Notes on using spline basis functions to model time and flow trends.
Elgin Perry
9 / 14/ 2012

At the last joint session of the CBP tidal and nontidal monitoring groups I suggested the possibility developing a trend assessment using a gam (Generalized Additive Model) approach (Wood, 2006). A gam employs a regression on a cubic spline basis and would give us a tool that is intermediate in structure to the low order polynomial basis in the Estimator model and the local smoothing approach in WRTDS. In what follows, you will see that regression on the spline basis is much more free form in the shape of the trend function than the low order polynomial basis used in Estimator but in this configuration, it does not have the flexibility to track changes in functional form over time as does WRTDS. For example the concentration to flow relationship is assumed constant for the period of record. The gam approach would require more data than required by Estimator and less data that needed for WRTDS.

Joel raised the issue of whether this method could be adapted to handle censored data. Theoretically, this seemed feasible, but I was not certain that it would be easy to find software to do it. I could not find a package available in R that was ready made for this, but I have put together some functions that I think illustrate proof of concept. I use functions from Wood (2006) to create the spline basis and then supply the spline basis to censReg() function to execute a censored data regression. The results compare favorably to regression on uncensored data which is what we would hope.

These models are illustrated using PO4 data from TF1.0, the fall line Patuxent station. This implementation uses the R-package but I think SAS has procedures, like SurvReg, that would permit a similar implementation with a little data step programming to create the spline basis. The program and data are attached if you would like to play with this.

The first thing I did was to fit a gam using the gam() function from Simon Wood's mgcv. This will serve as a basis for comparison to subsequent model fits that work toward using censored data. In the fit, I used an Estimator like model where the time and flow low order polynomial basis terms are replaced by spline basis terms. The seasonal terms are left as simple trig transforms on an annual cycle, but these too could be converted to smoothing terms if there is irregularity in the seasonal cycle.

The function call is

```
gampo4 <- gam(lnPO4~s(numdate)+s(lnFlow)+sindate+cosdate,data=ctp)
```

a the standard summary of results is

```
Family: gaussian
Link function: identity

Formula:
lnPO4 ~ s(numdate) + s(lnFlow) + sindate + cosdate

Parametric coefficients:
            Estimate Std. Error  t value Pr(>|t|)
(Intercept) -3.32486    0.02222 -149.639  < 2e-16 ***
sindate     -0.28114    0.03779   -7.439 5.59e-13 ***
cosdate     -0.23455    0.03407   -6.885 2.07e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
             edf Ref.df      F  p-value
s(numdate) 8.027  8.740 131.12  < 2e-16 ***
s(lnFlow)  3.289  4.146  12.44 7.84e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =  0.769   Deviance explained = 77.6%
GCV score = 0.22301  Scale est. = 0.21579   n = 442
```

These results show that all three independent variables, season, flow, and time trend are important contributors to prediction. A time series plot of the model fit (Figure 1) shows that the time trend is generally decreasing but not monotonically. The relationship between PO4 and flow (Figure 2) is not what I expected. Perhaps the initial decrease is due to dilution in a largely point source dominated river.

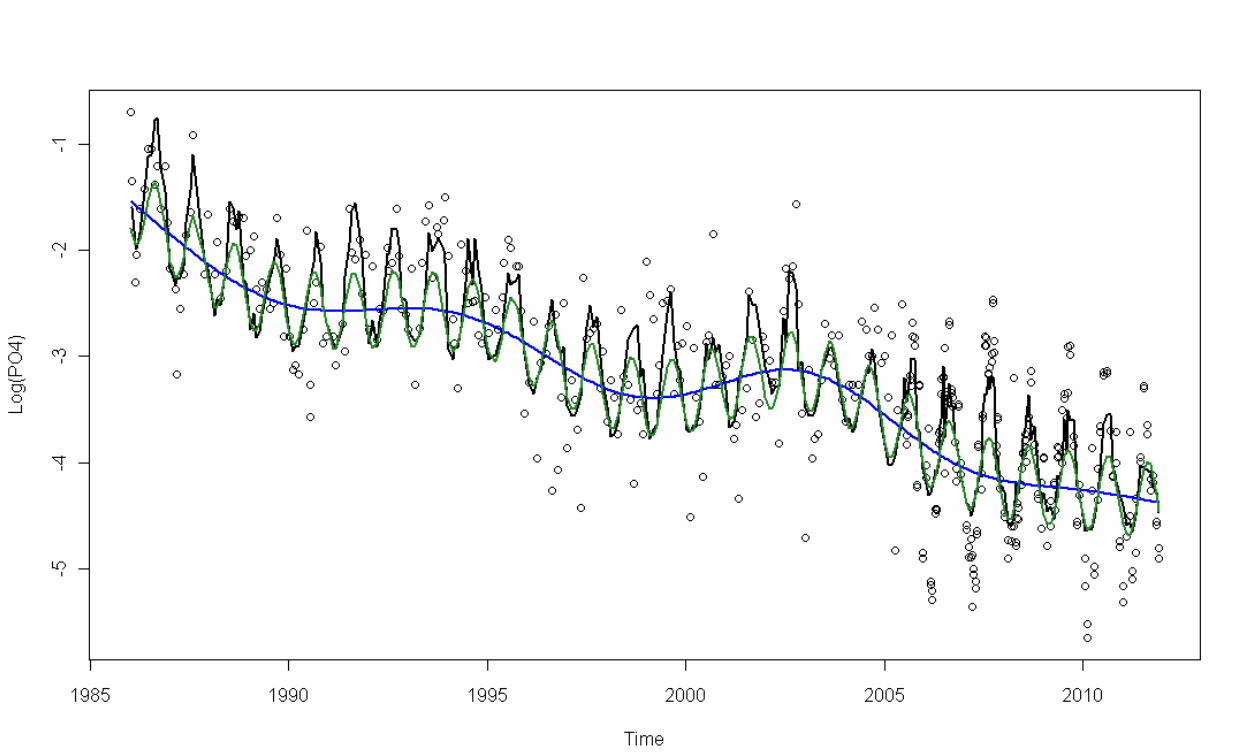

Figure 1. Generalized additive model fit for PO4 at station TF1.0. Circles are observed data, black line is model prediction with all terms, green line show time and season holding flow constant, and the blue line is model prediction of time trend holding season and flow constant.

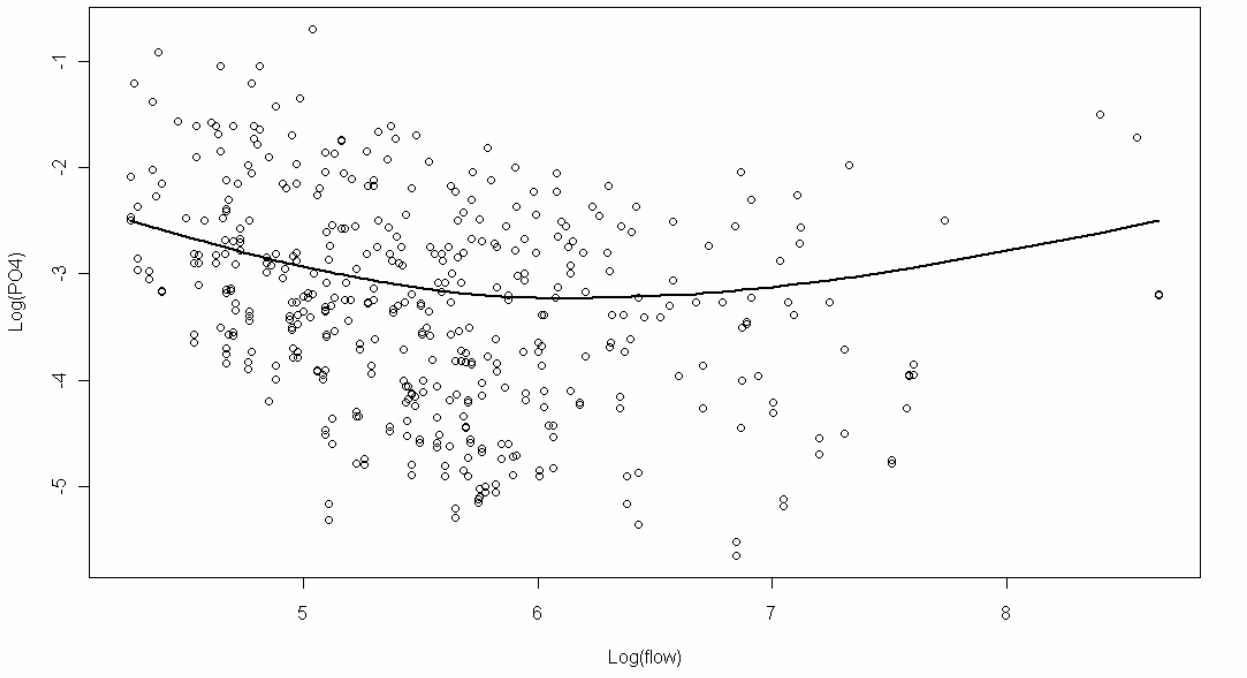

Figure 2. Trend of PO4 versus log(flow) holding time and season constant.

Next I copied two functions, rk() and spl.x(), from Wood (2006) that create the design matrix of spline basis terms. I used these function to create a design matrix to supply to lm(), a simple linear model function, to fit the same data. These functions (see program) and are given on pages 126 and 127 wo Wood. Note that by fitting the model in this way, I am not fully implementing the penalized regression as implemented by gam(). The gam() function includes a penalty for curvature in the prediction curve which helps to prevent it from becoming too wiggly, i.e. over fitting the data. In this step, I am not implementing this penalty part though I think it is feasible to do so. It's the next thing on my list. For now, I am controlling over fit by limiting the number of knots and looking at the graph. The important point is that, the results of this step will not match the results from gam() exactly, but they should be close.

The model call is

```
lm.1 <- lm(ctp$lnPO4~X-1)
```

where X is the design matrix created by rk() and spl.x(). The matrix X has 12 columns, one for the intercept, 6 for time spline basis, 3 for the flow spline basis, and two for seasonal terms.

```
> X[1:10,]
Int  st1    st2      st3      st4      st5     st6     ft1     ft2       ft3    sindate  cosdate
  1 0.0000 0.00174 -0.00298 -0.00468 -0.00298 0.00174 0.1764  0.000363 -0.001509  0.085  0.996
  1 0.0007 0.00174 -0.00296 -0.00467 -0.00297 0.00173 0.1643  0.000159 -0.001622  0.204  0.978
  1 0.0046 0.00174 -0.00289 -0.00459 -0.00294 0.00167 0.6028  0.000783  0.002250  0.746  0.665
  1 0.0059 0.00174 -0.00286 -0.00456 -0.00293 0.00165 0.3322  0.002155  0.000091  0.867  0.498
  1 0.0089 0.00174 -0.00280 -0.00450 -0.00291 0.00161 0.2525  0.001465 -0.000757  0.999  0.022
  1 0.0133 0.00175 -0.00271 -0.00441 -0.00288 0.00154 0.1413 -0.000242 -0.001831  0.775 -0.630
  1 0.0133 0.00175 -0.00271 -0.00441 -0.00288 0.00154 0.1413 -0.000242 -0.001831  0.775 -0.630
  1 0.0170 0.00175 -0.00264 -0.00433 -0.00285 0.00149 0.0870 -0.001258 -0.002296  0.282 -0.959
  1 0.0199 0.00175 -0.00258 -0.00427 -0.00282 0.00145 0.1252 -0.000535 -0.001972 -0.194 -0.980
  1 0.0231 0.00175 -0.00251 -0.00420 -0.00280 0.00140 0.0214 -0.002553 -0.002817 -0.653 -0.757
'
```

The model summary results are:

```
Call:
lm(formula = ctp$lnPO4 ~ X - 1)

Residuals:
     Min       1Q   Median       3Q      Max
-1.86104 -0.26219 -0.00035  0.30033  1.41093

Coefficients:
           Estimate Std. Error t value Pr(>|t|)
XInt     -1.604e+00  1.277e-01 -12.567  < 2e-16 ***
Xst1     -2.686e+00  1.540e-01 -17.439  < 2e-16 ***
Xst2     -2.628e+03  4.696e+02  -5.595 3.92e-08 ***
```

```
Xst3       9.677e+02  2.616e+02   3.700 0.000244 ***
Xst4      -3.526e+03  5.265e+02  -6.697 6.68e-11 ***
Xst5       1.306e+03  2.420e+02   5.397 1.12e-07 ***
Xst6      -2.825e+03  3.937e+02  -7.176 3.16e-12 ***
Xft1      -9.291e-03  2.559e-01  -0.036 0.971053
Xft2      -1.260e+02  2.457e+01  -5.129 4.41e-07 ***
Xft3      -4.174e+01  3.901e+01  -1.070 0.285257
Xsindate -2.576e-01  3.733e-02  -6.901 1.86e-11 ***
Xcosdate -2.239e-01  3.397e-02  -6.590 1.29e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4655 on 430 degrees of freedom
Multiple R-squared: 0.9822,     Adjusted R-squared: 0.9817
F-statistic:  1979 on 12 and 430 DF,  p-value: < 2.2e-16
```

Again, each of the model components appears important. A comparison of the model predictions from gam() and lm() (Figure 3) show that they are quite close. This establishes that using rk() and spl.x() to create spline basis functions actually works.

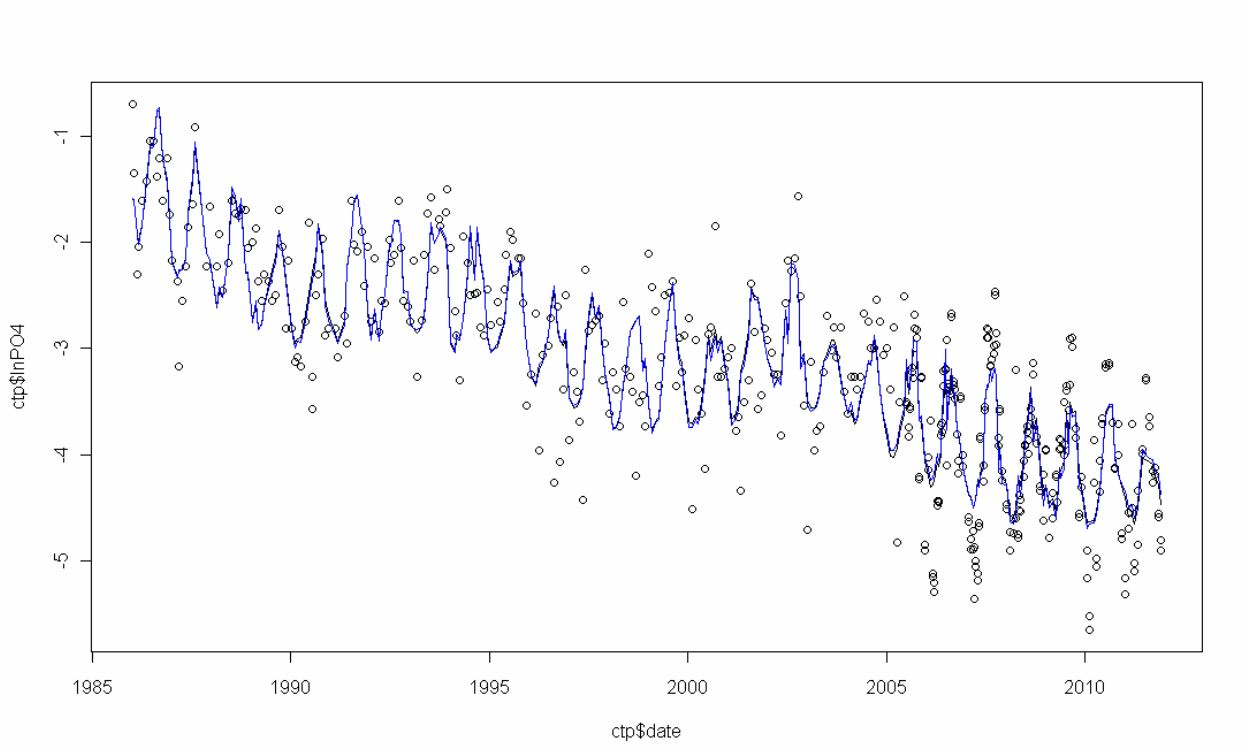

Figure 3. Comparison of fit by gam() (black) to fit by lm() (blue) using spline bases.

The next step is to use the design matrix created above with a maximum likelihood routine that will handle censored data. The r-package offers censReg() as one option for analyzing data with censoring. I am not sure what the detection limit for PO4 is, but for a test, I just censored the existing data at 0.01 using these statements

```
ctp$PO4c <- ifelse(ctp$PO4<0.01,0.01,ctp$PO4) # censor data at 0.01
ctp$lnPO4c <- log(ctp$PO4c)
```

This results in 47 out of 442 observations being censored. The model call to censReg() is:

```
lm.censored <- censReg(ctp$lnPO4c~X-1, left=log(0.01),data=ctp)
```

and the summary of results are:

```
Call:
censReg(formula = ctp$lnPO4c ~ X - 1, left = log(0.01), data = ctp)

Observations:
         Total  Left-censored     Uncensored Right-censored
           442             47            395              0

Coefficients:
           Estimate Std. error t value  Pr(> t)
XInt     -1.585e+00  1.254e-01 -12.637  < 2e-16 ***
Xst1     -2.663e+00  1.527e-01 -17.435  < 2e-16 ***
Xst2     -2.657e+03  4.598e+02  -5.780 7.48e-09 ***
Xst3      9.738e+02  2.557e+02   3.809  0.00014 ***
Xst4     -3.605e+03  5.167e+02  -6.977 3.02e-12 ***
Xst5      1.359e+03  2.376e+02   5.720 1.07e-08 ***
Xst6     -2.904e+03  3.903e+02  -7.440 1.01e-13 ***
Xft1     -5.502e-02  2.502e-01  -0.220  0.82597
Xft2     -1.354e+02  2.430e+01  -5.573 2.50e-08 ***
Xft3     -2.806e+01  3.842e+01  -0.730  0.46511
Xsindate -2.507e-01  3.709e-02  -6.758 1.40e-11 ***
Xcosdate -2.213e-01  3.379e-02  -6.550 5.74e-11 ***
logSigma -7.891e-01  3.624e-02 -21.776  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Newton-Raphson maximisation, 6 iterations
Return code 1: gradient close to zero
Log-likelihood: -290.445 on 13 Df
```

These are not very different from what was obtained using lm(). Comparing the predicted values from censReg() with those from gam() (Figure 4) shows that they are also quite close.

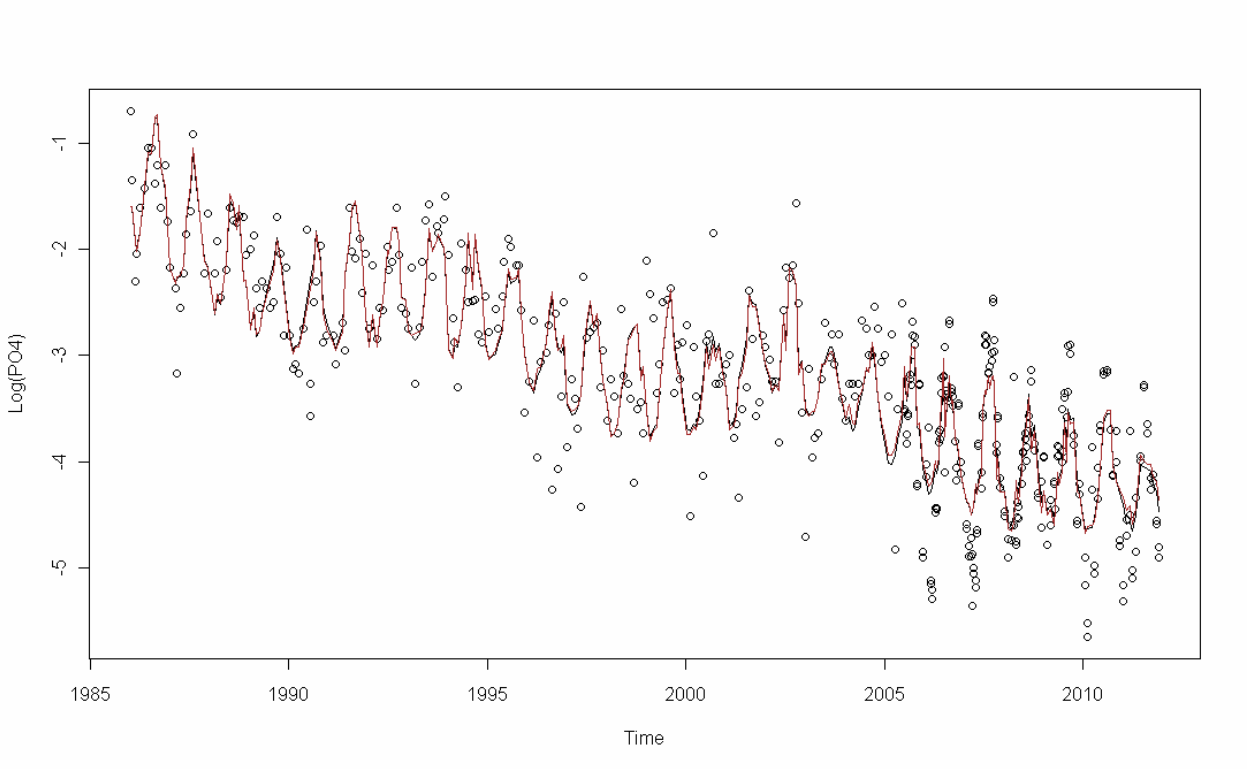

Figure 4. Comparison of predictions from censReg() (brown) to gam() (black).

I think this establishes the feasibility of adding flexibility to an Estimator type model using spline basis functions with censored data. Let me know what you think.

regards, Elgin
eperry@chesapeake.net