbolic factors from 0 to -3
> deviance(lm(log10(Y1) ~ hyperbolic(Flow, factor=0), data=DF))
```

```
[1] 0.01929823

> deviance(lm(log10(Y1) ~ hyperbolic(Flow, factor=-1), data=DF))

[1] 0.0427687

> deviance(lm(log10(Y1) ~ hyperbolic(Flow, factor=-2), data=DF))

[1] 0.06238566

> deviance(lm(log10(Y1) ~ hyperbolic(Flow, factor=-3), data=DF))

[1] 0.06505505

> # Try models with hyperbolic factors from 0 to -.75 (negtive 1 can be excluded)
> deviance(lm(log10(Y1) ~ hyperbolic(Flow, factor=0), data=DF))

[1] 0.01929823

> deviance(lm(log10(Y1) ~ hyperbolic(Flow, factor=-.25), data=DF))

[1] 0.01688832

> deviance(lm(log10(Y1) ~ hyperbolic(Flow, factor=-.50), data=DF))

[1] 0.02267335

> deviance(lm(log10(Y1) ~ hyperbolic(Flow, factor=-.75), data=DF))

[1] 0.0326104
```

The minimum sum of squared residuals was for the factor value of -.25. The printed results of the regression model is shown in the script below and the fitted line in figure 3.

```
> # The model with the hyperbolic transform
> lm(log10(Y1) ~ hyperbolic(Flow, factor=-.25), data=DF)

Call:
lm(formula = log10(Y1) ~ hyperbolic(Flow, factor = -0.25), data = DF)

Coefficients:
                     (Intercept)  hyperbolic(Flow, factor = -0.25)
                       -0.307090                          0.000239
```

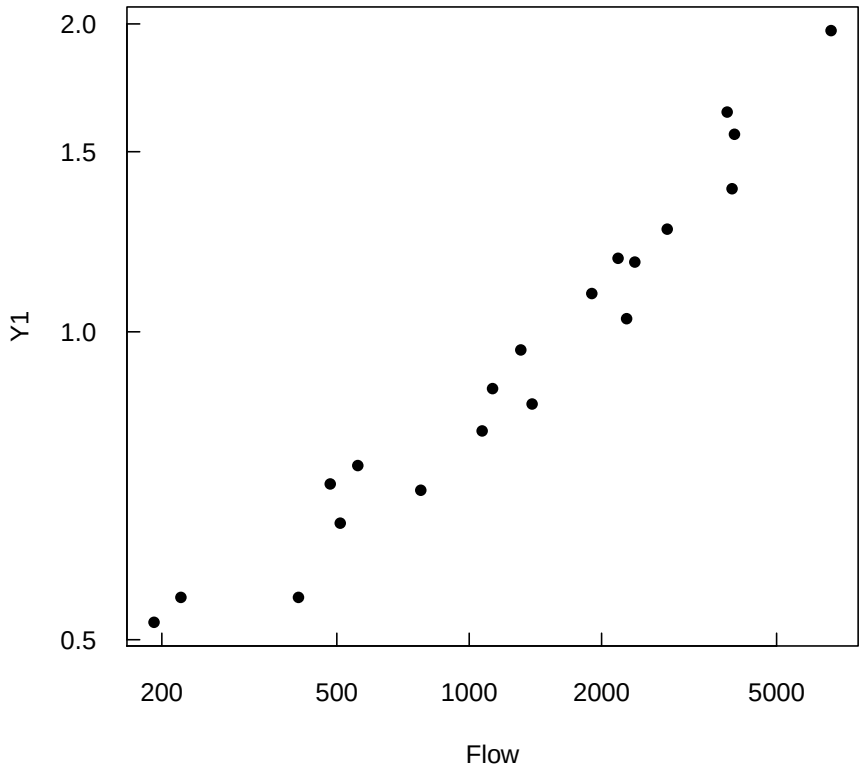

Figure 4. A curved log-log relation with a hyperbolic fit.

All of the models fit the data better than the simple, linear fit. The hyperbolic model overestimates the smallest couple of values, but in general that transform does a fair job of fitting certain kinds of mixture models.

2 Modeling Flow Mixture Relations

As described in the latter part of the previous section, there can be times when the flow-concentration relation represents a mixture model with different source concentrations at high and low flows. The `sCurve` function in `smwrBase` is a very flexible transform for modeling flow mixture models. It is much more flexible than the `hyperbolic` function but requires the fitting of more arguments. The fitted arguments are `location`, which is the transition point in the curve; `scale`, which describes the rate of change at the transition point, and `shape`, which controls how quickly the curve approaches the limits. Both `scale` and `shape` must be greater than 0.

The data in this case are Y2 in the DF dataset. Figure 5 shows the relation between flow and Y2 with a LOWESS smooth to help fit the curve. The sequence of scripts below show one manual approach to fitting the flow mixture model.

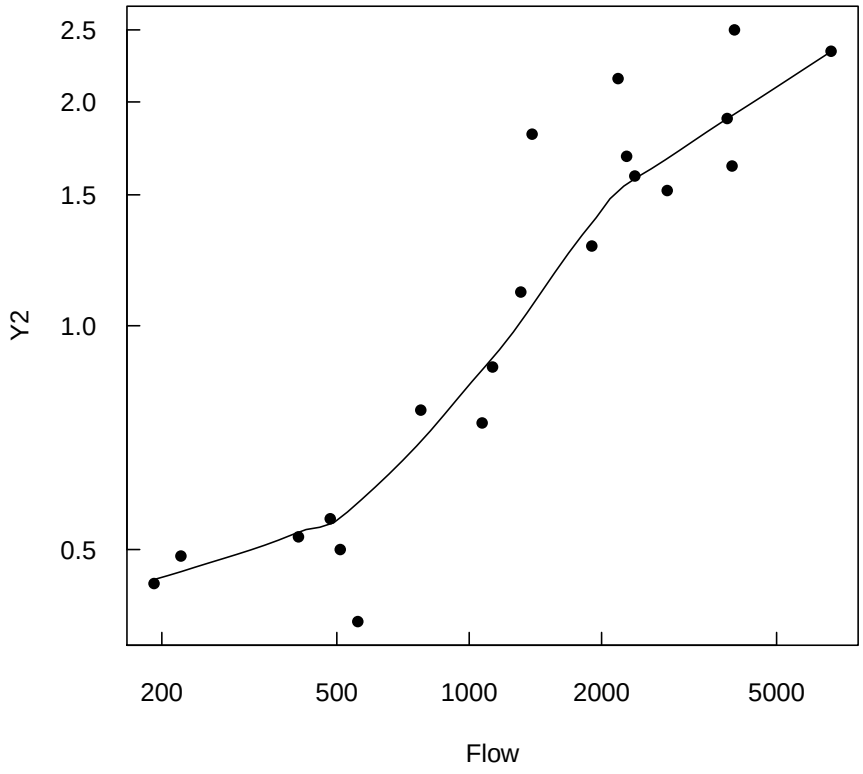

Figure 5. A flow mixture model relation.

Inspection of figure 5 shows that the transition appears to be at a flow value a bit larger than 1,000 and less than 1,500. The first attempt at the model building will be to use 1,200 as the value for location. Because the data are common-log transformed, the location must also be common-log transformed. The fit is constructed in the script below and corresponding graph is shown in figure 6.

```
> # The model with the hyperbolic transform
> lm(log10(Y2) ~ sCurve(log10(Flow), location=log10(1200)), data=DF)

Call:
lm(formula = log10(Y2) ~ sCurve(log10(Flow), location = log10(1200)),
    data = DF)

Coefficients:
                                  (Intercept)
                                      0.00196
sCurve(log10(Flow), location = log10(1200))
                                      0.90982
```

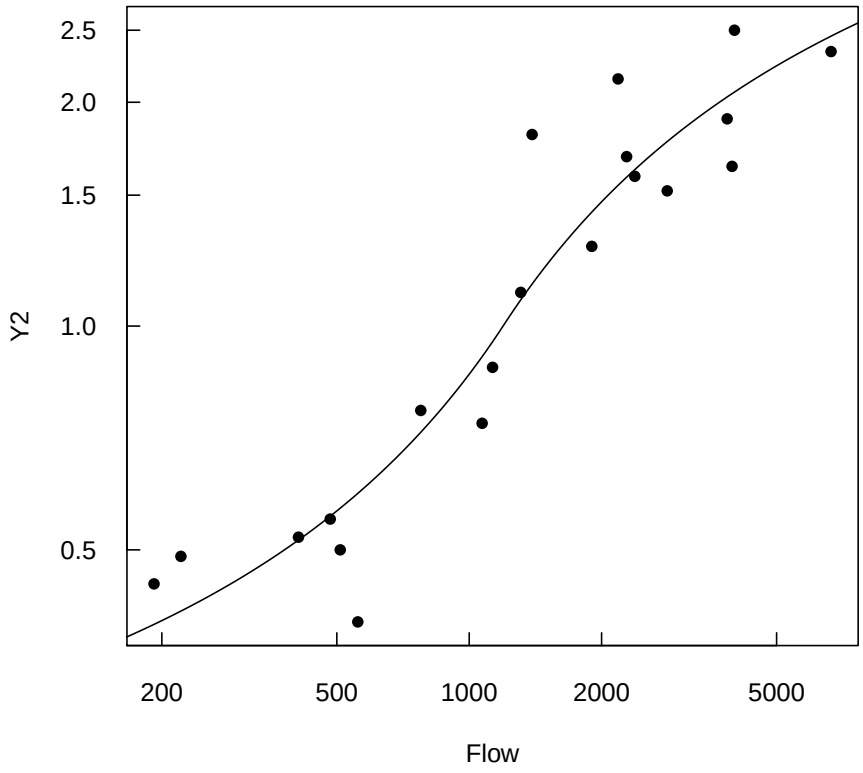

Figure 6. A log-log relation with a curved fit.

The slope indicated by the data at the transition point in figure 6 is much greater than 1, so increase scale and refit.

```
> # The model with the sCurve transform
> lm(log10(Y2) ~ sCurve(log10(Flow), location=log10(1200), scale=5), data=DF)

Call:
```

```
lm(formula = log10(Y2) ~ sCurve(log10(Flow), location = log10(1200),
    scale = 5), data = DF)

Coefficients:
                                            (Intercept)
                                              -0.004898
sCurve(log10(Flow), location = log10(1200), scale = 5)
                                               0.420611
```

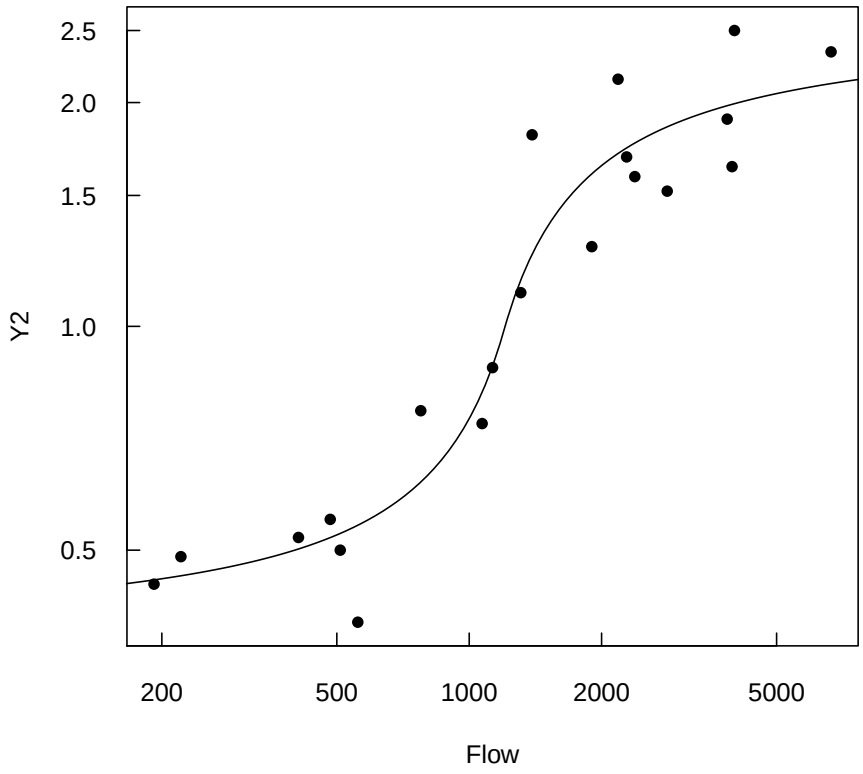

Figure 7. A flow-mixture modeled using the `sCurve` function.

The curve in figure 7 shows a very good fit to the data. There is no reason to pursue further modifications to the arguments to `sCurve`. The `sCurve` function is most sensitive to the `location` and `scale` arguments. The `shape` argument may only need to be adjusted when the scatter of the fit is small.

3 Modeling Seasonal Variation

Helsel and Hirsch (2002) describe multiple regression with periodic functions to model seasonal variability in section 12.4.3. They describe the approach by adding two variables, one for the sine and one for the cosine of annual time. The `fourier` function in `smwrBase` simplifies the addition of those variables by adding both sine and cosine transformations to the model. It also forces both or neither to be considered in best-subset selection procedures. The `k.max` argument can be used to add higher-order seasonal terms; the `k.max` argument is the number of cycles per year.

This example uses column Y3 in the DF dataset with Date as the explanatory variable. The script below shows how to construct the model, and the fit is shown in figure 8.

```
> # The model with fourier transform
> lm(log10(Y3) ~ fourier(Date), data=DF)

Call:
lm(formula = log10(Y3) ~ fourier(Date), data = DF)

Coefficients:
          (Intercept)  fourier(Date)sin(k=1)  fourier(Date)cos(k=1)
             -0.04152                0.23159                0.38081
```

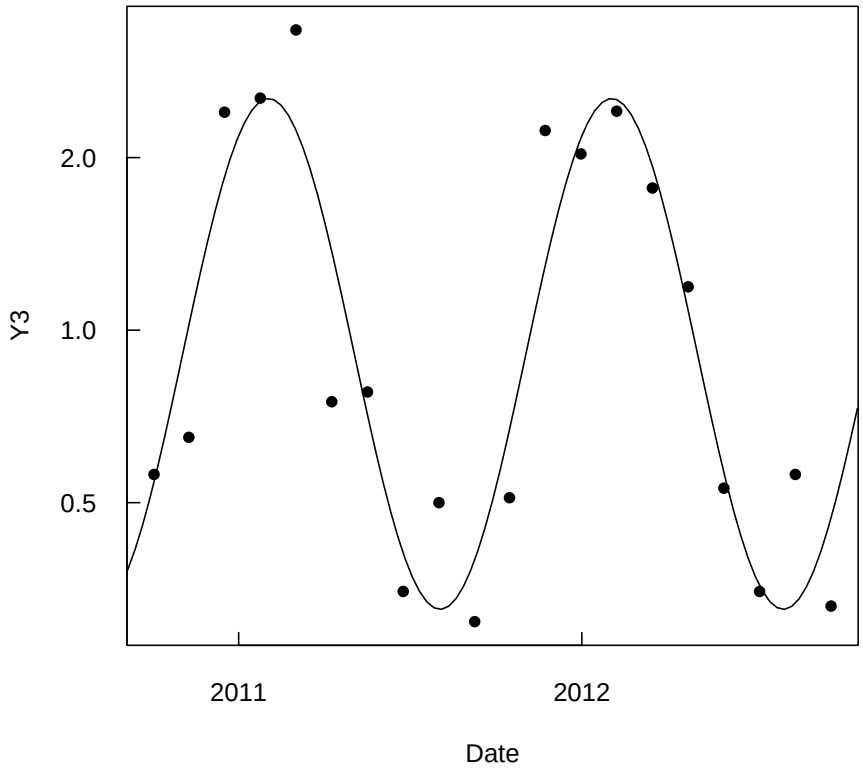

Figure 8. A periodic seasonal pattern fitted using the `fourier` function.

The names of the coefficients can be cumbersome, but they are derived from the name of the term in the formula, fourier(Date), followed by either sin or cos (indicating the transform), and the order of the term in parenthesis.

References

[1] Helsel, D.R., and Hirsch, R.M., 2002, Statistical methods in water resources: U.S. Geological Survey Techniques of Water-Resources Investigations, book 4, chap. A3, 522 p. Available online at: http://pubs.usgs.gov/twri/twri4a3/pdf/twri4a3-new.pdf.

[2] Johnson, N.M., Likens, G.E., Borman, F.H., Fisher, D.W., and Pierce, R.S., 1969, A working model for the variation in stream water chemistry at the Hubbard Brook Experimental Forest, New Hampshire: Water Resources Research, v. 5, no. 6, p. 1353-1363.