A comparison of R's `tidem()` and Matlab's `t_tide` for tidal analysis

Clark Richards

2020-02-04

Introduction

A recent issue posted to the `oce` Github page found discrepancies between the oce function `tidem()` and the Matlab package `t_tide`. The example cited uses data from the NOAA tides and currents website, which have been downloaded here in the CSV file `"CO-OPS_8720218_wl.csv"`.

The purpose of this report is to document a series of tests and comparisons to see:

1. If the two functions actually produce different tidal fits under the same conditions, and
2. What conditions (e.g. arguments, etc) provide the best match for tidal fits and predictions.

Data

The data, obtained from a link on the GH issue, are provided in a csv file with the following form:

```
Date,Time (GMT),Predicted (m),Preliminary (m),Verified (m)
2019/08/15,00:00,0.691,-,0.751
2019/08/15,00:06,0.708,-,0.777
2019/08/15,00:12,0.723,-,0.771
2019/08/15,00:18,0.738,-,0.801
2019/08/15,00:24,0.751,-,0.837
2019/08/15,00:30,0.762,-,0.863
2019/08/15,00:36,0.773,-,0.875
2019/08/15,00:42,0.781,-,0.895
2019/08/15,00:48,0.789,-,0.905
```

Default tidal fit in R

The data can be loaded into R, and processed using the `tidem()` function in oce with:

```
library(oce)
```

```
## Loading required package: testthat
```

```
## Loading required package: gsw
```

```
e <- read.csv("CO-OPS_8720218_wl.csv")
t <- as.POSIXct(paste(e$Date, e$Time..GMT.), tz="UTC")
e <- e$Verified..m.
sl <- as.sealevel(e, t)
m <- tidem(sl)
```

```
## Note: the tidal record is too short to fit for constituents:  SA SSA MSM MF SIG1 RHO1 TAU1 BET1 CHI1
```

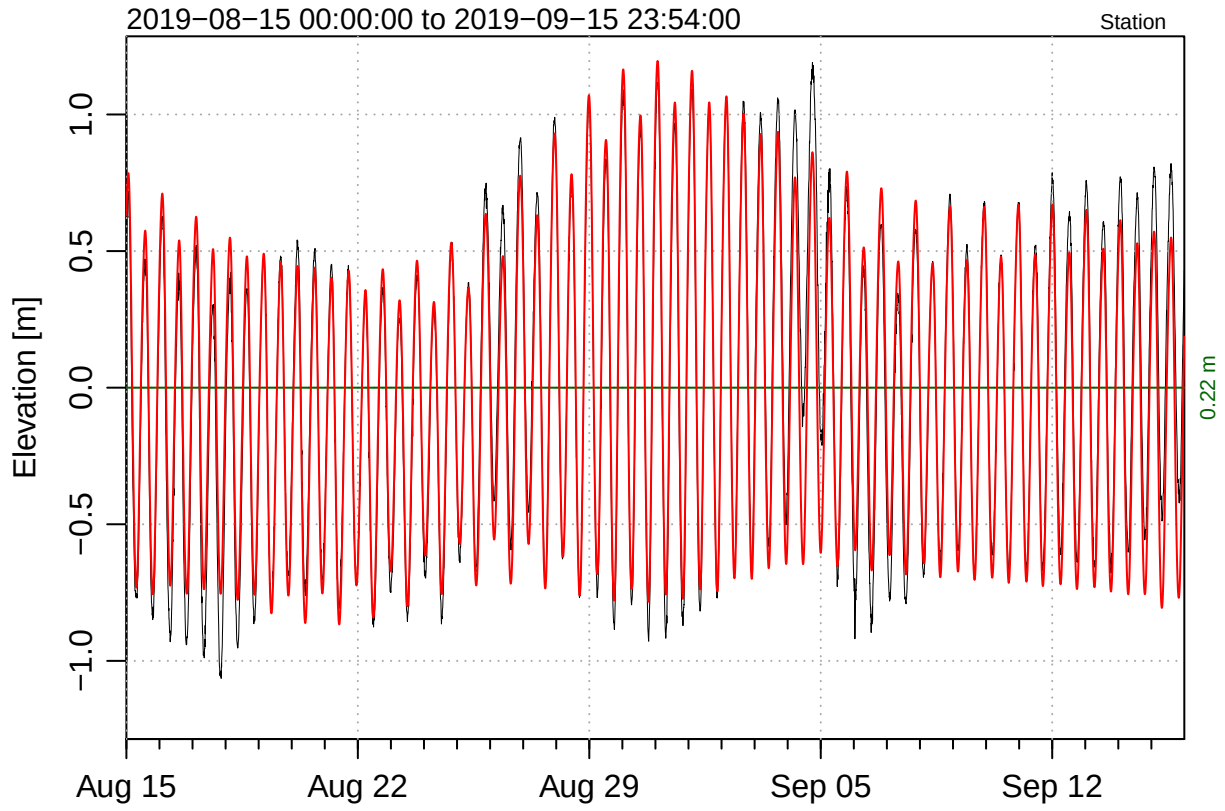
The results can be viewed by looking at a `summary()` of the object:

```
summary(m)
```

```
## tidem summary
## -------------
##
## Call:
## tidem(t = sl)
## RMS misfit to data:  0.1381428
##
## Fitted Model:
##          Freq Amplitude Phase       p
## Z0   0.000000  0.230167   0.0 < 2e-16 ***
## MM   0.001512  0.158694  28.8 < 2e-16 ***
## MSF  0.002822  0.067187 358.3 2.8e-11 ***
## ALP1 0.034397  0.009362  30.3 0.01158 *
## 2Q1  0.035706  0.004546 132.1 0.17815
## Q1   0.037219  0.012508 159.9 0.00464 **
## O1   0.038731  0.060395 202.5 3.5e-09 ***
## NO1  0.040269  0.005807 344.1 0.39586
## K1   0.041781  0.079132 229.0 < 2e-16 ***
## J1   0.043293  0.004263 219.1 0.22700
## OO1  0.044831  0.003579 266.1 0.27172
## UPS1 0.046343  0.002460 107.4 0.50203
## EPS2 0.076177  0.024778 355.7 0.00012 ***
## MU2  0.077689  0.035582  17.1 < 2e-16 ***
## N2   0.078999  0.184835  15.5 < 2e-16 ***
## M2   0.080511  0.666149  29.6 < 2e-16 ***
## L2   0.082024  0.070815  35.4 < 2e-16 ***
## S2   0.083333  0.099766  49.4 < 2e-16 ***
## ETA2 0.085074  0.017726 307.5 0.06386 .
## MO3  0.119242  0.011012 311.1 0.00088 ***
## M3   0.120767  0.004126 201.6 0.20877
## MK3  0.122292  0.008852 169.7 0.24359
## SK3  0.125114  0.001332 144.4 0.68104
## MN4  0.159511  0.015209 204.1 0.00311 **
## M4   0.161023  0.036210 199.3 6.3e-08 ***
## SN4  0.162333  0.015240 358.9 0.44842
## MS4  0.163845  0.011808 211.4 0.00339 **
## S4   0.166667  0.005324 290.4 0.21890
## 2MK5 0.202804  0.001819 243.1 0.59182
## 2SK5 0.208447  0.003879  29.9 0.26156
## 2MN6 0.240022  0.004262 196.2 0.33584
## M6   0.241534  0.008063 233.1 0.01863 *
## 2MS6 0.244356  0.006855 247.2 0.12272
## 2SM6 0.247178  0.002769 273.3 0.58167
## 3MK7 0.283315  0.000578  83.6 0.88729
## M8   0.322046  0.002084  52.1 0.51530
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## * Processing Log
##     - 2020-02-06 15:23:21 UTC: `create 'tidem' object`
##     - 2020-02-06 15:23:21 UTC: `tidem(t = sl)`
```

and visualized by plotting the original data along with a prediction based on the tidal fit (note that the `plot()` method for `sealevel` objects removes the mean, so to compare the original and predicted time series we have to add it back in through the `Z0` consituent):

```
plot(sl, which=1)
lines(sl[['time']], predict(m)-m[['amplitude']][1], col=2)
```



Default tidal fit in Matlab

To run the Matlab code requires a working installation of Matlab, with the `t_tide` package installed in a location known to the Matlab path. The following code was run:

```
data = readtable('CO-OPS_8720218_wl.csv');

time = strcat(data.Date, {' '}, data.Time_GMT_);
time = datenum(time, 'yyyy/mm/ddHH:MM');

timeh = (time - time(1))*24;
interval = (time(2) - time(1))*24;

% [NAME,FREQ,TIDECON,XOUT]=t_tide(data.Verified_m_, 'interval', interval, 'start time', time(1) ...
%    , 'rayleigh', ['M2']);
[NAME,FREQ,TIDECON,XOUT]=t_tide(data.Verified_m_, 'interval', interval, 'start time', time(1));

clf
plot(time, data.Verified_m_, 'k')
```

```
hold on
plot(time, XOUT, 'r')

rmse = sqrt(mean((data.Verified_m_ - XOUT).^2));

save('ml_prediction.mat', 'XOUT')
```

which produced the following summary output of the fit:

```
   number of standard constituents used: 35
   Points used: 7679 of 7680
   percent of var residual after lsqfit/var original:  6.81 %
   Greenwich phase computed, no nodal corrections
   Using nonlinear bootstrapped error estimates
   Generating prediction without nodal corrections, SNR is 2.000000
   percent of var residual after synthesis/var original: 12.18 %
-----------------------------------
date: 04-Feb-2020
nobs = 7680,  ngood = 7679,  record length (days) = 32.00
start time: 15-Aug-2019
rayleigh criterion = 1.0
Greenwich phase computed, no nodal corrections

x0= 0.23, x trend= 0

var(x)= 0.28007   var(xp)= 0.2465   var(xres)= 0.034125
percent var predicted/var original= 88.0 %

     tidal amplitude and phase with 95% CI estimates

tide   freq       amp     amp_err    pha    pha_err     snr
 MM   0.0015122    0.1587    0.121    28.77    46.90      1.7
 MSF  0.0028219    0.0672    0.104   358.28   104.86     0.41
*ALP1 0.0343966    0.0094    0.006    30.03    30.14      2.8
 2Q1  0.0357064    0.0046    0.004   132.40    69.47      1.3
*Q1   0.0372185    0.0125    0.006   159.80    24.46        5
*O1   0.0387307    0.0604    0.005   202.53     4.95  1.5e+02
 NO1  0.0402686    0.0058    0.005   343.62    46.43      1.5
*K1   0.0417807    0.0791    0.005   228.94     3.62  2.2e+02
 J1   0.0432929    0.0043    0.005   219.76    68.73     0.87
 OO1  0.0448308    0.0035    0.004   265.62    74.38     0.68
 UPS1 0.0463430    0.0024    0.004   106.40   117.73     0.46
*EPS2 0.0761773    0.0247    0.016   355.61    44.84      2.4
*MU2  0.0776895    0.0356    0.017    17.17    28.57      4.6
*N2   0.0789992    0.1848    0.018    15.50     5.57  1.1e+02
*M2   0.0805114    0.6661    0.018    29.63     1.50  1.3e+03
*L2   0.0820236    0.0708    0.018    35.42    13.18       15
*S2   0.0833333    0.0997    0.016    49.44    10.11       38
 ETA2 0.0850736    0.0177    0.017   307.72    59.49      1.1
*MO3  0.1192421    0.0110    0.005    39.52    26.42      5.4
 M3   0.1207671    0.0041    0.004   200.94    67.08     0.85
*MK3  0.1222921    0.0089    0.005    47.18    35.51      3.5
 SK3  0.1251141    0.0014    0.004    35.31   164.89     0.12
*MN4  0.1595106    0.0152    0.010   180.22    29.97      2.6
*M4   0.1610228    0.0362    0.010   176.40    15.54       14
```

```
*SN4  0.1623326    0.0153    0.009   346.08    33.64      2.6
 MS4  0.1638447    0.0118    0.009   199.82    44.17      1.6
 S4   0.1666667    0.0053    0.007   289.84    85.96     0.52
 2MK5 0.2028035    0.0018    0.002   107.41    69.86        1
*2SK5 0.2084474    0.0038    0.002   279.05    33.71      2.9
 2MN6 0.2400221    0.0043    0.004   161.02    62.71      1.1
*M6   0.2415342    0.0080    0.004   198.63    27.78        4
*2MS6 0.2443561    0.0068    0.004   224.05    38.05      2.4
 2SM6 0.2471781    0.0027    0.004   260.87    91.61     0.49
 3MK7 0.2833149    0.0006    0.001   299.61   135.59     0.26
*M8   0.3220456    0.0021    0.001     6.36    37.23      2.6
```

The prediction, stored in the variable `XOUT`, was saved to a mat file which can be read into R and compared with the `tidem()` prediction:

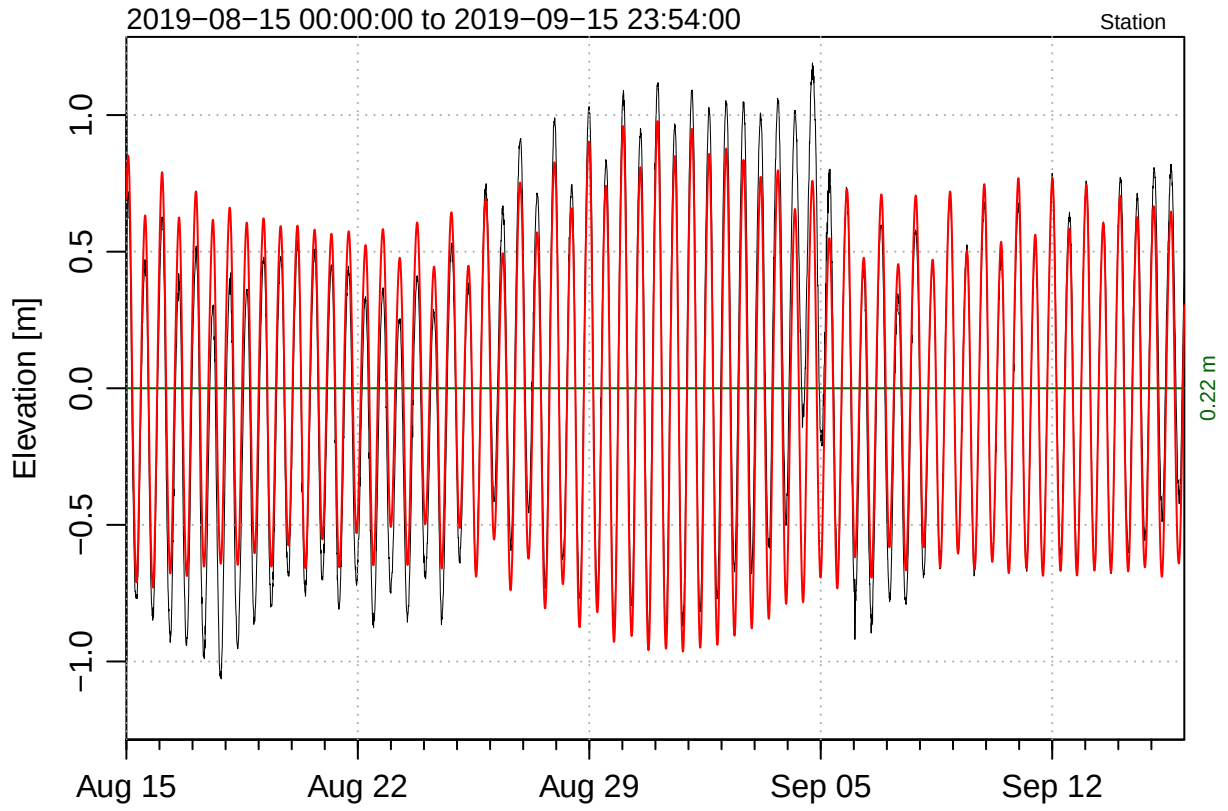
```
library(R.matlab)
```

```
## R.matlab v3.6.2 (2018-09-26) successfully loaded. See ?R.matlab for help.
```

```
##
## Attaching package: 'R.matlab'
```

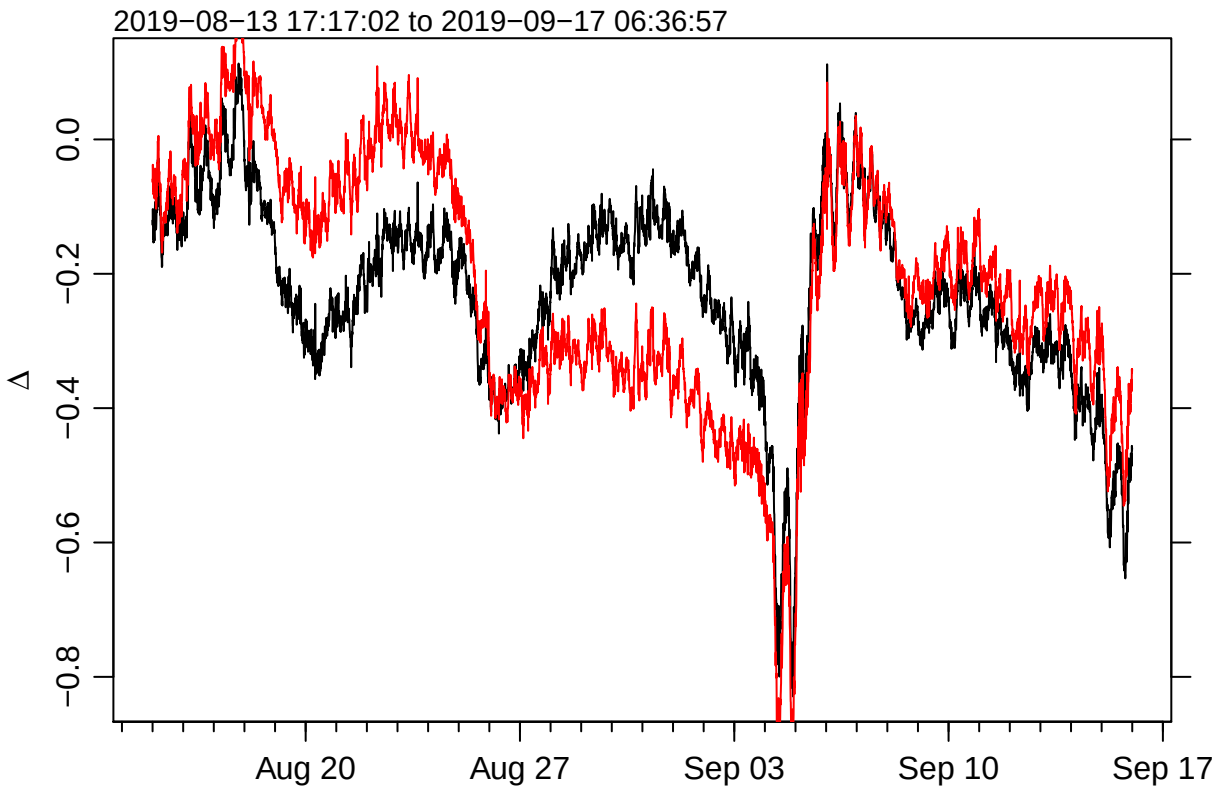
```
## The following objects are masked from 'package:base':
##
##     getOption, isOpen
```

```
xout <- drop(readMat('ml_prediction.mat')$XOUT)
plot(sl, which=1)
lines(sl[['time']], xout, col=2)
```



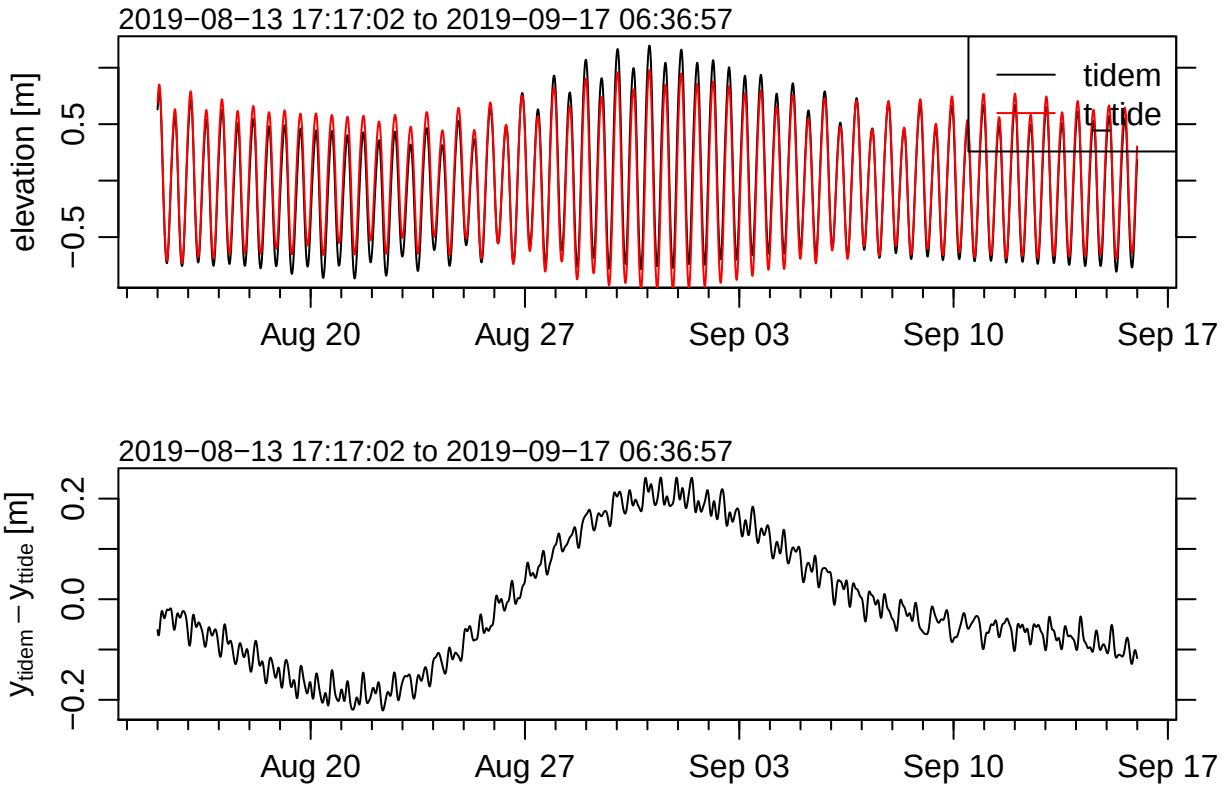
Comparing the residuals from the two predictions can give a sense of the differences:

```
oce.plot.ts(sl[['time']], predict(m)-m[['amplitude']][1] - sl[['elevation']],
            ylab=expression(Delta))
lines(sl[['time']], xout - sl[['elevation']], col=2)
```



The `tidem()` and `t_tide` predictions can be plotted:

```
par(mfrow=c(2, 1))
oce.plot.ts(sl[['time']], predict(m)-m[['amplitude']][1],
            ylab='elevation [m]')
lines(sl[['time']], xout, col=2)
legend('topright', c('tidem', 't_tide'), lty=1, col=1:2)

oce.plot.ts(sl[['time']], predict(m)-m[['amplitude']][1] - xout,
            ylab=expression(y[tidem]-y[ttide]~"["*m*"]"))
```



The two predictions are clearly not the same – there is variability at very low frequencies (with amplitude of about 0.2 m) as well as some higher frequency variability. Comparing the root mean square error of the two predictions gives

```
rmse <- function(x, xr) {
    sqrt( mean( (x - xr)^2 ) )
}
rmse(predict(m), e)
```

```
## [1] 0.1381338
```

```
rmse(xout+0.23, e) # need to add the mean back in
```

```
## [1] 0.1852284
```

Why do these two fits look different? To narrow this down lets try fitting with a specified set of tidal constituents (rather than letting the `tidem` and `t_tide` defaults take over).

Comparison of tidem and t_tide fitting parameters

Let's take a closer look at the reported output tables to see where the differences might be coming from. First, we note that the `t_tide` approach lists the `Z0` component as `x0`. This "constituent" amounts to the mean value of the time series. `t_tide` permits fitting a linear trend to the data as well as an offset from zero, but that is not done here.

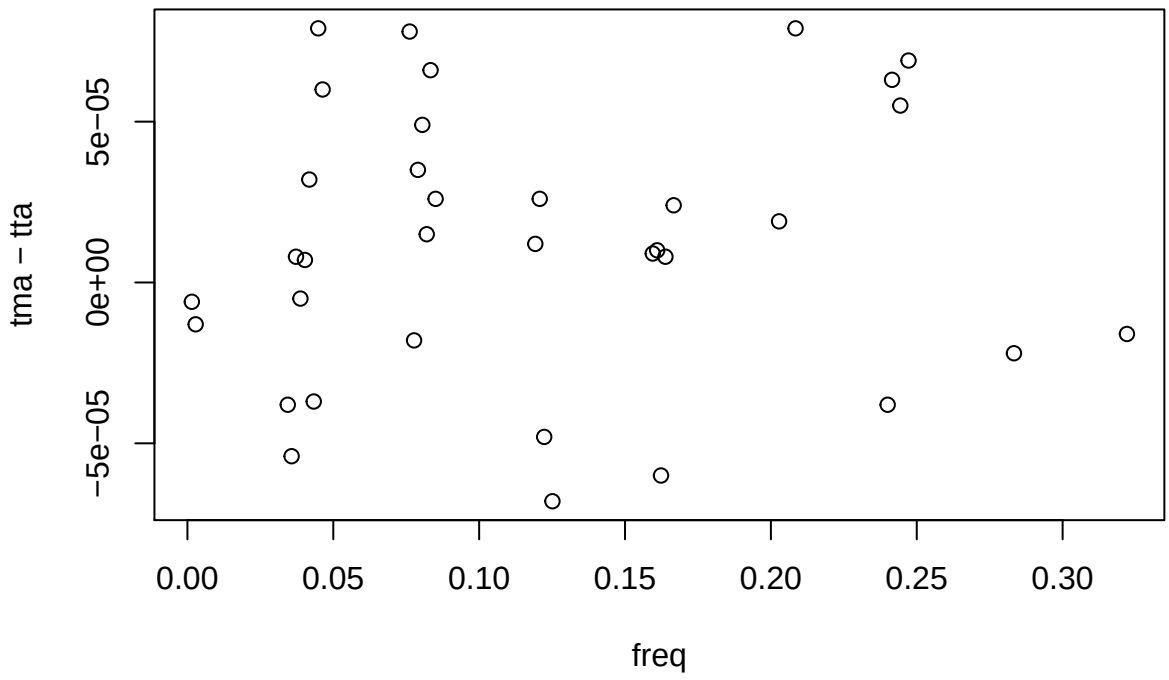
```
##          Freq Amplitude Phase       p        tide   freq       amp     amp_err    pha    pha_err
## Z0   0.000000  0.230167   0.0 < 2e-16 ***
## MM   0.001512  0.158694  28.8 < 2e-16 ***     MM   0.0015122    0.1587    0.121    28.77    46.90
```

```
## MSF  0.002822  0.067187 358.3 2.8e-11 ***     MSF  0.0028219    0.0672    0.104   358.28   104.86
## ALP1 0.034397  0.009362  30.3 0.01158 *      *ALP1 0.0343966    0.0094    0.006    30.03    30.14
## 2Q1  0.035706  0.004546 132.1 0.17815         2Q1  0.0357064    0.0046    0.004   132.40    69.47
## Q1   0.037219  0.012508 159.9 0.00464 **     *Q1   0.0372185    0.0125    0.006   159.80    24.46
## O1   0.038731  0.060395 202.5 3.5e-09 ***    *O1   0.0387307    0.0604    0.005   202.53     4.95  1
## NO1  0.040269  0.005807 344.1 0.39586         NO1  0.0402686    0.0058    0.005   343.62    46.43
## K1   0.041781  0.079132 229.0 < 2e-16 ***    *K1   0.0417807    0.0791    0.005   228.94     3.62  2
## J1   0.043293  0.004263 219.1 0.22700         J1   0.0432929    0.0043    0.005   219.76    68.73
## OO1  0.044831  0.003579 266.1 0.27172         OO1  0.0448308    0.0035    0.004   265.62    74.38
## UPS1 0.046343  0.002460 107.4 0.50203         UPS1 0.0463430    0.0024    0.004   106.40   117.73
## EPS2 0.076177  0.024778 355.7 0.00012 ***    *EPS2 0.0761773    0.0247    0.016   355.61    44.84
## MU2  0.077689  0.035582  17.1 < 2e-16 ***    *MU2  0.0776895    0.0356    0.017    17.17    28.57
## N2   0.078999  0.184835  15.5 < 2e-16 ***    *N2   0.0789992    0.1848    0.018    15.50     5.57  1
## M2   0.080511  0.666149  29.6 < 2e-16 ***    *M2   0.0805114    0.6661    0.018    29.63     1.50  1
## L2   0.082024  0.070815  35.4 < 2e-16 ***    *L2   0.0820236    0.0708    0.018    35.42    13.18
## S2   0.083333  0.099766  49.4 < 2e-16 ***    *S2   0.0833333    0.0997    0.016    49.44    10.11
## ETA2 0.085074  0.017726 307.5 0.06386 .       ETA2 0.0850736    0.0177    0.017   307.72    59.49
## MO3  0.119242  0.011012 311.1 0.00088 ***    *MO3  0.1192421    0.0110    0.005    39.52    26.42
## M3   0.120767  0.004126 201.6 0.20877         M3   0.1207671    0.0041    0.004   200.94    67.08
## MK3  0.122292  0.008852 169.7 0.24359        *MK3  0.1222921    0.0089    0.005    47.18    35.51
## SK3  0.125114  0.001332 144.4 0.68104         SK3  0.1251141    0.0014    0.004    35.31   164.89
## MN4  0.159511  0.015209 204.1 0.00311 **     *MN4  0.1595106    0.0152    0.010   180.22    29.97
## M4   0.161023  0.036210 199.3 6.3e-08 ***    *M4   0.1610228    0.0362    0.010   176.40    15.54
## SN4  0.162333  0.015240 358.9 0.44842        *SN4  0.1623326    0.0153    0.009   346.08    33.64
## MS4  0.163845  0.011808 211.4 0.00339 **      MS4  0.1638447    0.0118    0.009   199.82    44.17
## S4   0.166667  0.005324 290.4 0.21890         S4   0.1666667    0.0053    0.007   289.84    85.96
## 2MK5 0.202804  0.001819 243.1 0.59182         2MK5 0.2028035    0.0018    0.002   107.41    69.86
## 2SK5 0.208447  0.003879  29.9 0.26156        *2SK5 0.2084474    0.0038    0.002   279.05    33.71
## 2MN6 0.240022  0.004262 196.2 0.33584         2MN6 0.2400221    0.0043    0.004   161.02    62.71
## M6   0.241534  0.008063 233.1 0.01863 *      *M6   0.2415342    0.0080    0.004   198.63    27.78
## 2MS6 0.244356  0.006855 247.2 0.12272        *2MS6 0.2443561    0.0068    0.004   224.05    38.05
## 2SM6 0.247178  0.002769 273.3 0.58167         2SM6 0.2471781    0.0027    0.004   260.87    91.61
## 3MK7 0.283315  0.000578  83.6 0.88729         3MK7 0.2833149    0.0006    0.001   299.61   135.59
## M8   0.322046  0.002084  52.1 0.51530        *M8   0.3220456    0.0021    0.001     6.36    37.23
```

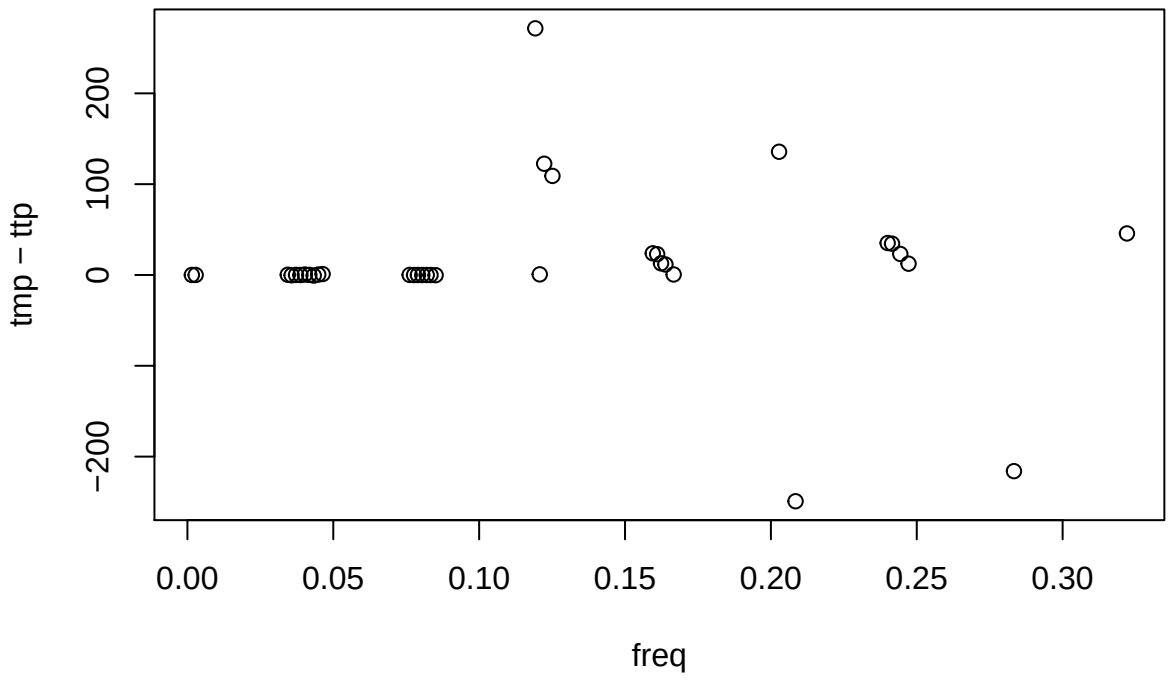
Note that by default `tidem` and `t_tide` fit the same number of constituents, in this case

Let's compare the fitted amplitudes and phases for the two approaches. Plotting the fit amplitudes and phases as a function of frequency shows that the fit amplitudes are all quite close, while for some constituents the phases are different.

```
plot(freq, tma-tta)
```

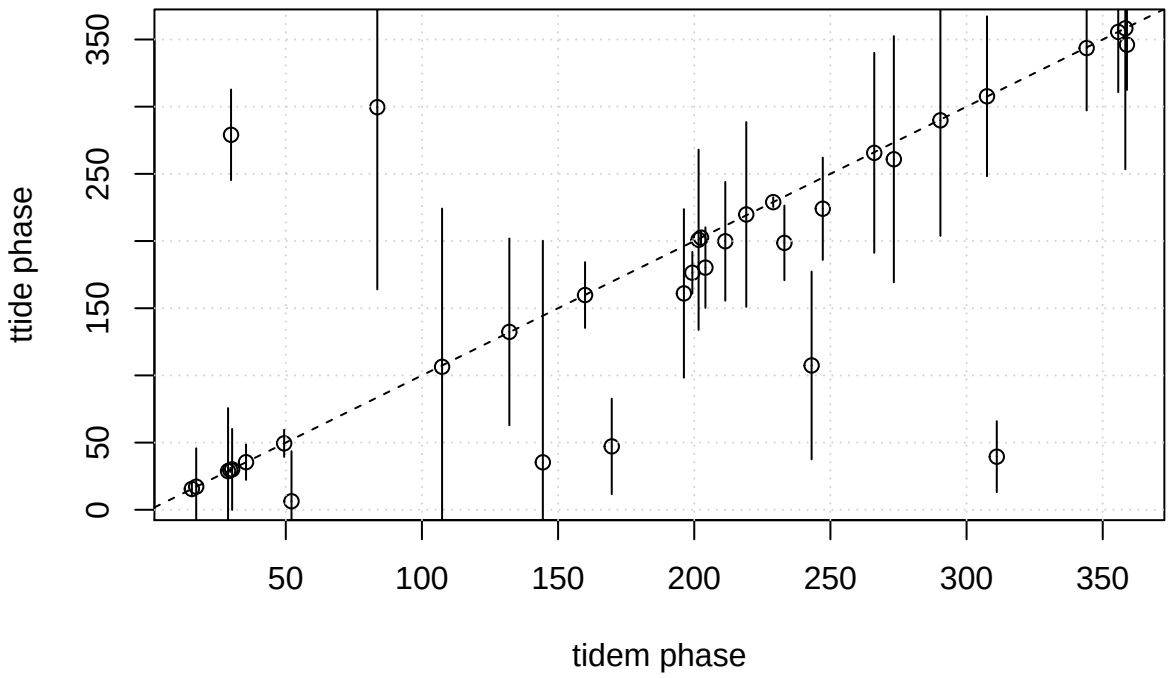



```
plot(freq, tmp-ttp)
```



`t_tide` provides uncertainty in the fitted parameters (I'm sure `tidem` must too . . .), which can help see

```
plot(tmp, ttp, xlab='tidem phase', ylab='ttide phase')
grid()
abline(0, 1, lty=2)
errorbars(tmp, ttp, 0, ttpe)
```



Tests with limited constituents

The constituents with the top 10 largest amplitudes are:

```
o <- order(tma, decreasing = TRUE)[1:10]
data.frame(name=name[o], tma=tma[o], tta=tta[o], tmp=tmp[o], ttp=ttp[o])
```

```
##    name      tma    tta   tmp    ttp
## 1    M2 0.666149 0.6661  29.6  29.63
## 2    N2 0.184835 0.1848  15.5  15.50
## 3    MM 0.158694 0.1587  28.8  28.77
## 4    S2 0.099766 0.0997  49.4  49.44
## 5    K1 0.079132 0.0791 229.0 228.94
## 6    L2 0.070815 0.0708  35.4  35.42
## 7   MSF 0.067187 0.0672 358.3 358.28
## 8    O1 0.060395 0.0604 202.5 202.53
## 9    M4 0.036210 0.0362 199.3 176.40
## 10  MU2 0.035582 0.0356  17.1  17.17
```

Note that the fit amplitudes are all very close, as are the phases – except for the M4 constituent. Let's try fitting but specifying only those components:

```
m <- tidem(sl, constituents=name[o])
summary(m)
```

```
## tidem summary
```

```
## -------------
##
## Call:
## tidem(t = sl, constituents = name[o])
## RMS misfit to data:  0.142018
##
## Fitted Model:
##        Freq Amplitude Phase       p
## Z0  0.00000   0.23025   0.0 < 2e-16 ***
## MM  0.00151   0.15849  28.7 < 2e-16 ***
## MSF 0.00282   0.06730 358.5 1.2e-10 ***
## O1  0.03873   0.05963 203.4 5.5e-10 ***
## K1  0.04178   0.07884 228.7 < 2e-16 ***
## MU2 0.07769   0.03241  19.5 < 2e-16 ***
## N2  0.07900   0.18665  15.4 < 2e-16 ***
## M2  0.08051   0.66415  29.7 < 2e-16 ***
## L2  0.08202   0.06876  37.1 < 2e-16 ***
## S2  0.08333   0.09720  51.5 < 2e-16 ***
## M4  0.16102   0.03503 200.1 7.7e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## * Processing Log
##     - 2020-02-06 15:23:23 UTC: `create 'tidem' object`
##     - 2020-02-06 15:23:23 UTC: `tidem(t = sl, constituents = name[o])`
```

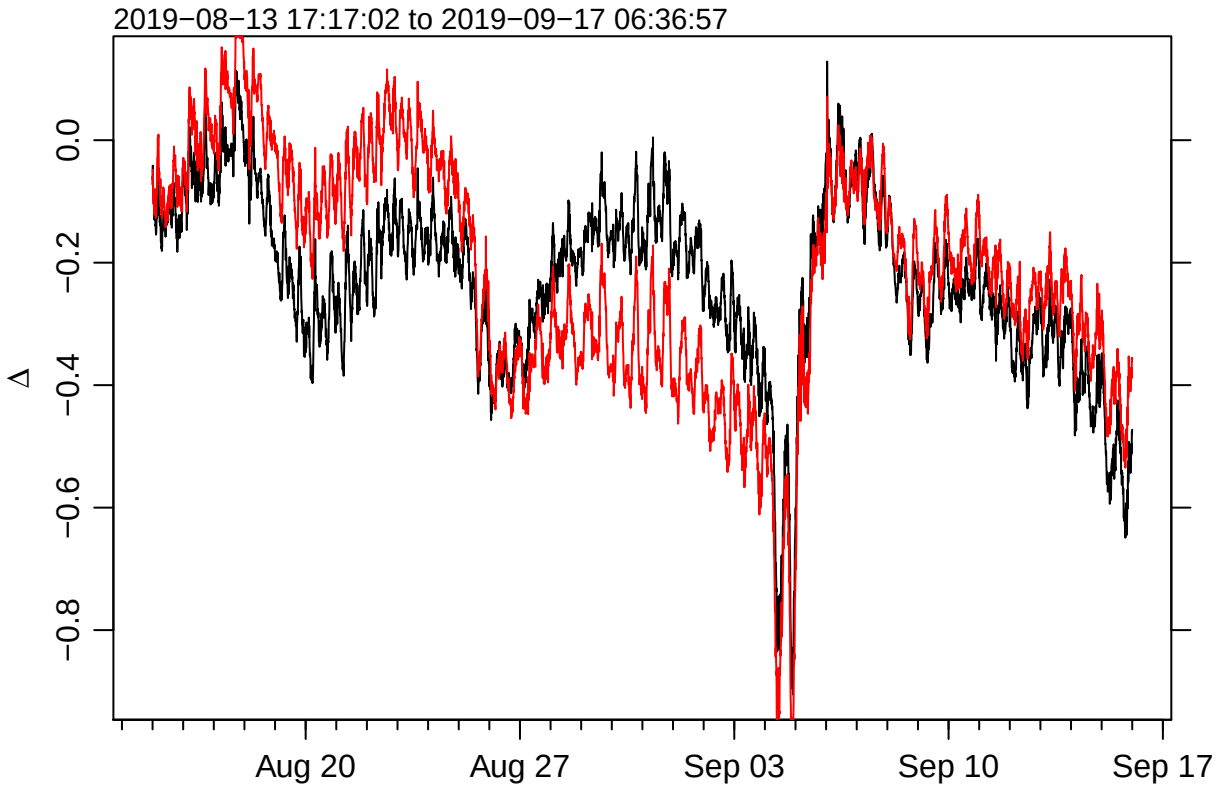
And in Matlab:

```
[NAME,FREQ,TIDECON,XOUT]=t_tide(data.Verified_m_, 'interval', interval, 'start time', time(1) ...
   , 'rayleigh', ['M2 '; 'N2 ' ; 'MM '; 'S2 '; 'K1 '; 'L2 '; 'MSF'; 'O1 '; 'M4 '; 'MU2']);
save('ml_prediction_ten_consituents.mat', 'XOUT')
```

```
number of standard constituents used: 10
   Points used: 7679 of 7680
   percent of var residual after lsqfit/var original:  7.20 %
   Greenwich phase computed, no nodal corrections
   Using nonlinear bootstrapped error estimates
   Generating prediction without nodal corrections, SNR is 2.000000
   percent of var residual after synthesis/var original: 12.65 %
---------------------------------
date: 06-Feb-2020
nobs = 7680,  ngood = 7679,  record length (days) = 32.00
start time: 15-Aug-2019
rayleigh criterion = 1.0
Greenwich phase computed, no nodal corrections

x0= 0.23, x trend= 0

var(x)= 0.28007   var(xp)= 0.24248   var(xres)= 0.035419
percent var predicted/var original= 86.6 %

     tidal amplitude and phase with 95% CI estimates

tide   freq       amp     amp_err    pha    pha_err     snr
 MM   0.0015122    0.1585    0.131    28.74    45.13      1.5
 MSF  0.0028219    0.0673    0.105   358.44    91.28     0.41
```

```
*O1   0.0387307    0.0597    0.010   203.42    11.17       33
*K1   0.0417807    0.0788    0.010   228.70     9.66       59
 MU2  0.0776895    0.0324    0.030    19.60    57.13      1.2
*N2   0.0789992    0.1866    0.033    15.36     8.28       32
*M2   0.0805114    0.6641    0.029    29.75     2.35  5.3e+02
*L2   0.0820236    0.0687    0.031    37.14    25.53      5.1
*S2   0.0833333    0.0972    0.028    51.56    17.51       12
*M4   0.1610228    0.0350    0.014   177.15    22.90      6.3
```

```
xout <- drop(readMat('ml_prediction_ten_consituents.mat')$XOUT)
oce.plot.ts(sl[['time']], predict(m)-m[['amplitude']][1] - sl[['elevation']],
            ylab=expression(Delta))
lines(sl[['time']], xout - sl[['elevation']], col=2)
```



Ok, so still not the same . . .

t_tide parameters synthesis

Reading through the `t_tide` documentation, there is an argument `synthesis`, which says:

```
 Computation of "predicted" tide (passed to t_predic, but note that
                                  the default value is different).
        'synthesis'      0 - use all selected constituents
                         scalar>0 - use only those constituents with a
                                    SNR greater than that given (1 or 2
                                    are good choices, 2 is the default).
```

```
                    <0 - return result of least-squares fit
                         (should be the same as using '0',
                         except that NaN-holes in original
                         time series will remain and mean/trend
                         are included).
```

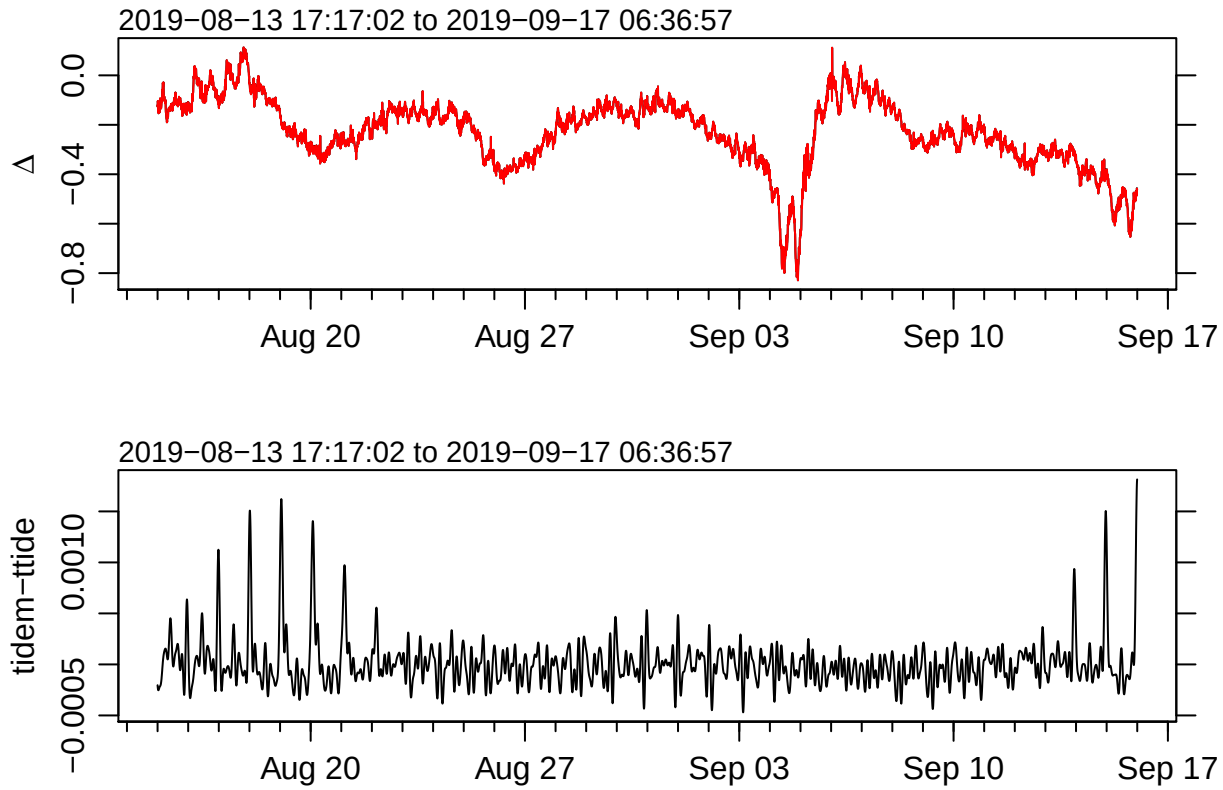
So, it appears that the default value is 2 – i.e. to only use constituents with an SNR > 2 to form the prediction. However, `tidem()` uses all fit constituents by default (I believe ... need to check on that).

```
[NAME,FREQ,TIDECON,XOUT]=t_tide(data.Verified_m_, 'interval', interval, 'start time', time(1) ...
    , 'synthesis', 0);
save('ml_prediction_all_consts.mat', 'XOUT')
```

```
xout <- drop(readMat('ml_prediction_all_consts.mat')$XOUT)
m <- tidem(sl)
```

```
## Note: the tidal record is too short to fit for constituents:  SA SSA MSM MF SIG1 RHO1 TAU1 BET1 CHI1
```

```
par(mfrow=c(2, 1))
oce.plot.ts(sl[['time']], predict(m)-m[['amplitude']][1] - sl[['elevation']],
            ylab=expression(Delta))
lines(sl[['time']], xout - sl[['elevation']], col=2)
oce.plot.ts(sl[['time']], predict(m)-m[['amplitude']][1] - xout,
            ylab='tidem-ttide')
```



The variance of the residuals is now much closer in the two approaches:

```
rmse(predict(m), sl[['elevation']])
```

```
## [1] 0.1381338
```

```r
rmse(xout, sl[['elevation']]-0.23)
```

```
## [1] 0.1381341
```