

```
> library(knitr)
> opts_chunk$set(
+ concordance=TRUE
+ )
```

# Instantaneous Time-Step Model

Dave Lorenz

July 26, 2017

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(siteNumber = "04027000"), data_type_cd=="uv",
+   select=c("parm_cd", "begin_date", "end_date"))
```

	parm_cd	begin_date	end_date
4	00010	2011-03-03	2017-07-26
14	00060	1986-10-01	2017-07-26
22	00065	2017-03-28	2017-07-26
27	00095	2011-03-06	2017-07-26
33	00300	2011-03-03	2017-07-26
43	00400	2011-03-17	2017-07-26
287	63680	2011-03-17	2017-07-26

```
> # 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"
[4] "Flow_Inst"      "Flow_Inst_cd" "SpecCond_Inst"
[7] "SpecCond_Inst_cd" "DO_Inst"      "DO_Inst_cd"
[10] "Turb_Inst"      "Turb_Inst_cd" "tz_cd"

> # 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)[which(names(BadData)=='dateTime.left')] <- "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
0	0	0	0
medium_cd	Chloride	dateTime	agency_cd
0	0	0	0
site_no.right	dateTime.right	Flow	Flow_cd
0	0	20	20
SpecCond	SpecCond_cd	DO	DO_cd
16	16	11	11
Turb	Turb_cd	tz_cd	
22	22	0	

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 and print the candidate model.
> BadChloride.lm <- 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="SPCC")
> print(BadChloride.lm)
```

\*\*\* Load Estimation \*\*\*

Station: Bad River near Odanah  
Constituent: Chloride

Number of Observations: 91  
 Number of Uncensored Observations: 91  
 Center of Decimal Time: 2012.573  
 Center of ln(Q): 6.7484  
 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	SPCC
1	NA	NA		83	-48.25	-12.17
2 - log(D0)	1	0.1114		84	-48.14	-16.56

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)	-8.7861	1.03501	-8.489	0
log(Flow)	1.4927	0.07159	20.851	0
fourier(dateTime)sin(k=1)	0.3321	0.06349	5.230	0
fourier(dateTime)cos(k=1)	0.2807	0.05480	5.122	0
log(SpecCond)	1.8499	0.15775	11.727	0
log(Turb)	-0.2885	0.04559	-6.328	0

#### AMLE Regression Statistics

Residual variance: 0.03693  
 R-squared: 97.22 percent  
 G-squared: 326.2 on 5 degrees of freedom  
 P-value: <0.0001  
 Prob. Plot Corr. Coeff. (PPCC):  
   r = 0.9689  
   p-value = 6e-04  
 Serial Correlation of Residuals: 0.4165

#### Variance Inflation Factors:

	VIF
log(Flow)	20.535
fourier(dateTime)sin(k=1)	3.278
fourier(dateTime)cos(k=1)	1.195
log(SpecCond)	9.487
log(Turb)	9.058

#### Comparison of Observed and Estimated Loads

Summary Stats: Loads in kg/d							
	Min	25%	50%	75%	90%	95%	Max
Est	787	3840	9570	19000	29400	34900	62600
Obs	889	3240	9610	21100	30000	34400	48100

#### Bias Diagnostics

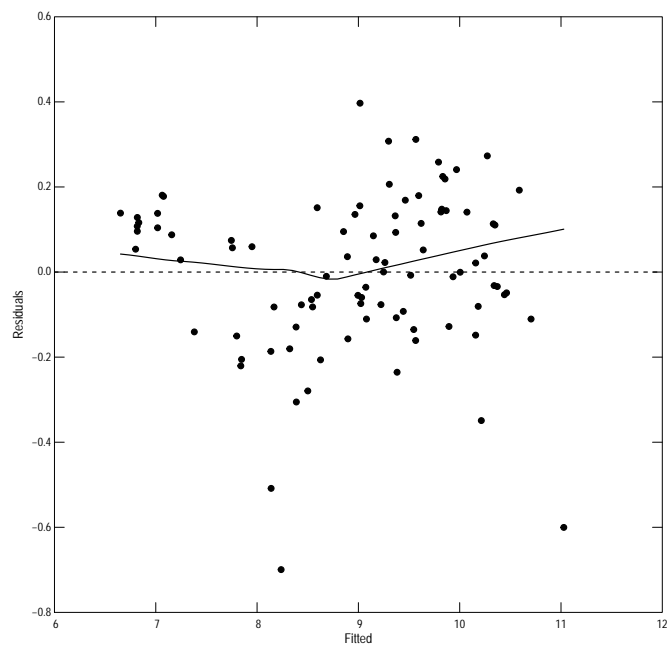
Bp: 0.5124 percent  
PLR: 1.005  
E: 0.8784

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.lm, which = 2, set.up=FALSE)  
> dev.off()
```

```
null device  
      1
```



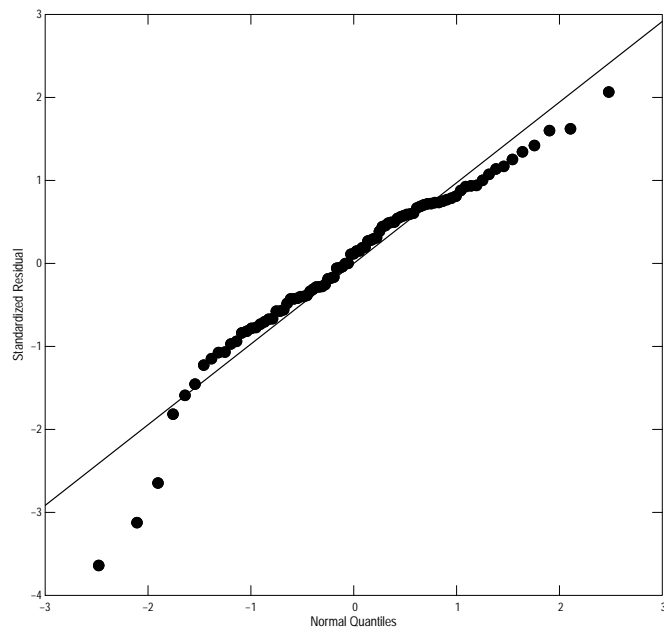
**Figure 1.** The residuals versus fitted graph.

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.lm, which = 5, set.up=FALSE)
> dev.off()
```

null device

1



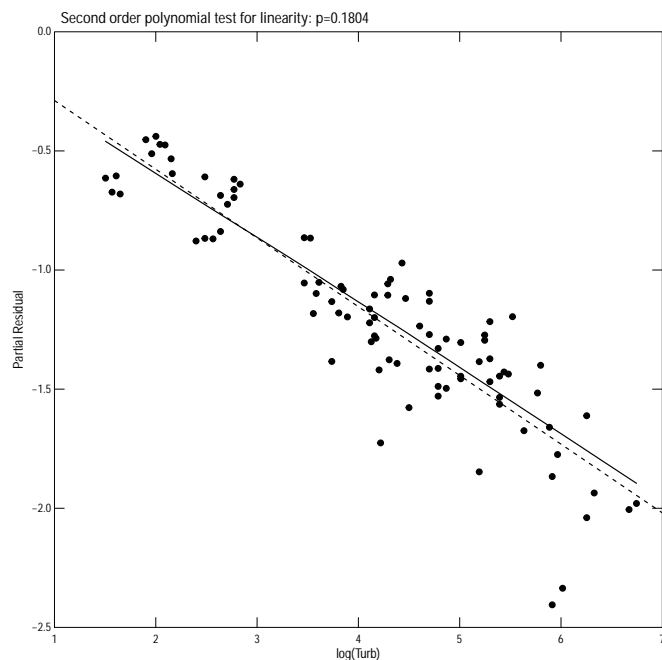
**Figure 2.** The residual Q-normal graph.

A complete review of the partial residual graphs is not included in this example. Only the partial residual for  $\log(\text{Turb})$  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.lm, which = "log(Turb)", set.up=FALSE)
> dev.off()
```

```
null device
      1
```





**Figure 3.** The partial residual for  $\log(\text{Turb})$  graph.

Build the model excluding  $\log(\text{DO})$  that was dropped in the subset selection procedure and changing  $\log(\text{Turb})$  to  $\text{Turb}$ .

```
> # 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: 91  
Number of Uncensored Observations: 91  
Center of Decimal Time: 2012.573  
Center of  $\ln(Q)$ : 6.7484

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)	-8.735913	1.156981	-7.551	0.0000
log(Flow)	1.377223	0.064373	21.395	0.0000
fourier(dateTime)sin(k=1)	0.166961	0.065130	2.564	0.0092
fourier(dateTime)cos(k=1)	0.395459	0.054757	7.222	0.0000
log(SpecCond)	1.822214	0.171909	10.600	0.0000
Turb	-0.001286	0.000243	-5.293	0.0000

AMLE Regression Statistics

Residual variance: 0.04086

R-squared: 96.93 percent

G-squared: 317 on 5 degrees of freedom

P-value: <0.0001

Prob. Plot Corr. Coeff. (PPCC):

r = 0.9486

p-value = 0

Serial Correlation of Residuals: 0.3105

Variance Inflation Factors:

	VIF
log(Flow)	15.008
fourier(dateTime)sin(k=1)	3.117
fourier(dateTime)cos(k=1)	1.078
log(SpecCond)	10.184
Turb	3.492

Comparison of Observed and Estimated Loads

-----

Summary Stats: Loads in kg/d

-----

	Min	25%	50%	75%	90%	95%	Max
Est	875	3720	10000	19900	29000	35500	56000
Obs	889	3240	9610	21100	30000	34400	48100

Bias Diagnostics

-----

Bp: -0.06404 percent

PLR: 0.9994

E: 0.9048

Selected Concentration Model:

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

Model coefficients:

	Estimate	Std. Error	z-score	p-value
(Intercept)	-9.630608	1.156981	-8.324	0.0000
log(Flow)	0.377223	0.064373	5.860	0.0000
fourier(dateTime)sin(k=1)	0.166961	0.065130	2.564	0.0092
fourier(dateTime)cos(k=1)	0.395459	0.054757	7.222	0.0000
log(SpecCond)	1.822214	0.171909	10.600	0.0000
Turb	-0.001286	0.000243	-5.293	0.0000

AMLE Regression Statistics

Residual variance: 0.04086

R-squared: 73.65 percent

G-squared: 121.4 on 5 degrees of freedom

P-value: <0.0001

Prob. Plot Corr. Coeff. (PPCC):

    r = 0.9486

    p-value = 0

Serial Correlation of Residuals: 0.3105

Comparison of Observed and Estimated Concentrations

-----  
Summary Stats: Concentrations in mg/l

	Min	25%	50%	75%	90%	95%	Max
Est	1.33	2.01	2.52	2.98	3.75	5.03	5.47
Obs	0.76	1.86	2.47	3.39	3.98	4.47	5.24

Bias Diagnostics

-----  
Bp: 0.4076 percent

PCR: 1.004

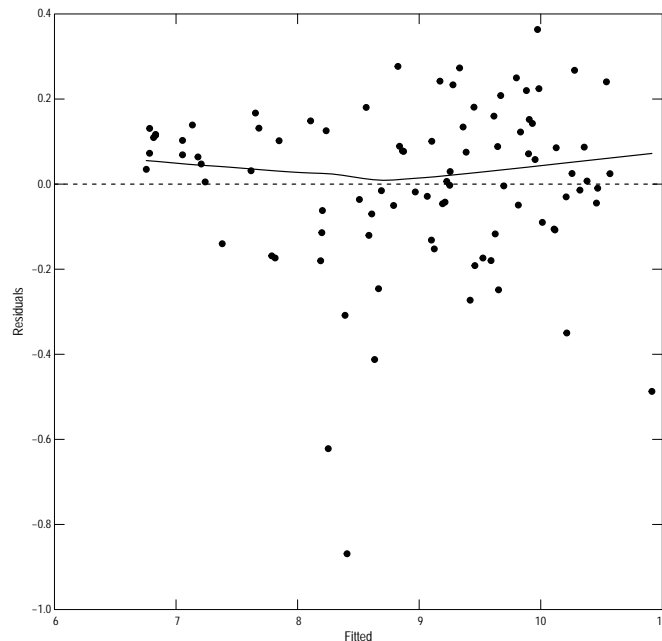
E: 0.8057

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 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.lm, which = 2, set.up=FALSE)
> dev.off()
```

```
null device
      1
```



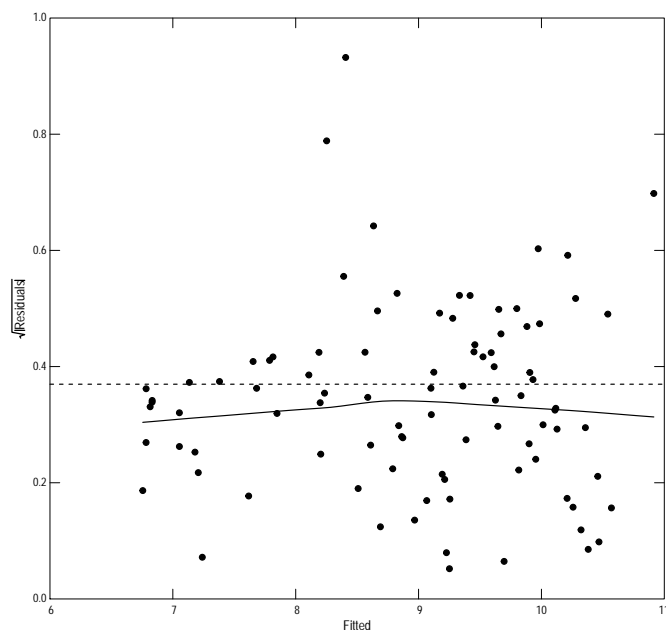
**Figure 4.** The residuals versus fitted graph for the revised model.

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 graph.
> setSweave("graph05", 6, 6)
> plot(BadChloride.lm, which = 3, set.up=FALSE)
> dev.off()
```

null device  
1

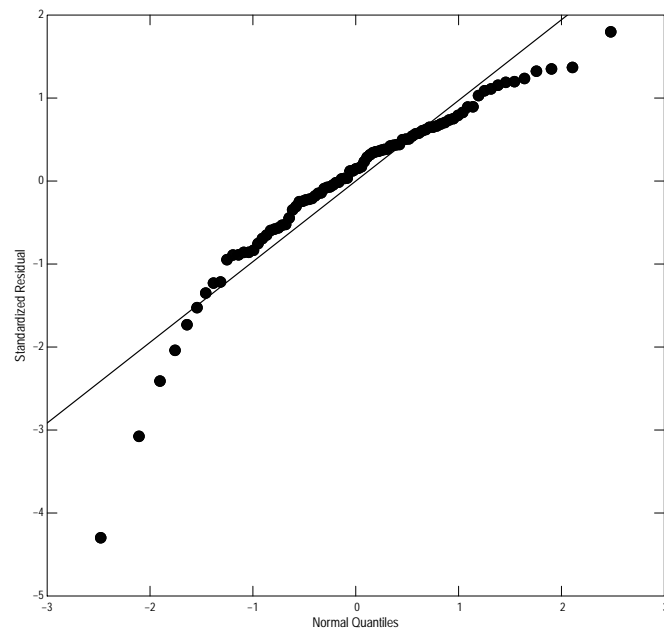


**Figure 5.** The S-L graph for the revised model.

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.lm, which = 5, set.up=FALSE)
> dev.off()
```

```
null device
1
```

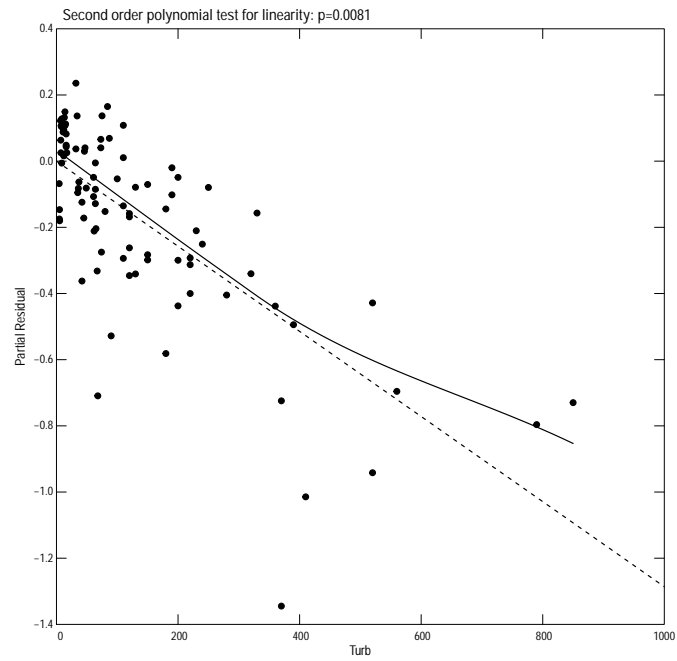


**Figure 6.** The residual Q-normal graph for the revised model.

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.lm, 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 on 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 unnecessary 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 concentrations
> Bad063014.est <- predConc(BadChloride.lmr, Bad063014, by="unit")
> # Display the first and last few rows.
> head(Bad063014.est)
```

	Date	Flow	Conc	Std.Err	SEP	L95
1	2014-06-30 00:00:00	909	3.206783	0.1701770	0.6766423	2.072006
2	2014-06-30 00:15:00	909	3.190345	0.1669653	0.6725893	2.062188
3	2014-06-30 00:30:00	909	3.202481	0.1693782	0.6755914	2.069423
4	2014-06-30 00:45:00	909	3.202376	0.1693835	0.6755719	2.069352
5	2014-06-30 01:00:00	903	3.194332	0.1679888	0.6736327	2.064487
6	2014-06-30 01:15:00	903	3.135483	0.1609324	0.6602456	2.027796

U95

1	4.751496
2	4.725648
3	4.744757
4	4.744608
5	4.732072
6	4.642401

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

	Date	Flow	Conc	Std.Err	SEP	L95
91	2014-06-30 22:30:00	770	2.902802	0.1333234	0.6076208	1.882311
92	2014-06-30 22:45:00	770	2.891596	0.1313947	0.6049664	1.875470
93	2014-06-30 23:00:00	770	2.902618	0.1333370	0.6075870	1.882185



```

94 2014-06-30 23:15:00 764 2.886619 0.1308481 0.6038555 1.872338
95 2014-06-30 23:30:00 764 2.893919 0.1321357 0.6055907 1.876786
96 2014-06-30 23:45:00 758 2.827291 0.1251479 0.5907996 1.834746

```

```

      U95

```

```

91 4.288640
92 4.271298
93 4.288380
94 4.263768
95 4.275082
96 4.174493

```

```

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

```

```

      Date      Flow      Conc  Std.Err      SEP      L95      U95
1 2014-06-30 829.4792 3.022303 0.1465827 0.6345756 1.957127 4.470139

```

```

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

```

```

[1] 3.022303

```

### 3 Aggregate Loads

Estimating concentrations or loads by day assumes, but does not require consistent number of unit values per day. Both `predLoad` and `predConc` assume that inconsistent number of unit values per day are due to missing values and return missing values for the estimates for days that do not have the average number of observations per day. Inconsistent number of observations per day can be the result of deleted bad values, maintenance, or a change in frequency of sampling. The data can be resampled to a uniform number per day using the `resampleUVdata` function or the check can be suppressed by setting the `allow.incomplete` argument to `TRUE`.

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 unnecessary 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	4699.580	199.7177	982.6638	3048.889	6940.518
2	2014-07-02	733.5521	5106.895	229.8322	1074.9831	3303.289	7560.287
3	2014-07-03	702.1146	4780.645	211.0034	1003.4463	3096.201	7070.005
4	2014-07-04	519.5417	3341.828	131.7786	697.7895	2169.380	4932.858
5	2014-07-05	399.8750	2640.163	102.3442	549.5820	1716.231	3892.813
6	2014-07-06	336.1875	2281.347	90.2305	474.6432	1483.324	3363.124

```

> tail(Bad0714.day)

```

	Date	Flow	Flux	Std.Err	SEP	L95
26	2014-07-26	204.7604	1515.465	65.50596	316.3818	983.8545
27	2014-07-27	200.3438	1597.807	69.70615	334.8495	1035.5505
28	2014-07-28	275.9167	2396.812	112.83862	502.8753	1552.5920
29	2014-07-29	277.8333	2264.661	102.88114	474.1389	1468.3797
30	2014-07-30	372.5625	3104.713	152.48636	655.0647	2006.1106
31	2014-07-31	477.9062	3808.447	193.80864	803.8597	2460.3980

U95

26	2236.833
27	2361.625
28	3544.064
29	3346.084
30	4600.155
31	5643.657

```

> # 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	2878.334	116.7498	119.231	2651.695	3118.992

```

> # 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 desirable to use the approximation.
> predLoad(BadChloride.lr, Bad0714, by="month", seopt="app")

```

	Period	Ndays	Flux	Std.Err	SEP	L95	U95
1	July 2014	31	2878.334	112.6392	119.2305	2651.696	3118.991

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

```
[1] 2878.334
```