Instantaneous Time-Step Model

Dave Lorenz

March 24, 2015

This example illustrates how to set up and use a instantaneous time-step model. These models are typically used when there is additional explanatory variable information such as surrogate unit values, like specific conductance. The intent is often to model both the concentration or flux at any time and the load over a period of time.

This example uses data from the Bad River near Odanah, Wisc., USGS gaging station 04027000. The example will build a model of chloride.

```
> # Load the necessary packages and the data
> library(rloadest)
> library(dataRetrieval)
> # What unit values are available?
> subset(whatNWISdata("04027000", "uv"),
+   select=c("parm_cd", "srsname", "begin_date", "end_date"))

    parm_cd                  srsname begin_date   end_date
12    00010       Temperature, water 2011-03-03 2015-03-24
19    00060 Stream flow, mean. daily 2007-10-01 2015-03-24
25    00065             Height, gage 2014-11-24 2015-03-24
32    00095     Specific conductance 2011-03-06 2015-03-24
39    00300                   Oxygen 2011-03-03 2015-03-24
44    00400                       pH 2011-03-17 2015-03-24
284   63680                Turbidity 2011-03-17 2015-03-24
285   63680                Turbidity 2012-03-24 2015-03-24

> # Get the QW data
> BadQW <- importNWISqw("04027000", "00940",
+   begin.date="2011-04-01", end.date="2014-09-30")
> # Merge data and time and set timezone (2 steps)
> BadQW <- transform(BadQW, dateTime=sample_dt + as.timeDay(sample_tm))
> BadQW <- transform(BadQW, dateTime=setTZ(dateTime, tzone_cd))
> # Now the Unit values data
```

```
> BadUV <- readNWISuv("04027000", c("00060", "00095", "00300", "63680"),
+   startDate="2011-04-01", endDate="2014-09-30", tz="America/Chicago")
> BadUV <- renameNWISColumns(BadUV)
> names(BadUV)

 [1] "agency_cd"        "site_no"          "dateTime"         "tz_cd"
 [5] "Flow_Inst_cd"     "Flow_Inst"        "SpecCond_Inst_cd" "SpecCond_Inst"
 [9] "DO_Inst_cd"       "DO_Inst"          "Turb_Inst_cd"     "Turb_Inst"
[13] "NTU_Turb_Inst_cd" "NTU_Turb_Inst"

> # Strip _Inst off column names
> names(BadUV) <- sub("_Inst", "", names(BadUV))
> # Merge the data
> BadData <- mergeNearest(BadQW, "dateTime", right=BadUV, dates.right="dateTime",
+   max.diff="4 hours")
> # Rename the left-hand dateTime column
> names(BadData)[9] <- "dateTime"
```

1 Build the Instantaneous Time-Step Model

The first step in building the model is to determine which of the surrogates are most appropriate to include in the model. There can be many factors that contribute to deciding which explanatory variables to include in the model. From previous experience the user may decide to include or exclude specific surrogates and flow or seasonal terms. For this example, temperature (parameter code 00010) and pH (parameter code 000400) were excluded as they typically have very little influence on nitrate or nitrate concentration. Other factors include the availability of surrogate values. The output in the code below indicates that NTU_Turb has few observations (more missing values) that Turb, and will not be included in the candidate explanatory variables.

```
> # Print the number of missing values in each column
> sapply(BadData, function(col) sum(is.na(col)))

  site_no.left      sample_dt      sample_tm       tzone_cd      medium_cd
             0              0              0              0              0
 sample_end_dt  sample_end_tm       Chloride       dateTime      agency_cd
             0              0              0              0              0
 site_no.right dateTime.right          tz_cd        Flow_cd           Flow
             0              0              0             23             23
   SpecCond_cd       SpecCond          DO_cd             DO        Turb_cd
            22             22             16             16             27
          Turb    NTU_Turb_cd       NTU_Turb
            27             74             83
```

This example will include the other surrogates and flow and seasonal terms in the candidate model. The code below demonstrates the use of `selBestSubset` to select the initial candidate model.

```
> # Create the and print the candidate model.
> BadChloride.lr <- selBestSubset(Chloride ~ log(Flow) + fourier(dateTime) +
+   log(SpecCond) + log(DO) + log(Turb), data=BadData,
+   flow="Flow", dates="dateTime", time.step="instantaneous",
+   station="Bad River near Odanah", criterion="SPPC")
> print(BadChloride.lr)

*** Load Estimation ***

Station: Bad River near Odanah
Constituent: Chloride
```

```
           Number of Observations: 85
Number of Uncensored Observations: 85
           Center of Decimal Time: 2012.58
                  Center of ln(Q): 6.772
                 Period of record: 2011-04-12 07:59:00 to 2014-07-15 14:15:00

Model Evaluation Criteria Based on AMLE Results
-----------------------------------------------

       Step Df Deviance Resid. Df Resid. Dev  SPPC
1           NA       NA        77     -15.95 19.59
2 - log(DO)  1   0.3468        78     -15.60 15.50
Model # 99 selected

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

Chloride ~ log(Flow) + fourier(dateTime) + log(SpecCond) + log(Turb)

Model coefficients:
                          Estimate Std. Error z-score p-value
(Intercept)                -5.5744    1.45776  -3.824  0.0001
log(Flow)                   1.2973    0.09535  13.606  0.0000
fourier(dateTime)sin(k=1)   0.3669    0.09069   4.045  0.0001
fourier(dateTime)cos(k=1)   0.2045    0.07107   2.877  0.0036
log(SpecCond)               1.3980    0.22357   6.253  0.0000
log(Turb)                  -0.2498    0.05468  -4.569  0.0000

AMLE Regression Statistics
Residual variance: 0.05243
R-squared: 95.56 percent
G-squared: 264.7 on 5 degrees of freedom
P-value: <0.0001
Prob. Plot Corr. Coeff. (PPCC):
  r = 0.9659
  p-value = 5e-04
Serial Correlation of Residuals: 0.2815

Variance Inflation Factors:
                             VIF
log(Flow)                 23.010
fourier(dateTime)sin(k=1)  3.568
fourier(dateTime)cos(k=1)  1.203
log(SpecCond)             11.621
log(Turb)                 10.351
```

```
Comparison of Observed and Estimated Loads
------------------------------------------
      Summary Stats: Loads in kg/d
---------------------------------------------
    Min  25%   50%   75%   90%   95%   Max
Est 920 7700 11500 21400 28400 35000 58000
Obs 920 7260 11500 21000 32800 35200 40400

Bias Diagnostics
----------------
 Bp: 0.08092 percent
PLR: 1.001
  E: 0.8249
```

Only log(DO) was dropped from the model. The printed report indicates some potential problems with the regression—the PPCC test indicates the residuals are not normally distributed and several variance inflation factors are relatively large, greater than 10. But the bias diagnostics show very little bias in the comparison of the estimated to observed values.

A few selected graphs will help understand the issues identified in the printed report and suggest an alternative model. Figure 1 shows the residuals versus fitted graph, which indicates some very large residuals at larger fitted values. It also suggests some heteroscedasticity in the residual pattern.

```
> # Plot the overall fit, choose plot number 2.
> setSweave("graph01", 6, 6)
> plot(BadChloride.lr, which = 2, set.up=FALSE)
> dev.off()

null device
          1
```

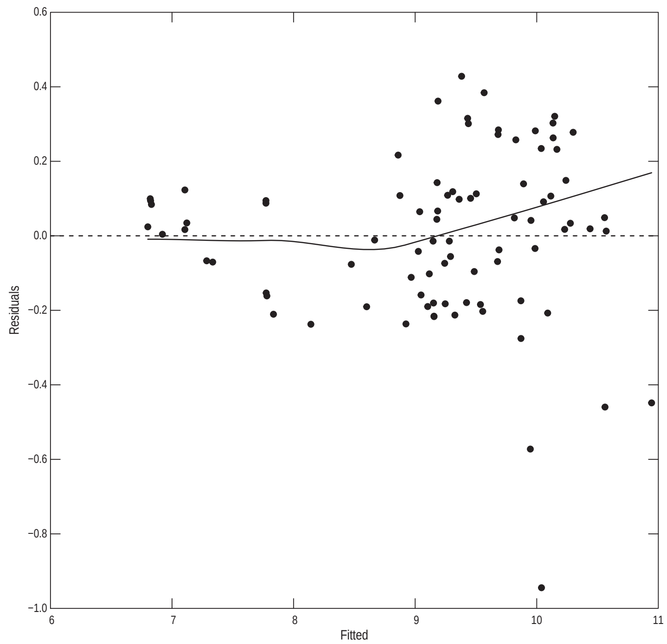

Figure 1. The residuals versus fitted graph.

The S-L plot is not shown. The residual Q-normal graph indicates the reason for the very low p-value indicated by the PPCC test—the large residual values indicated in figure 1 skew the distribution.

```
> # Plot the residual Q-normal graph.
> setSweave("graph02", 6, 6)
> plot(BadChloride.lr, which = 5, set.up=FALSE)
> dev.off()

null device
          1
```

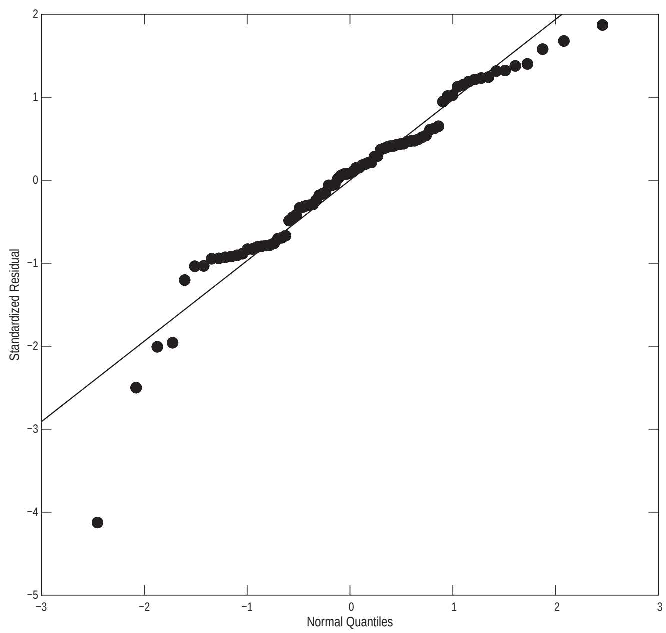

Figure 2. The residual Q-normal graph.

A complete review of the partial residual graphs is not included in this example. Only the partial residual for `log(Turb)` is shown. The graph indicates the lack of fit, especially for the largest values of Turbidity. This suggests that the log transform is not appropriate.

```
> # Plot the residual Q-normal graph.
> setSweave("graph03", 6, 6)
> plot(BadChloride.lr, which = "log(Turb)", set.up=FALSE)
> dev.off()
```

```
null device
          1
```

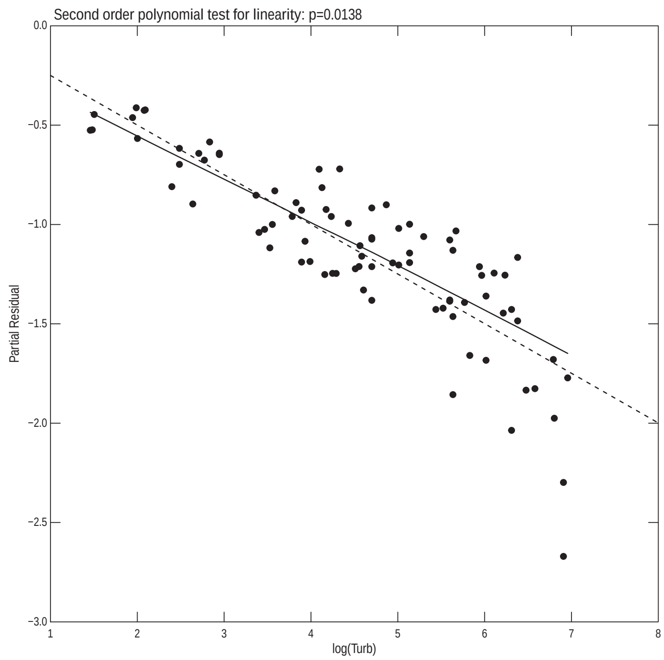

Figure 3. The partial residual for `log(Turb)` graph.

Build the model excluding `log(DO)` that was dropped in the subset selection procedure and changing `log(Turb)` to `Turb`.

```
> # Create the and print the revised model.
> BadChloride.lr <- loadReg(Chloride ~ log(Flow) + fourier(dateTime) +
+   log(SpecCond) + Turb, data=BadData,
+   flow="Flow", dates="dateTime", time.step="instantaneous",
+   station="Bad River near Odanah")
> print(BadChloride.lr, load.only=FALSE)
```

```
*** Load Estimation ***

Station: Bad River near Odanah
Constituent: Chloride

           Number of Observations: 85
Number of Uncensored Observations: 85
           Center of Decimal Time: 2012.58
                  Center of ln(Q): 6.772
```

```
                  Period of record: 2011-04-12 07:59:00 to 2014-07-15 14:15:00

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

Chloride ~ log(Flow) + fourier(dateTime) + log(SpecCond) + Turb

Model coefficients:
                          Estimate Std. Error z-score p-value
(Intercept)               -7.33549  1.0913995  -6.721   0e+00
log(Flow)                  1.27900  0.0550056  23.252   0e+00
fourier(dateTime)sin(k=1)  0.24883  0.0681555   3.651   3e-04
fourier(dateTime)cos(k=1)  0.29203  0.0570742   5.117   0e+00
log(SpecCond)              1.65230  0.1687211   9.793   0e+00
Turb                      -0.00122  0.0001442  -8.464   0e+00

AMLE Regression Statistics
Residual variance: 0.03476
R-squared: 97.05 percent
G-squared: 299.6 on 5 degrees of freedom
P-value: <0.0001
Prob. Plot Corr. Coeff. (PPCC):
  r = 0.9916
  p-value = 0.2894
Serial Correlation of Residuals: 0.166

Variance Inflation Factors:
                             VIF
log(Flow)                 11.549
fourier(dateTime)sin(k=1)  3.039
fourier(dateTime)cos(k=1)  1.170
log(SpecCond)              9.982
Turb                       3.371

Comparison of Observed and Estimated Loads
------------------------------------------
      Summary Stats: Loads in kg/d
---------------------------------------------
    Min  25%   50%   75%   90%   95%   Max
Est 917 7530 11900 19300 29400 36400 60900
Obs 920 7260 11500 21000 32800 35200 40400

Bias Diagnostics
----------------
 Bp: 0.4108 percent
PLR: 1.004
```

```
  E: 0.8596

Selected Concentration Model:
-----------------------------

Chloride ~ log(Flow) + fourier(dateTime) + log(SpecCond) + Turb

Model coefficients:
                          Estimate Std. Error z-score p-value
(Intercept)               -8.23018  1.0913995  -7.541   0e+00
log(Flow)                  0.27900  0.0550056   5.072   0e+00
fourier(dateTime)sin(k=1)  0.24883  0.0681555   3.651   3e-04
fourier(dateTime)cos(k=1)  0.29203  0.0570742   5.117   0e+00
log(SpecCond)              1.65230  0.1687211   9.793   0e+00
Turb                      -0.00122  0.0001442  -8.464   0e+00

AMLE Regression Statistics
Residual variance: 0.03476
R-squared: 76.28 percent
G-squared: 122.3 on 5 degrees of freedom
P-value: <0.0001
Prob. Plot Corr. Coeff. (PPCC):
  r = 0.9916
  p-value = 0.2894
Serial Correlation of Residuals: 0.166


Comparison of Observed and Estimated Concentrations
---------------------------------------------------
      Summary Stats: Concentrations in mg/l
----------------------------------------------
     Min  25%  50%  75%  90%  95%  Max
Est 1.15 2.00 2.46 2.91 3.88 4.10 5.25
Obs 0.76 1.85 2.37 3.08 3.96 4.28 4.96

Bias Diagnostics
----------------
 Bp: 0.3245 percent
PCR: 1.003
  E: 0.8115
```

The report for the revised model indicates less severe problems than from the first candidate model—the p-value for the PPCC test is greater than 0.05, the variance inflation inflation factors are lower although `log(Flow)` is still greater than 10, and the bias diagnostics from the observed and estimated loads and concentrations are still good.

A review of selected diagnostic plots indicates a much better overall fit. Figure 4 shows the residuals versus fitted graph, which indicates a less severe problem of large residuals at larger fitted values. It also suggests some heteroscedasticity in the residual pattern as with the first candidate model.

```
> # Plot the overall fit, choose plot number 2.
> setSweave("graph04", 6, 6)
> plot(BadChloride.lr, which = 2, set.up=FALSE)
> dev.off()

null device
          1
```

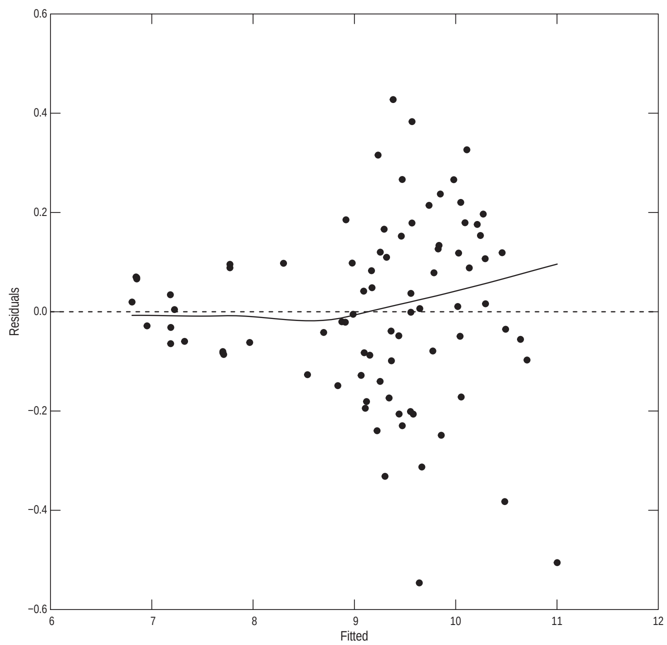

Figure 4. The residuals versus fitted graph for the revised model.

For this model, the S-L plot is shown. It shows an increase in heteroscedasticity as the fitted values increase. That heteroscedasticity can introduce bias into the estimated values as the bias correction factor will be a bit too small for the larger values and too large for the smaller values. The potential bias for this model is expected to be small because the residual variance is small, 0.03476 natural log units, therefore the bias correction is very small, less than 2 percent, and the potential change to the bias correction very small, much less than 1/2 percent.

```
> # Plot the S-L grpah.
> setSweave("graph05", 6, 6)
> plot(BadChloride.lr, which = 3, set.up=FALSE)
> dev.off()
```

```
null device
          1
```

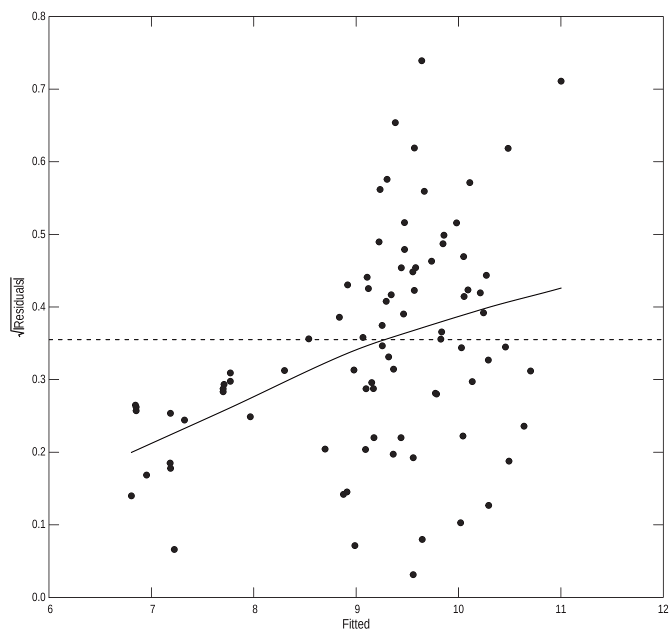

Figure 5. The S-L graph for the revised model.

The residual Q-normal graph shows much better agreement to the normal distribution than the original candidate model—the effect of the lowest residuals is much less.

```
> # Plot the residual Q-normal graph.
> setSweave("graph06", 6, 6)
> plot(BadChloride.lr, which = 5, set.up=FALSE)
> dev.off()
```

```
null device
          1
```

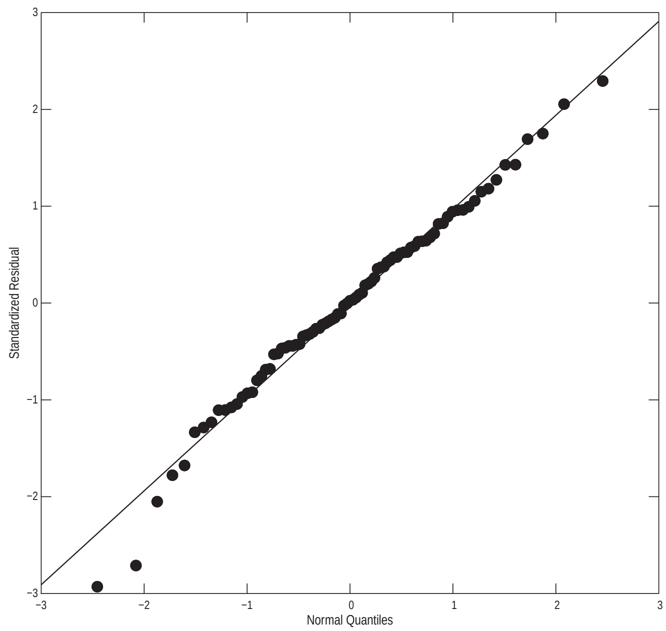

Figure 6. The residual Q-normal graph for the revised model.

A complete review of the partial residual graphs is not included in this example. Only the partial residual for `Turb` is shown to compare to the original model. In this case, the untransformed variable appears to fit reasonably well.

```
> # Plot the residual Q-normal graph.
> setSweave("graph07", 6, 6)
> plot(BadChloride.lr, which = "Turb", set.up=FALSE)
> dev.off()
```

```
null device
          1
```

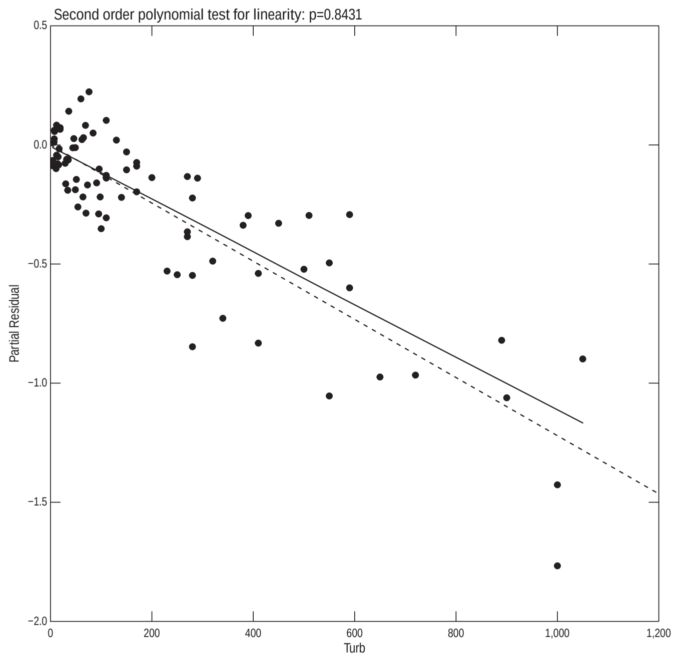

Figure 7. The partial residual for `Turb` graph for the revised model.

2 Instantaneous Concentrations

Estimating the instantaneous concentrations or loads from the model is relatively straight forward. The `predConc` and `predLoad` functions will estimate concentrations or loads, actually fluxes, for any time, given explanatory variables with no missing values. This example will focus one a single day, June 30, 2014.

```
> # Extract one day from the UV data
> Bad063014 <- subset(BadUV, as.Date(as.POSIXlt(dateTime)) == "2014-06-30")
> # Remove the unecessary surrogates from the data set.
> # This reduces the likelihood of missing values in the dataset
> Bad063014 <- Bad063014[, c("dateTime", "Flow", "SpecCond", "Turb")]
> # Simple check
> any(is.na(Bad063014))
```

```
[1] FALSE
```

```
> # Estimate concetrations
> Bad063014.est <- predConc(BadChloride.lr, Bad063014, by="unit")
> # Display the first and last few rows.
> head(Bad063014.est)
```

```
                 Date Flow     Conc   Std.Err       SEP      L95      U95
1 2014-06-30 00:00:00  909 3.315789 0.1483089 0.6410364 2.223455 4.766606
2 2014-06-30 00:15:00  909 3.299546 0.1462430 0.6375876 2.213003 4.742480
3 2014-06-30 00:30:00  909 3.311449 0.1477848 0.6401211 2.220654 4.760175
4 2014-06-30 00:45:00  909 3.311294 0.1477819 0.6400920 2.220548 4.759954
5 2014-06-30 01:00:00  903 3.305057 0.1468059 0.6387257 2.216595 4.750585
6 2014-06-30 01:15:00  903 3.249339 0.1411140 0.6272261 2.180268 4.668656
```

```
> tail(Bad063014.est)
```

```
                  Date Flow     Conc   Std.Err       SEP      L95      U95
91 2014-06-30 22:30:00  770 3.058103 0.1205894 0.5876831 2.055698 4.387281
92 2014-06-30 22:45:00  770 3.046816 0.1194058 0.5853628 2.048326 4.370707
93 2014-06-30 23:00:00  770 3.057821 0.1205899 0.5876313 2.055505 4.386882
94 2014-06-30 23:15:00  764 3.043614 0.1190648 0.5847037 2.046236 4.366004
95 2014-06-30 23:30:00  764 3.050892 0.1198473 0.5862035 2.050985 4.376700
96 2014-06-30 23:45:00  758 2.988006 0.1148736 0.5736139 2.009432 4.285213
```

```
> # The daily mean concentration can also be easily estimated
> predConc(BadChloride.lr, Bad063014, by="day")
```

```
        Date     Flow     Conc   Std.Err       SEP      L95      U95
1 2014-06-30 829.4792 3.158582 0.1302371 0.6083021 2.121373 4.534724

> # Compare to the mean of the unit values:
> with(Bad063014.est, mean(Conc))

[1] 3.158582
```

3 Aggregate Loads

Estimating loads for periods longer than one day requires consistent number of unit values in each day. The consistent number per day is required to be able to keep track of within-day and between day variances. The `resampleUVdata` function can be used to force a consistent number of unit values per day. It is not required for this example, but useful when the unit values are not consistent or when there is a change to or from daylight savings time.

Just as with estimating instantaneous values, missing values are not permitted. Missing values can occur with surrogates due to short-term malfunctions, calibration, or long-term malfunctions. Missing values from short-term malfunctions, generally spikes in the data that are removed during processing, or that occur during calibrations can easily be interpolated using the `fillMissing` function in (smwrBase) and are illustrated in this example. Longer-term missing values are much more difficult to fix. They require the careful balancing of need, developing alternate regression models and possible caveats of the interpretation of loads.

```
> # Extract one month from the UV data, done in two steps
> Bad0714 <- subset(BadUV, as.Date(as.POSIXlt(dateTime)) >= "2014-07-01")
> Bad0714 <- subset(Bad0714, as.Date(as.POSIXlt(dateTime)) <= "2014-07-31")
> # Remove the unecessary surrogates from the data set.
> # This reduces the likelihood of missing values in the dataset
> Bad0714 <- Bad0714[, c("dateTime", "Flow", "SpecCond", "Turb")]
> # Simple check on each column, how many in each column?
> sapply(Bad0714, function(x) sum(is.na(x)))
```

```
dateTime     Flow SpecCond     Turb 
       0        0        3       17 
```

```
> # Fix each column, using the defaults of fillMissing
> Bad0714$SpecCond <- fillMissing(Bad0714$SpecCond)
> Bad0714$Turb <- fillMissing(Bad0714$Turb)
> # Verify filled values
> sapply(Bad0714, function(x) sum(is.na(x)))
```

```
dateTime     Flow SpecCond     Turb 
       0        0        0        0 
```

```
> # Estimate daily loads
> Bad0714.day <- predLoad(BadChloride.lr, Bad0714, by="day")
> # Display the first and last few rows.
> head(Bad0714.day)
```

```
        Date     Flow     Flux  Std.Err       SEP      L95      U95
1 2014-07-01 690.9271 4999.268 188.4703  961.0055 3360.170 7172.870
2 2014-07-02 733.5521 5389.557 209.8556 1041.0709 3615.312 7745.524
3 2014-07-03 702.1146 5062.956 194.9826  975.9416 3399.136 7271.018
4 2014-07-04 519.5417 3620.096 135.9578  696.7035 2432.021 5196.104
5 2014-07-05 399.8750 2895.400 112.0630  556.6967 1945.926 4154.565
6 2014-07-06 336.1875 2518.843 101.0843  484.4965 1692.567 3614.753
```

```
> tail(Bad0714.day)
```

```
         Date     Flow     Flux   Std.Err      SEP      L95      U95
26 2014-07-26 204.7604 1647.717  72.96417 318.4157 1105.095 2368.332
27 2014-07-27 200.3438 1723.719  76.54007 334.0974 1154.653 2480.076
28 2014-07-28 275.9167 2509.353 114.02758 486.0095 1681.434 3609.530
29 2014-07-29 277.8333 2379.386 105.39399 460.0444 1595.477 3420.586
30 2014-07-30 372.5625 3184.768 147.53720 619.2851 2130.508 4587.265
31 2014-07-31 477.9062 3849.708 183.75181 748.8520 2574.951 5545.704
```

```
> # And the month
> Bad0714.mon <- predLoad(BadChloride.lr, Bad0714, by="month")
> Bad0714.mon
```

```
     Period Ndays    Flux  Std.Err      SEP      L95      U95
1 July 2014    31 3060.67 118.2118 115.4483 2840.608 3293.095
```

```
> # Compare to the results using the approximate standard error:
> # For long periods, the processing time to the exact seopt can be very large
> # and may be desireable to use the approximation.
> predLoad(BadChloride.lr, Bad0714, by="month", seopt="app")
```

```
     Period Ndays    Flux Std.Err      SEP      L95      U95
1 July 2014    31 3060.67 113.787 115.4478 2840.609 3293.094
```

```
> # Compare to the mean of the daily values:
> with(Bad0714.day, mean(Flux))
```

```
[1] 3060.67
```