Application 2: Analysis of an Uncensored Constituent using a Seasonal Model

Dave Lorenz

July 16, 2013

This example illustrates the construction of a rating-curve load model from a user-defined formula rather than a predefined rating curve model. It also demonstrates how to make load estimates for data outside of the calibration data.

Many constituents are subject to seasonal loading patterns that are driven by external factors such as the length of the growing season, air temperature, and anthropogenic activities. Some of the predefined models implement sine and cosine terms to reflect the seasonality of constituent load. These models are applicable to constituents that change continuously over the seasonal cycle, but are not appropriate for other constituents that are subject to an abrupt seasonal change. Such is the case for pesticides, where springtime application results in a distinct change in the loading pattern for a discrete period of time (fig. 1). When such changes are evident in the data, it often is beneficial to use categorical explanatory variables, often called dummy variables in older literature, (Judge and others, 1988; Helsel and Hirsch, 2002) to handle the abrupt change.

In this example, data from the St. Joseph River near Newville, Ind., is used with a user-defined periodic seasonal model to estimate atrazine loads. Observations of atrazine load show two distinct relations with regard to streamflow; for a given streamflow, atrazine loads are elevated during the 3 month period following pesticide application, May through July, and dramatically lower thereafter. This two-tiered loading response is modeled using the categorical explanatory variables present in per,

$$\log(Load_i) = \alpha_0 + \alpha_1 per_i + \alpha_2 lnQ_i + \alpha_3 per_i lnQ_i + \epsilon_i, \quad (1)$$

where lnQ_i is the centered log of streamflow and per_i is a categorical variable that indicates the seasonal period for observation i. The regression line for estimation during the May though July time period will have an intercept equal to a_0+a_1, and a slope of a_2+a_3. For the other time period (August through April), per will equal 0 and the regression equation corresponds to the simple model used in Application 1.

Part 2 illustrates the diagnostic graphs that can be used to verify the model assumptions or improve the model. A seasonal-wave approach to modeling seasonal is introduced and other graphs to help better understand the model.

```
> # Load the rloadest package and the data
> library(rloadest)
> data(app2.calib)
> head(app2.calib)
```

```
       DATES TIMES FLOW Atrazine
1 1996-03-07  1130  570     0.15
2 1996-04-30   845  945     0.19
3 1996-05-07   815  627     0.18
4 1996-06-04   830  428     0.55
5 1996-06-19   900 2070    16.00
6 1996-07-19   830  132     1.40
```

```
> # Plot the seasonal pattern of Atrazine
> # setSweave is required for the vignette.
> setSweave("app2_01", 5, 5)
> with(app2.calib, seasonPlot(DATES, Atrazine, yaxis.log=TRUE))
> graphics.off()
```

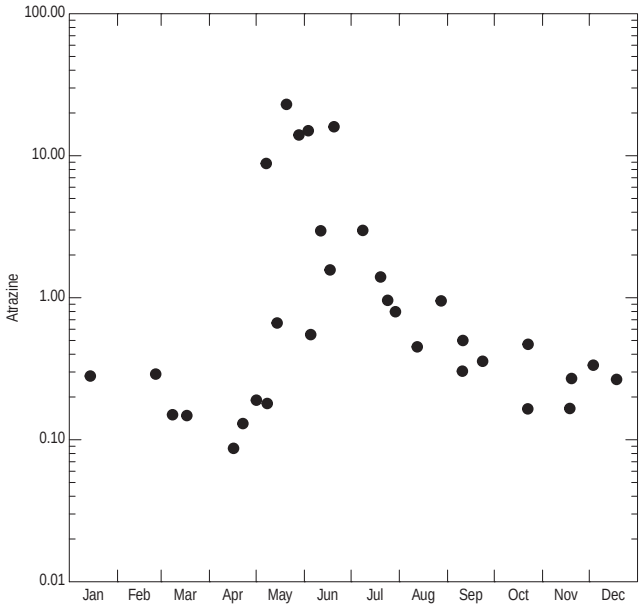

Figure 1. The seasonal pattern of atrazine concentrations.

1 Build the Model

The first step is to create the categorical explanatory variable in the calibration data set. That is easily done using the `seasons` function, which is in the USGSwsBase package. The order of the `breaks` argument in `seasons` must be sequential through the year, but can overlap the end of the year.

```
> # Add Period to the calibration data.
> app2.calib <- transform(app2.calib, Period=seasons(DATES,
+                         breaks=c("Apr", "Jul")))
```

The `loadReg` function is used to build the rating-curve model for constituent load estimation. The basic form of the call to `loadReg` is similar to the call to `lm` in that it requires a formula and data source. The response variable in the formula is the constituent concentration, which is converted to load per day (flux) based on the units of concentration and the units of flow. The `conc.units`, `flow.units`, and `load.units` arguments to `loadReg` define the conversion. For these data, the concentration units (`conc.units`) are "ug/L", the flow units are "cfs" (the default), and the load units for the model are "pounds." Two additional pieces of information are required for `loadReg`—the names of the flow column and the dates column. A final option, the station identifier, can also be specified.

User defined models can be constructed using using the usual rules for constructing regression models in `R`. For this example, we'll take advantage of the * operator to add both the period and log centered flow terms and the interaction term. Details of the differences between the printed output from LOADEST and rloadest were described in application 1, so the short form is printed in this example.

```
> # Create and print the load model.
> app2.lr <- loadReg(Atrazine ~ Period*center(log(FLOW)), data = app2.calib,
+                    flow = "FLOW", dates = "DATES", conc.units="ug/L",
+                    load.units="pounds",
+                    station="St.Joseph River near Newville, Ind.")
> # Warnings are not printed in the vignette
> warnings()

NULL

> app2.lr

*** Load Estimation ***

Constituent:Atrazine
```

```
           Number of Observations: 32
Number of Uncensored Observations: 32
           Center of Decimal Time: 1997.034
                  Center of ln(Q): 6.1625
                Period of record: 1996-03-07 to 1997-11-18

Selected Load Model:
--------------------

Atrazine ~ Period * center(log(FLOW))

Model coefficients:
                                         Estimate Std. Error z-score p-value
(Intercept)                               -0.4097     0.2154  -1.902  0.0486
PeriodSeason Ending Jul                    2.4074     0.3268   7.366  0.0000
center(log(FLOW))                          0.8366     0.1959   4.271  0.0001
PeriodSeason Ending Jul:center(log(FLOW))  1.2005     0.3392   3.540  0.0006

AMLE Regression Statistics
Residual variance: 0.8355
R-squared: 80.87 percent
G-squared: 52.93 on 3 degrees of freedom
P-value: <0.0001
Prob. Plot Corr. Coeff. (PPCC):
  r = 0.9378
  p-value = 0.003
Serial Correlation of Residuals: 0.2667

Variance Inflation Factors:
                  PeriodSeason Ending Jul
                                 1.006729
                       center(log(FLOW))
                                 1.503928
PeriodSeason Ending Jul:center(log(FLOW))
                                 1.510574

Comparison of Observed and Estimated Loads
------------------------------------------
      Summary Stats: Loads in pounds/d
------------------------------------------
     Min  25%  50%  75%  90%   95% Max
Est 0.18 0.70 1.68 5.65 45.2  79.2 198
Obs 0.16 0.42 1.14 2.99 89.1 120.0 179

Bias Diagnostics
----------------
```

```
 Bp: -14.06 percent
PLR: 0.8594
  E: 0.8641
```

2 Estimate Loads

Unlike LOADEST, rloadest requires to the user to build the rating-curve model before estimating loads and will only estimate loads for one type of aggregation at a time, for example total load or loads by season. Before we can estimate loads, we need to get the data and add the same seasonal definition that was used to build the load model.

```
> # Load the estimation data and add Period
> data(app2.est)
> app2.est <- transform(app2.est, Period=seasons(DATES,
+                       breaks=c("Apr", "Jul")))
```

The `predLoad` function is used to estimate loads. It estimates loads in units per day, which is referred to as flux in rloadest. The arguments for `predLoad` are `fit`, the model output from `loadReg`; `newdata`, the estimation dataset; `load.units`, the load units for the estimates, which are taken from the model output if not specified; `by`, a character string indicating how to aggregate the load estimates; `sd`, how to compute the standard error of the load; `allow.incomplete`, a logical value that indicates whether or not to allow incomplete periods to be estimated; and `print`, indicating whether to print a summary.

Unlike the `predict` function in base `R`, `newdata` is required. The columns in `newdata` must match the column names in the calibration dataset. For predefined models, the column names for dates and flow must match.

The `by` argument must be "unit," "day,", "month," "water year," "calendar year," "total," or the name of a grouping column in `newdata`. The "unit" option is not available in version 0.1.

The argument `sd` must be "exact" in version 0.1. Any other value will not give correct values. The argument `allow.incomplete` is not fully implemented in version 0.1.

The total load estimate for Application 2 that matches that in LOADEST can be made using call in the `R` code below.

```
> predLoad(app2.lr, newdata = app2.est, by="total",
+          print=TRUE)

------------------------------------------------------------------------
Constituent Output File Part IIa: Estimation (test for extrapolation)
 Load Estimates for 1996-03-01 to 1998-02-28
------------------------------------------------------------------------

Streamflow Summary Statistics
-----------------------------------------?
```

```
WARNING: The maximum estimation data set steamflow exceeds the maximum
calibration data set streamflow. Load estimates require extrapolation.


------------------------------------------------------------
Constituent Output File Part IIb: Estimation (Load Estimates)
 Load Estimates for 1996-03-01 to 1998-02-28
------------------------------------------------------------

Flux Estimates, in pounds/d, using AMLE
---------------------------------------

  Period Ndays     Flux  Std.Err      SEP      L95      U95
1  total   730 21.63485 19.86159 21.05066 3.131524 76.77985
```

To create the matching seasonal loads in LOADEST, a column that defines the seasons must be added to the estimation data set. That can also be accomplished using the `seasons` function as shown below. The order of the seasons will be different from LOADEST, but the season names will make sense.

```
> app2.est <- transform(app2.est, Season=seasons(DATES,
+                       breaks=c("Jan", "Apr", "Jul", "Oct")))
```

The seasonal load estimates for Application 2 that matches that in LOADEST can be made using call in the `R` code below. In this case, we'll save the output dataset and print it.

```
> app2.seas <- predLoad(app2.lr, newdata = app2.est, by="Season")
> app2.seas

             Period Ndays       Flux    Std.Err        SEP        L95
1 Season Ending Jan   184  1.5055248  0.4645424  0.4892784  0.7694202
2 Season Ending Apr   178  1.8121571  0.6163888  0.6437245  0.8688689
3 Season Ending Jul   184 82.0341969 78.7623286 83.4815220 11.0154468
4 Season Ending Oct   184  0.5411122  0.1430111  0.1572237  0.2974099
          U95
1   2.6644482
2   3.3560436
3 300.1206683
4   0.9078635
```

3 Diagnostic Plots

Figure 2 shows the AMLE 1:1 line as a dashed line and the solid line is a LOWESS smooth curve. The LOWESS curve indicates departure from regression line for larger fitted values. Figure 2 is related for figure 11 in Runkel and others (2004), but collapses the two regression lines into a single line. The largest 5 values show exactly the same pattern between the two figures,being above the regression line, which helps to understand the -14.06 Bp statistic.

```
> # setSweave is required for the vignette.
> setSweave("app2_02", 5, 5)
> plot(app2.lr, which=1, set.up=FALSE)
> graphics.off()
```

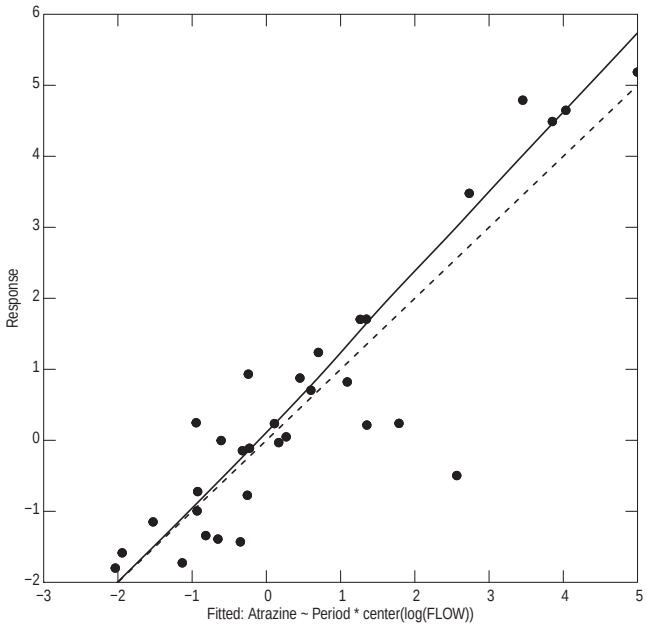

Figure 2. The rating-curve regression model.

Figure 3 is a scale-location (S-L) graph that is a useful graph for assessing heteroscedasticity of the residuals. The horizontal dashed line is the expected value of the square root of the absolute value of the residuals and the solid line is the LOWESS smooth. In this case, only 1 of the largest seven residuals is below the expected value line and most of the other smaller residuals are below the line, which suggests in increasing variance as the estimated load increases.

```
> # setSweave is required for the vignette.
> setSweave("app2_03", 5, 5)
> plot(app2.lr, which=3, set.up=FALSE)
> graphics.off()
```

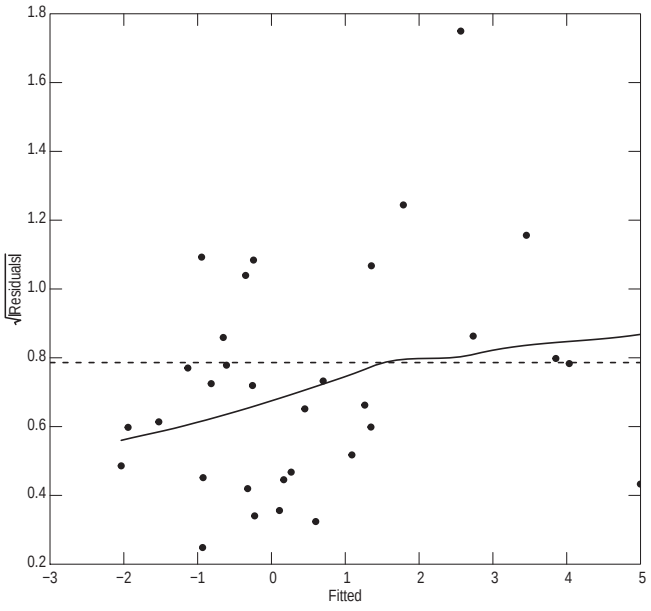

Figure 3. The scale-location graph for the regression model.

The correlogram in figure 4 is a adaptation of the correlogram from time-series analysis, which deals with regular samples. The horizontal dashed line is the zero value and the solid line is a kernel smooth rather than a LOWESS line. The kernel smooth gives a better fit in this case. The solid line should be very close to the horizontal line. In this case, there is a suggestion of a seasonal lack of fit. Note that because the time frame of the calibration period is only 20 months long, the smoothed line is not very reliable greater than about 10 months (about 0.8 units on the X-axis).

```
> # setSweave is required for the vignette.
> setSweave("app2_04", 5, 5)
> plot(app2.lr, which=4, set.up=FALSE)
> graphics.off()
```

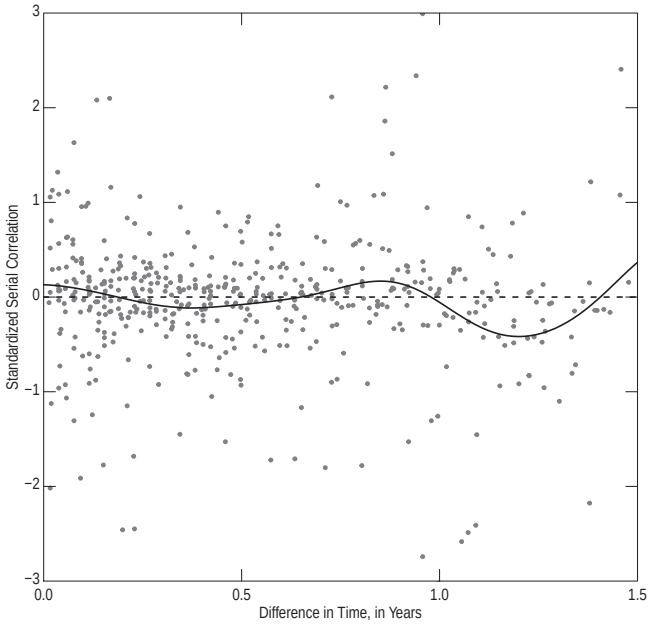

Figure 4. The correlogram from the regression model.

Figure 5 shows the q-normal plot of the residuals. The visual appearance of figure 5 confirms the results of the PPCC test in the printed output—the residuals are not normally distributed but are left-skewed.

```
> # setSweave is required for the vignette.
> setSweave("app2_05", 5, 5)
> plot(app2.lr, which=5, set.up=FALSE)
> graphics.off()
```

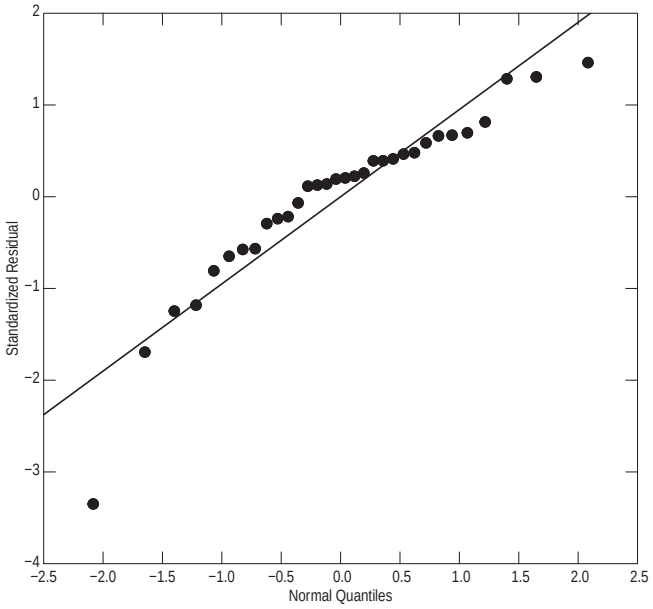

Figure 5. The Q-normal plot of the residuals.

4 Part 2, Building a Seasonal-wave Load Model

All of the diagnostic plots in the previous section indicated a cause for concern about the validity of the periodic regression model. Vecchia and others (2008) describe a seasonal-wave function that often works well for pesticide models.

The USGSwsStats package contains the tools necessary to construct a seasonal-wave model. Building a good regression model is a multi-step process, required identifying the timing of the peak concentration and the other parameters of the seasonal-wave model.

The first step in constructing the seasonal-wave model is the identify the peak and potential values for the other parameters of the model. That involves building a regression model without any seasonal terms, and using the `seasonalPeak` function on the residuals to construct a first guess on those parameters. In this case, because there are no censored values, we can use `lm` instead of `censReg`. Note that it does not matter whether we use load or concentration because the residuals are the same.

```
> # Create the limited regression model.
> app2.lm <- lm(log(Atrazine) ~ center(log(FLOW)), data = app2.calib)
> app2.sp <- seasonalPeak(dectime(app2.calib$DATES), residuals(app2.lm))
> app2.sp

Unconfirmed seasonal peak:
Default value: 0.421
Alternate values: 0.186 0.425
```

The next step in constructing the seasonal-wave model is to confirm the peak. This step requires the `confirm` function, which is interactive and cannot be demonstrated in a vignette. In this case, we can accept the default selection and estimated parameters. The user should step through the interactive process.

```
> # Show the plot for this example
> setSweave("app2_06", 5, 5)
> confirm(app2.sp, plot.only=TRUE)
> graphics.off()
> # Confirm the seasonalPeak analysis for a single peak.
> app2.sp <- confirm(app2.sp, all=TRUE)
> app2.sp

Confirmed seasonal peak:
Number of peaks: 1
Time of peak: 0.421
```

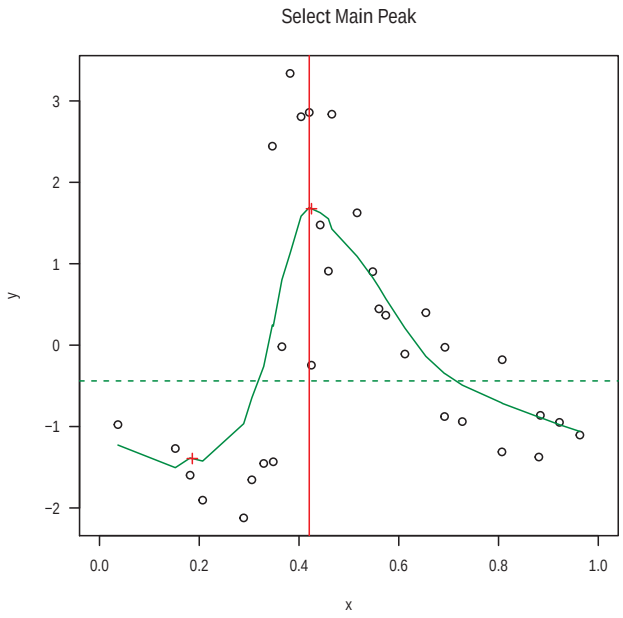

Figure 6. The seasonal peak graph.

The `selBestWave` function can be used to select the "best" parameters for the seasonal-wave model. It requires a column in decimal time format. The following code adds the column `Dectime` and executes `selBestWave`. The results from `selBestWave` are simply printed, but could be saved. Even though the timing of the peak is pretty clear from the graph, we'll take advantage of the exhaustive search to find the "best" peak too.

```
> # Add Dectime.
> app2.calib <- transform(app2.calib, Dectime=dectime(DATES))
> # Find the best model
> selBestWave(log(Atrazine) ~ center(log(FLOW)), data = app2.calib,
+             "Dectime", app2.sp, exhaustive=TRUE)

    Cmax Loading Hlife     Test
1 0.4505       2     3 93.64057
2 0.4505       2     4 93.72557
3 0.4385       2     4 93.82006
4 0.4625       2     3 94.08585
5 0.4385       2     3 95.21835
6 0.4145       1     3 95.28731
```

```
7 0.4625       2     4 95.41145
8 0.4265       2     4 95.43476
```

The "best" model has the timing of the peak at about .45 instead of 0.419 (a bit later in the year), a pesticide loading period of 2 months and a decay rate indicated by a half-life of 3 months (the second slowest decay rate among the default choices). We are now ready to build and evaluate the seasonal-wave load model.

```
> # Create and print the seasonal-wave load model.
> app2.lrsw <- loadReg(Atrazine ~ center(log(FLOW)) +
+                        seasonalWave(Dectime, 0.45, 2, 3),
+                      data = app2.calib, flow = "FLOW",
+                      dates = "DATES", conc.units="ug/L",
+                      load.units="pounds",
+                      station="St.Joseph River near Newville, Ind.")
> app2.lrsw

*** Load Estimation ***

Constituent:Atrazine

           Number of Observations: 32
Number of Uncensored Observations: 32
           Center of Decimal Time: 1997.034
                  Center of ln(Q): 6.1625
                Period of record: 1996-03-07 to 1997-11-18

Selected Load Model:
--------------------

Atrazine ~ center(log(FLOW)) + seasonalWave(Dectime, 0.45, 2,
    3)

Model coefficients:
                                  Estimate Std. Error z-score p-value
(Intercept)                         0.8969     0.1764   5.084       0
center(log(FLOW))                   1.3854     0.1720   8.055       0
seasonalWave(Dectime, 0.45, 2, 3)   4.0836     0.5653   7.224       0

AMLE Regression Statistics
Residual variance: 0.9428
R-squared: 77.65 percent
G-squared: 47.95 on 2 degrees of freedom
P-value: <0.0001
Prob. Plot Corr. Coeff. (PPCC):
```

```
  r = 0.9806
  p-value = 0.2585
Serial Correlation of Residuals: 0.2514

Variance Inflation Factors:
               center(log(FLOW)) seasonalWave(Dectime, 0.45, 2, 3)
                         1.02745                           1.02745

Comparison of Observed and Estimated Loads
------------------------------------------
      Summary Stats: Loads in pounds/d
------------------------------------------
     Min  25%  50%  75%  90%   95% Max
Est 0.16 0.67 2.41 8.83 20.4  65.5 161
Obs 0.16 0.42 1.14 2.99 89.1 120.0 179

Bias Diagnostics
----------------
 Bp: -29.86 percent
PLR: 0.7014
  E: 0.7309
```

The Bias Diagnostics indicate a much poorer fit for this model than the periodic seasonal model originally fit. The seasonal-wave model should be rejected. The periodic seasonal model could be used, but the load estimates are poor. A better understanding of why the periodic seasonal model is better than the seasonal-wave model can be gained by graphing the relation between flow and concentration (fig. 7).

```
> # Use colorPlot to show the relation by season
> setSweave("app2_07", 5, 5)
> AA.pl <- with(app2.calib, colorPlot(FLOW, Atrazine, color=Period,
+         yaxis.log=TRUE, xaxis.log=TRUE))
> addExplanation(AA.pl, "ul", title="Period")
> graphics.off()
```

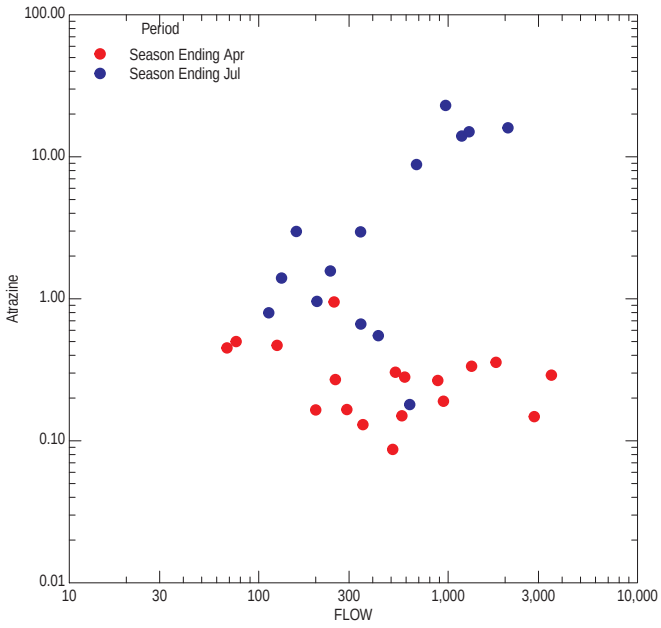

Figure 7. The relation between atrazine concentration and streamflow by period.

Figure 7 is related to figure 11 in Runkel and others (2004), but shows concentration instead of load. Four values stand out as lying outside of their respective seasons—three in the "Season Ending Jul" have concentrations less than about 0.7 and one in the "Season Ending Apr" has a concentrations greater than about 0.7. They are shown below. These atypical observations could contribute to the poor performance of the load models.

```
> subset(app2.calib, Period=="Season Ending Jul" & Atrazine < .7)

        DATES TIMES FLOW Atrazine            Period  Dectime
3  1996-05-07   815  627    0.180 Season Ending Jul 1996.348
4  1996-06-04   830  428    0.550 Season Ending Jul 1996.425
19 1997-05-14  1230  346    0.664 Season Ending Jul 1997.366

> subset(app2.calib, Period=="Season Ending Apr" & Atrazine > .7)

       DATES TIMES FLOW Atrazine            Period  Dectime
7 1996-08-27   730  250     0.95 Season Ending Apr 1996.654
```