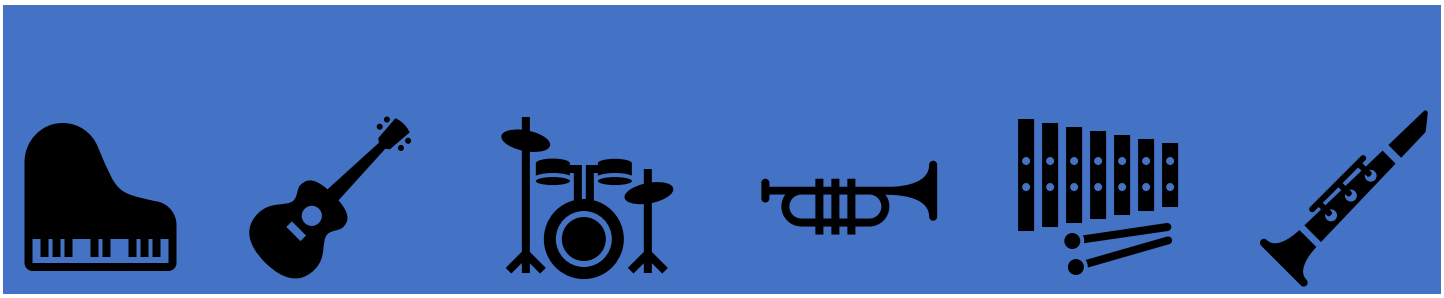
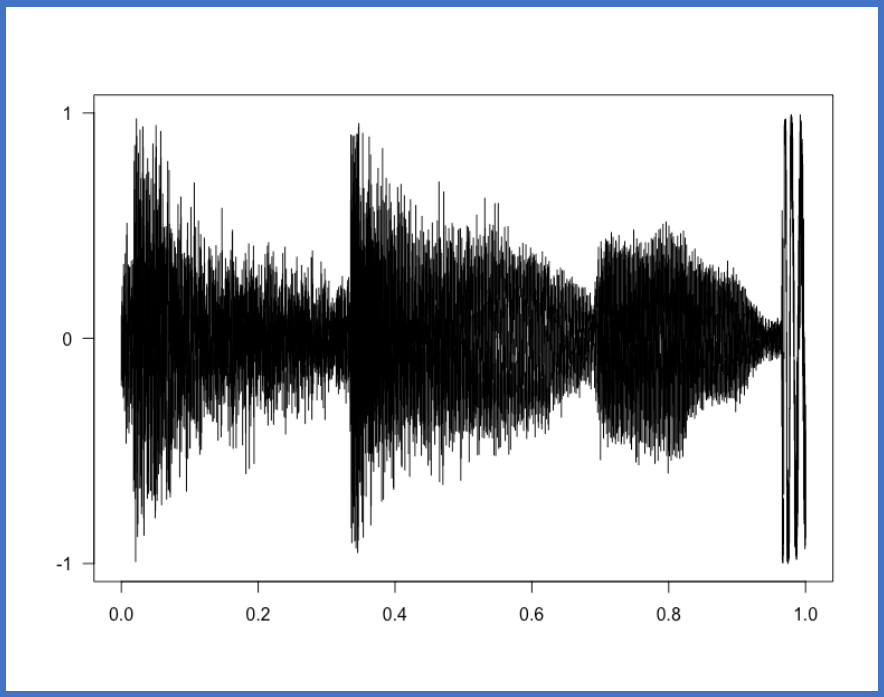


MODELING THE PERIODIC VARIATIONS IN MUSIC SIGNALS



Joey Gadbois

NONPARAMETRIC STATISTICS STAT 560

Introduction

Advancements in technology throughout the years has allowed for music to be played in numerous different ways and places. Music can be represented through audio signals such that it can be played on the radio, headphones, or anywhere that can output audio sounds. Audio signals represented in the time domain are a combination of any number of sinusoids of different frequencies to create a continuous line that has a very random looking appearance to it.

For audio to be analyzed it is sampled, where the standard sampling rate is 44.1 kHz per second in music but can be any rate above 40 kHz. The general method of analysis for audio signals is that of Fourier analysis since it is a very versatile topic in mathematics that is said to be able to consider any periodic function as a Fourier series. While Fourier analysis is well established in the analysis and synthesis of audio, there could be other methods that can be applied to gain insights on the audio representation of sounds and music.

Data

The data used for this study is a song by The Chainsmokers, called Closer, chosen because of its high degree of popularity and with the assumption that most people will have at least heard it before. While the song was chosen in hopes that people know the song, this project is only illustrated on 50 milliseconds of the song. A 50-millisecond subsample of audio consists of 2,205 out of 44,100 samples. Then, the 2,205 is taken at every fifth point resulting in 441 data points to be modeled. The full 2,205 will be used in prediction.

Methodology

Attempting to model musical audio signals, nonparametric regression methods including Loess regression and Thin-Plate smoothing splines will be implemented due to their abilities to estimate periodic functions with a relationship that is impossible to determine parametrically. They will be fitted on 441 samples of audio mentioned above at various settings to evaluate performance. The performance of these methods will be evaluated by making predictions of the full 2,205 samples and observing how well the real signal can be modeled by nonparametric regression.

Loess regression is implemented linearly and quadratically with smoothing parameters of 0.01, 0.02, 0.03, and 0.04. AICC optimization does not work with this data due to it searching for a very smooth curve, while the signal data varies from one extreme to another very frequently. Thin-Plate splines will be fitted at the degrees of smoothness of 2, 3, and 4.

Results

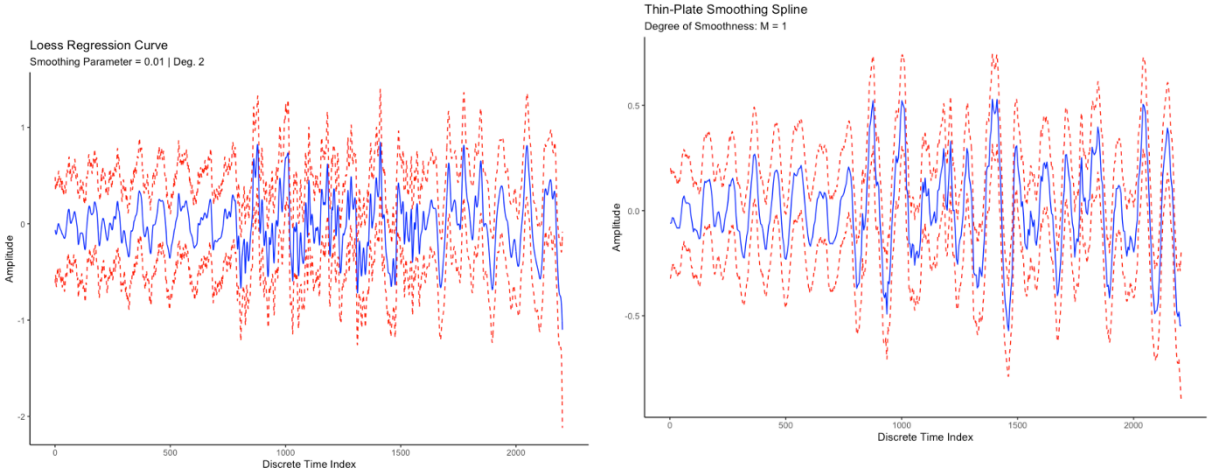
For loess regression, the first degree estimated curves produced decent results. The smoothing parameter that yielded the most accurate representation of the signal is that of 0.01. For the smoothing parameters 0.02, 0.03, and 0.04, the curves appeared to be too smooth for the given data.

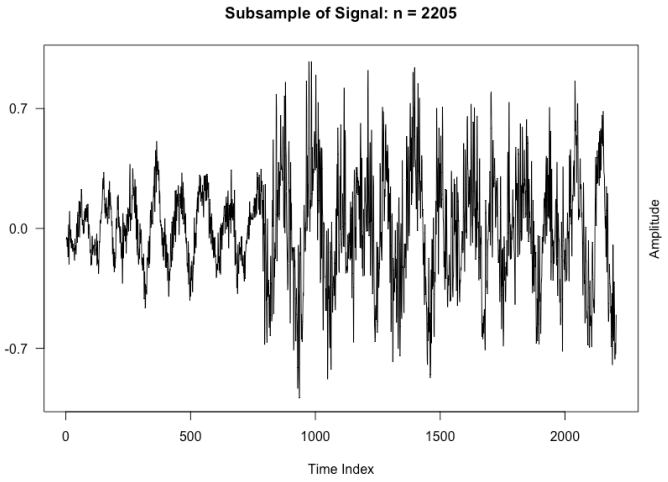
For second degree estimates, the estimated curves gave a better representation overall with the various smoothing parameters. Depending on preference, the optimal smoothing

parameter setting is 0.01 or 0.02. Smoothing parameter option 0.01 produces a more variant curve, but the confidence limits are much larger.

For thin-plate splines, the curve produced from $m = 1$ degree of smoothness yields the most accurate results. Using $m = 2, 3$ as the degree of smoothness performs better than the loess options that are not selected as optimal but given the data the estimated curves are too smooth.

The graphs of an optimal selection loess regression and thin-plate splines is shown below along with a graph of the actual signal.





Conclusion

In conclusion, audio signals are very complex sinusoidal functions with unexpected behaviors at times. The second-degree loess curve with a selected smoothing parameter seemed to perform the best in capturing the periodic variations in the signal. It also provided larger confidence limits which often times isn't ideal, but for this specific type of data is preferred because it ensures that most of the sample points are accounted for, giving a better opportunity to capture large points of amplitude in the signal.

Appendix

- **Section 1:** R Codes
- **Section 2:** SAS Codes
- **Section 3:** Output graphs

Section 1: R Codes

Functions:

```
# load libraries
load_libraries = function(){
 library(tuneR)
 library(tidyverse)
 library(broom)
 library(ggpubr)
 library(fANCOVA)
 library(fields)
}

# audio functions
signal_load_process = function(path, start, stop){
 # loads signal from provided path of an audio file
 # start is starting point of song in seconds
 # stop is ending point inn seconds
 # converts stereo signal to mono and normalizes it
 audio = readWave(path, from = start, to = stop, units = 'seconds')
 ymono = mono(audio, which = 'both')
 ymono = normalize(ymono, unit = '1')
 str(ymono)
 return(ymono)
}
signal_subsamples = function(y, samp_length){
 # extracts signal array and splits it evenly by
 # the length you provide
 s = y@left
 ysplit = seq_along(s)
 samples = split(s, ceiling(ysplit/samp_length))
 return(samples)
}

# data functions
data_signal = function(y){
 # creates dataset from signal consisting of
 # y = sampled signal, n = discrete time index
 df = tibble(n=1:length(y), y=y)
}
data_every_n = function(n, y){
 # creates new dataset that takes every nth
 # data point in the signal dataset
 n1 = n[seq(1, length(n), 5)]
 y1 = y[seq(1, length(y), 5)]
 df = tibble(n=n1, y=y1)
 df %>% print()
 return(df)
```

```
}

# functions to create plotting data
df_preds_loess = function(fit, df_pred, conf_lev){
 # create dataset to plot the curve including values for
 # predictions, confidence bands, standard errors, original signal
 # fit: fitted curve
 # data: df for predictions
 # conf_lev: confidence level for intervals
 alpha = 1-conf_lev
 confidence = 1 - (alpha/2)
 crit_val = qnorm(p = confidence)
 n = df_pred$n; y = df_pred$y
 pse = predict(fit, df_pred, se = T)
 p = pse$fit
 s = pse$se.fit
 lb = p - crit_val * s
 ub = p + crit_val * s
 df = data.frame(y=y, n=n, pred=p, se=s, low_cb=lb, upp_cb=ub)
 df = as_tibble(df)
 return(df)
}
df_preds_spline = function(fit, df_pred, conf_lev){
 # create dataset to plot the spline including values for
 # predictions, confidence bands, standard errors, original signal
 # fit: fitted curve
 # data: df for predictions
 # conf_lev: confidence level for intervals
 alpha = 1-conf_lev
 confidence = 1 - (alpha/2)
 crit_val = qnorm(p = confidence)

 n = df_pred$n; y = df_pred$y
 p = predict(fit, x = n)
 s = predictSE(fit, x = n)
 lb = p - crit_val * s
 ub = p + crit_val * s

 df = data.frame(y=y, n=n, pred=p, se=s, low_cb=lb, upp_cb=ub)
 df = as_tibble(df)
 return(df)
}
curve_plot = function(df, type = c('loess', 'tp'), sub = ''){
 if (type == 'loess'){
  title = 'Loess Regression Curve'
 }else{
  title = 'Thin-Plate Smoothing Spline'
 }
 df %>% ggplot(aes(x=n)) + geom_point(aes(y=y), color='cyan') +
  geom_line(aes(y=pred), color='red') +
  geom_line(aes(y=low_cb), color='red', lty=2) +
  geom_line(aes(y=upp_cb), color='red', lty=2) +
  ggtitle(title, subtitle = sub) + theme_classic() +
  labs(x='Discrete Time Index', y='Amplitude')
}
```

```
curve_only_plot = function(df, type = c('loess', 'tp'), sub = ''){
 if (type == 'loess'){
  title = 'Loess Regression Curve'
 }else{
  title = 'Thin-Plate Smoothing Spline'
 }
 df %>% ggplot(aes(x=n)) +
  geom_line(aes(y=pred), color='blue') +
  geom_line(aes(y=low_cb), color='red', lty=2) +
  geom_line(aes(y=upp_cb), color='red', lty=2) +
  ggtitle(title, subtitle = sub) + theme_classic() +
  labs(x='Discrete Time Index', y='Amplitude')
}
```

Loess Regression

```
# Loess Regression Degree 1
sm1 = list()
dl1 = list()
p1 = list()
subs1 = list('Smoothing Parameter = 0.01 | Deg. 1',
         'Smoothing Parameter = 0.02 | Deg. 1',
         'Smoothing Parameter = 0.03 | Deg. 1',
         'Smoothing Parameter = 0.04 | Deg. 1')

for (i in 1:4){
 span = i/100
 s = loess(y ~ n, data = xdf, span = span, degree = 1, family = 'gaussian')
 sm1 = append(sm1, list(s))
 d = df_preds_loess(s, df_pred = xdf_samps, conf_lev = 0.95)
 dl1 = append(dl1, list(d))
 p = curve_plot(d, type = 'loess', sub = subs1[[i]])
 p1 = append(p1, list(p))
}

ggarrange(p1[[1]], p1[[2]], p1[[3]], p1[[4]], nrow = 2, ncol = 2)

# Loess Regression Degree 2
sm2 = list()
dl2 = list()
p2 = list()
subs2 = list('Smoothing Parameter = 0.01 | Deg. 2',
         'Smoothing Parameter = 0.02 | Deg. 2',
         'Smoothing Parameter = 0.03 | Deg. 2',
         'Smoothing Parameter = 0.04 | Deg. 2')

for (i in 1:4){
 span = i/100
 s = loess(y ~ n, data = xdf, span = span, degree = 2, family = 'gaussian')
 sm2 = append(sm2, list(s))
 d = df_preds_loess(s, df_pred = xdf_samps, conf_lev = 0.95)
 dl2 = append(dl2, list(d))
 p = curve_plot(d, type = 'loess', sub = subs2[[i]])
 p2 = append(p2, list(p))
}
```

```
ggarrange(p2[[1]], p2[[2]], p2[[3]], p2[[4]], nrow = 2, ncol = 2)
```

Thin-Plate Smoothing:

```
# Thin-Plate Splines
attach(xdf)
tpm = list()
dtp = list()
p3 = list()
subs3 = list('Degree of Smoothness: M = 1',
        'Degree of Smoothness: M = 2',
        'Degree of Smoothness: M = 3',
        'Degree of Smoothness: M = 4')
for (i in 1:4){
 m = Tps(x=n, Y=y, m=(i))
 tpm = append(tpm, list(m))
 d = df_preds_spline(m, xdf_samps, 0.95)
 dtp = append(dtp, list(d))
 p = curve_plot(dtp[[i]], type = 'tp', sub = subs3[[i]])
 p3 = append(p3, list(p))
}

ggarrange(p3[[1]], p3[[2]], p3[[3]], p3[[4]], nrow = 2, ncol = 2)
```

Section 2: SAS Codes

Loess Regression

```
symbol1 color=blue value=dot;
symbol2 color=red value=none interpol=join line=1;
symbol3 color=red value=none interpol=join line=2;
symbol4 color=red value=none interpol=join line=2;

proc loess data=df;
  model y = n / degree=1 clm smooth=0.01;
  ods output OutputStatistics=result1;
  score data=dfp;
run;

title 'Linear Loess | SP = 0.01';
proc gplot data=result1;
  by SmoothingParameter;
  plot (DepVar Pred LowerCL UpperCL)*n/ overlay name='pL1';
run;

proc loess data=df;
  model y = n / degree=1 clm smooth=0.02;
  ods output OutputStatistics=result2;
  score data=dfp;
run;

title 'Linear Loess | SP = 0.02';
proc gplot data=result2;
  by SmoothingParameter;
```

```
  plot (DepVar Pred LowerCL UpperCL)*n/ overlay name='pL2';
run;

proc loess data=df;
  model y = n / degree=1 clm smooth=0.03;
  ods output OutputStatistics=result3;
  score data=dfp;
run;

title 'Linear Loess | SP = 0.03';
proc gplot data=result3;
  by SmoothingParameter;
  plot (DepVar Pred LowerCL UpperCL)*n/ overlay name='pL3';
run;

proc loess data=df;
  model y = n / degree=1 clm smooth=0.04;
  ods output OutputStatistics=result4;
  score data=dfp;
run;

title 'Linear Loess | SP = 0.04';
proc gplot data=result4;
  by SmoothingParameter;
  plot (DepVar Pred LowerCL UpperCL)*n/ overlay name='pL4';
run;

goptions display;
proc greplay nofs tc=sashelp.templt template=l2r2;
  igout gseg;
  treplay 1:pL1 2:pL2 3:pL3 4:pL4;
run;

proc loess data=df;
  model y = n / degree=2 clm smooth=0.01;
  ods output OutputStatistics=result11;
  score data=dfp;
run;

title 'Linear Loess | SP = 0.01';
proc gplot data=result11;
  by SmoothingParameter;
  plot (DepVar Pred LowerCL UpperCL)*n/ overlay name='pL11';
run;

proc loess data=df;
  model y = n / degree=2 clm smooth=0.02;
  ods output OutputStatistics=result22;
  score data=dfp;
run;

title 'Linear Loess | SP = 0.02';
proc gplot data=result22;
  by SmoothingParameter;
  plot (DepVar Pred LowerCL UpperCL)*n/ overlay name='pL22';
run;
```

```
proc loess data=df;
  model y = n / degree=2 clm smooth=0.03;
  ods output OutputStatistics=result33;
  score data=dfp;
run;

title 'Linear Loess | SP = 0.03';
proc gplot data=result33;
  by SmoothingParameter;
  plot (DepVar Pred LowerCL UpperCL)*n/ overlay name='pL33';
run;

proc loess data=df;
  model y = n / degree=2 clm smooth=0.04;
  ods output OutputStatistics=result44;
  score data=dfp;
run;

title 'Linear Loess | SP = 0.04';
proc gplot data=result44;
  by SmoothingParameter;
  plot (DepVar Pred LowerCL UpperCL)*n/ overlay name='pL44';
run;

goptions display;
proc greplay nofs tc=sashelp.templt template=l2r2;
  igout gseg;
  treplay 1:pL11 2:pL22 3:pL33 4:pL44;
run;
```

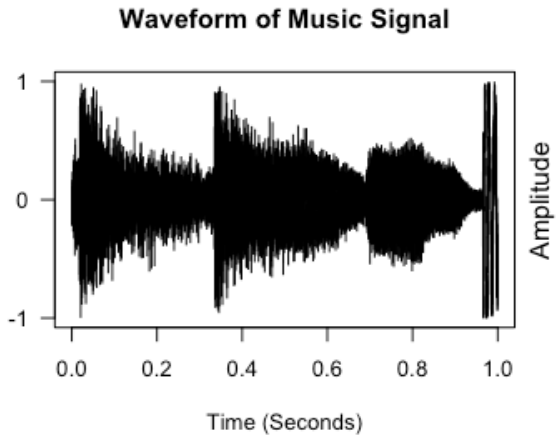
Thin-Plate Smoothing:

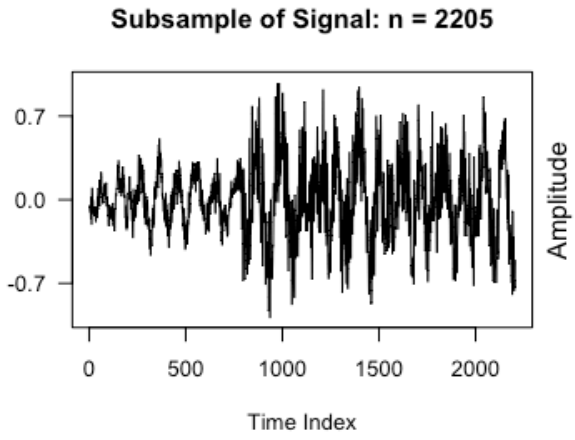
```
proc tpspline data=df;
  model y = (n) / m=1;
  output out=pm1 pred lclm uclm;
run;
title 'm=1';
proc gplot data=pm2;
  plot (y P_y LCLM_y UCLM_y)*n/ overlay name='pm1';
run;

proc tpspline data=df;
  model y = (n) / m=2;
  output out=pm2 pred lclm uclm;
  score data=dfp out=pm2res;
run;
title 'm=2';
proc gplot data=pm2res;
  plot (y P_y LCLM_y UCLM_y)*n/ overlay name='plotm2';
run;

proc tpspline data=df;
  model y = (n) / m=3;
  output out=pm3 pred lclm uclm;
run;
title 'm=3';
proc gplot data=pm3;
```

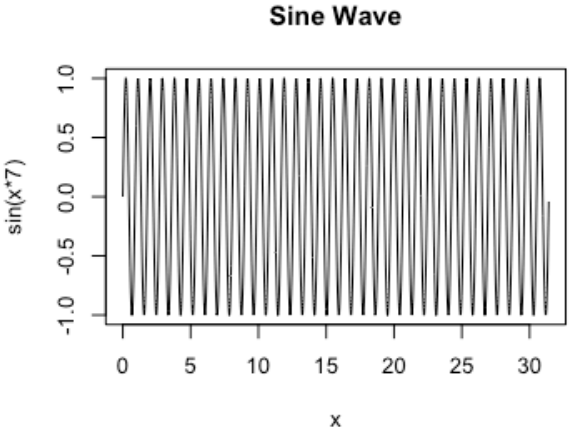
```
  plot (y P_y LCLM_y UCLM_y)*n/ overlay name='pm3';
run;

goption display;
proc greplay nofs tc=sashelp.templt template=l2r2;
igout gseg;
treplay 1:pm1 2:pm2 3:pm3;
run;
```

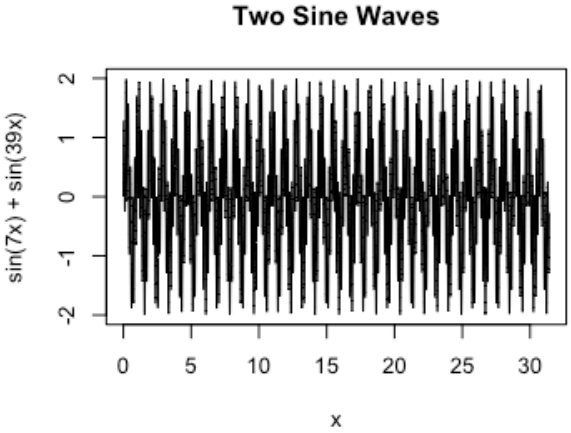
Section 3: Output Graphs

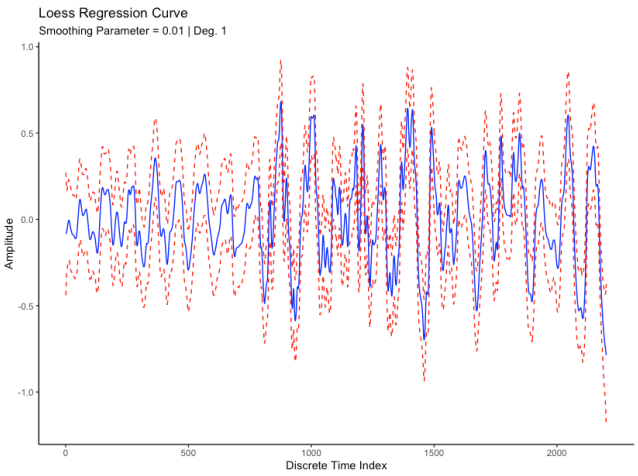
Graphs in R:

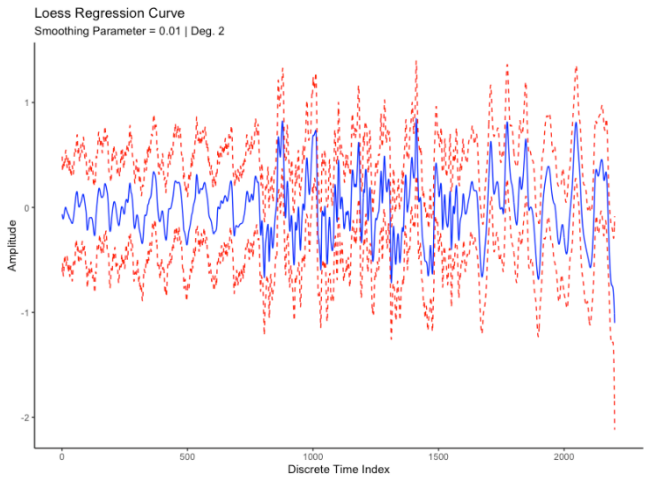


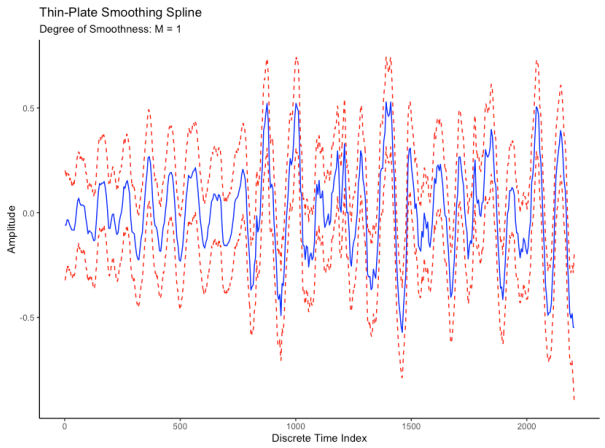


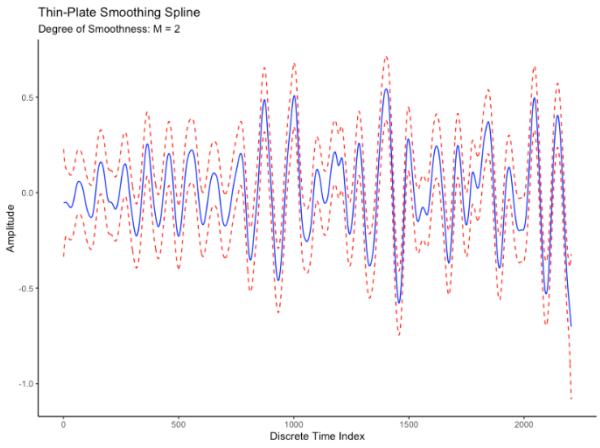


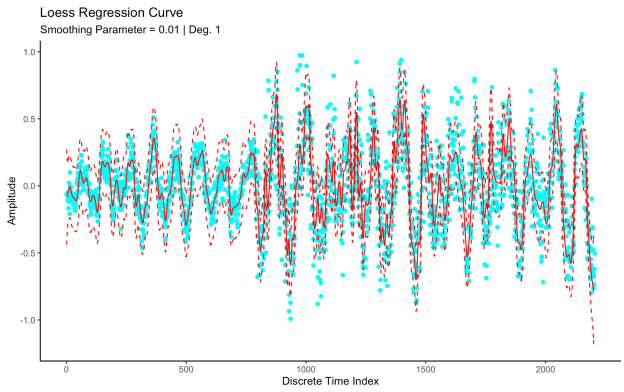


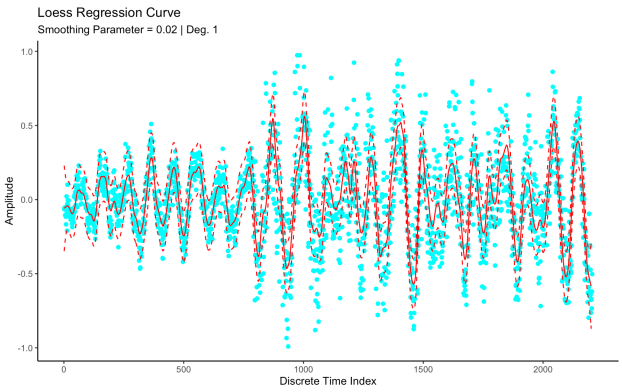


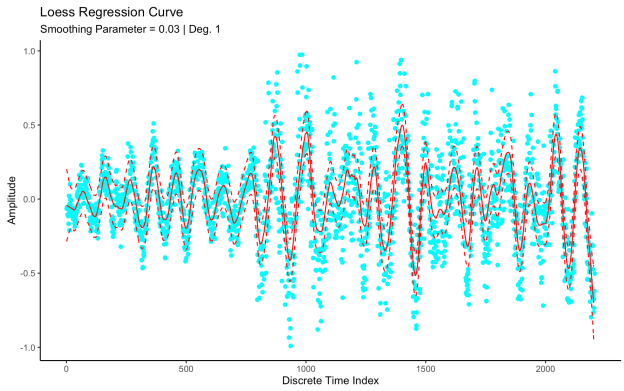


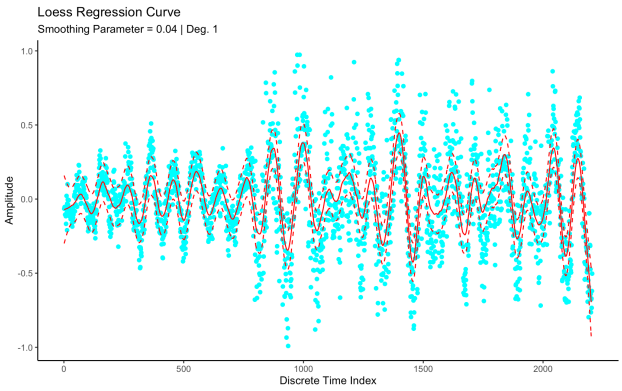

Thin-Plate Smoothing Spline
Degree of Smoothness: M = 1
Amplitude
Discrete Time Index

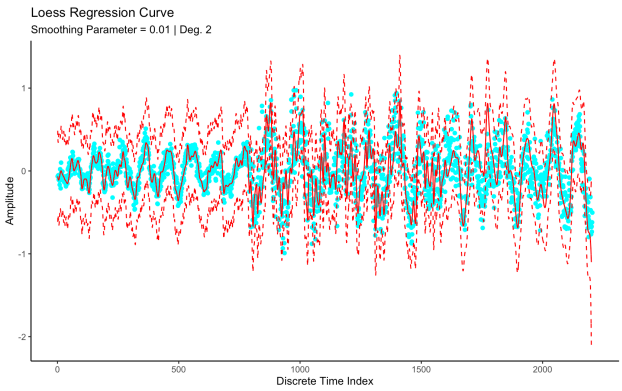

Thin-Plate Smoothing Spline
Degree of Smoothness: M = 2
Amplitude
Discrete Time Index

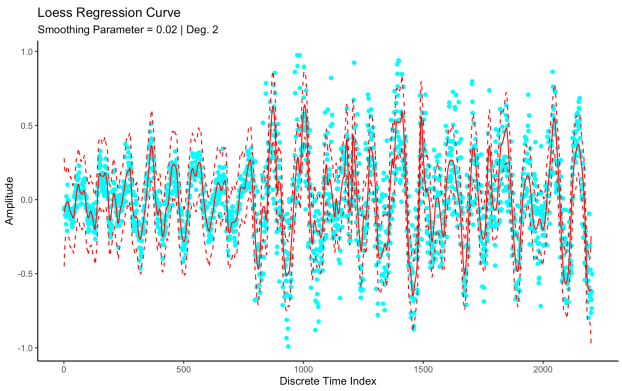

Loess Regression Curve
Smoothing Parameter = 0.01 | Deg. 1
Amplitude
Discrete Time Index

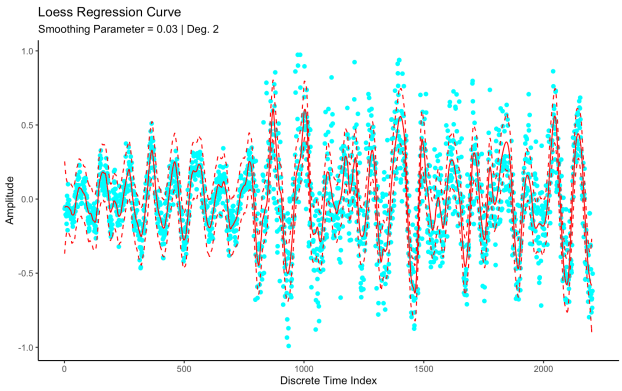

Loess Regression Curve
Smoothing Parameter = 0.02 | Deg. 1
Amplitude
Discrete Time Index

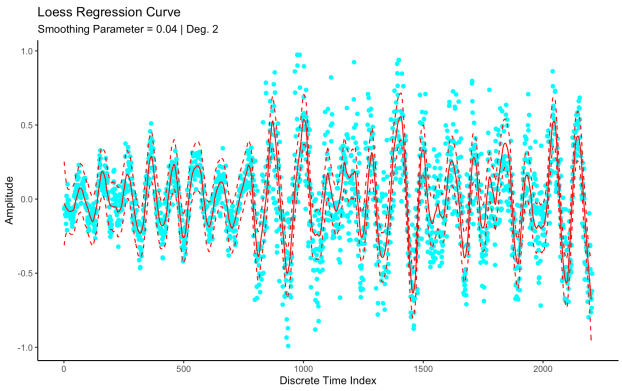

Loess Regression Curve
Smoothing Parameter = 0.03 | Deg. 1
Amplitude
Discrete Time Index

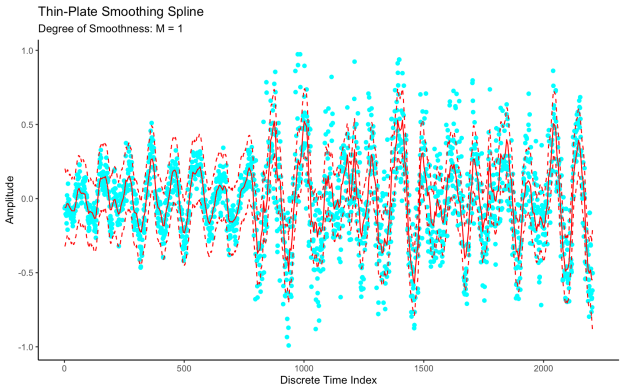

Loess Regression Curve
Smoothing Parameter = 0.04 | Deg. 1
Amplitude
Discrete Time Index

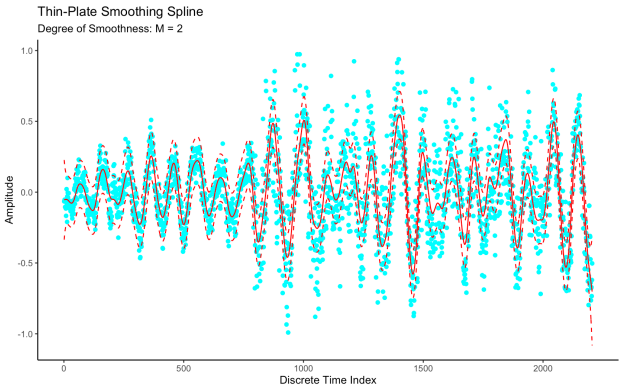

Loess Regression Curve
Smoothing Parameter = 0.01 | Deg. 2
Amplitude
Discrete Time Index

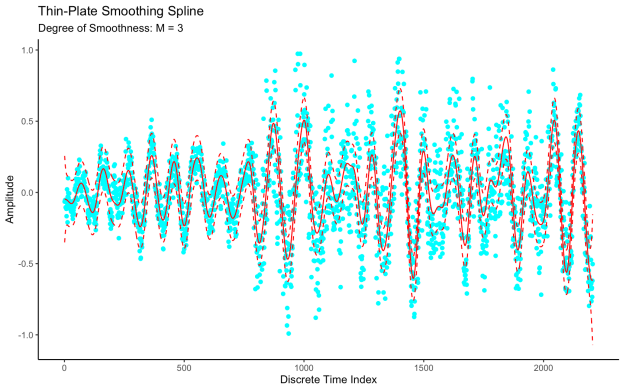

Loess Regression Curve
Smoothing Parameter = 0.02 | Deg. 2
Amplitude
Discrete Time Index

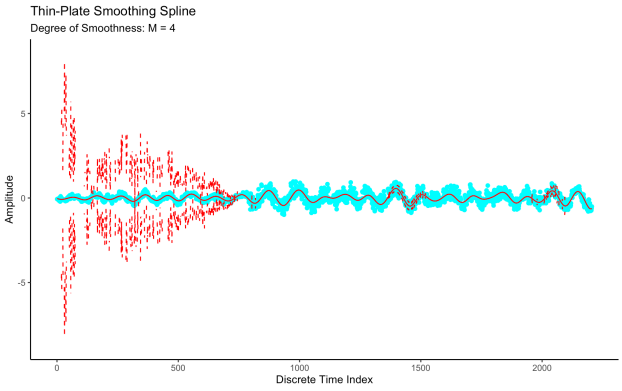

Loess Regression Curve
Smoothing Parameter = 0.03 | Deg. 2
Amplitude
Discrete Time Index


Loess Regression Curve
Smoothing Parameter = 0.04 | Deg. 2
Amplitude
Discrete Time Index

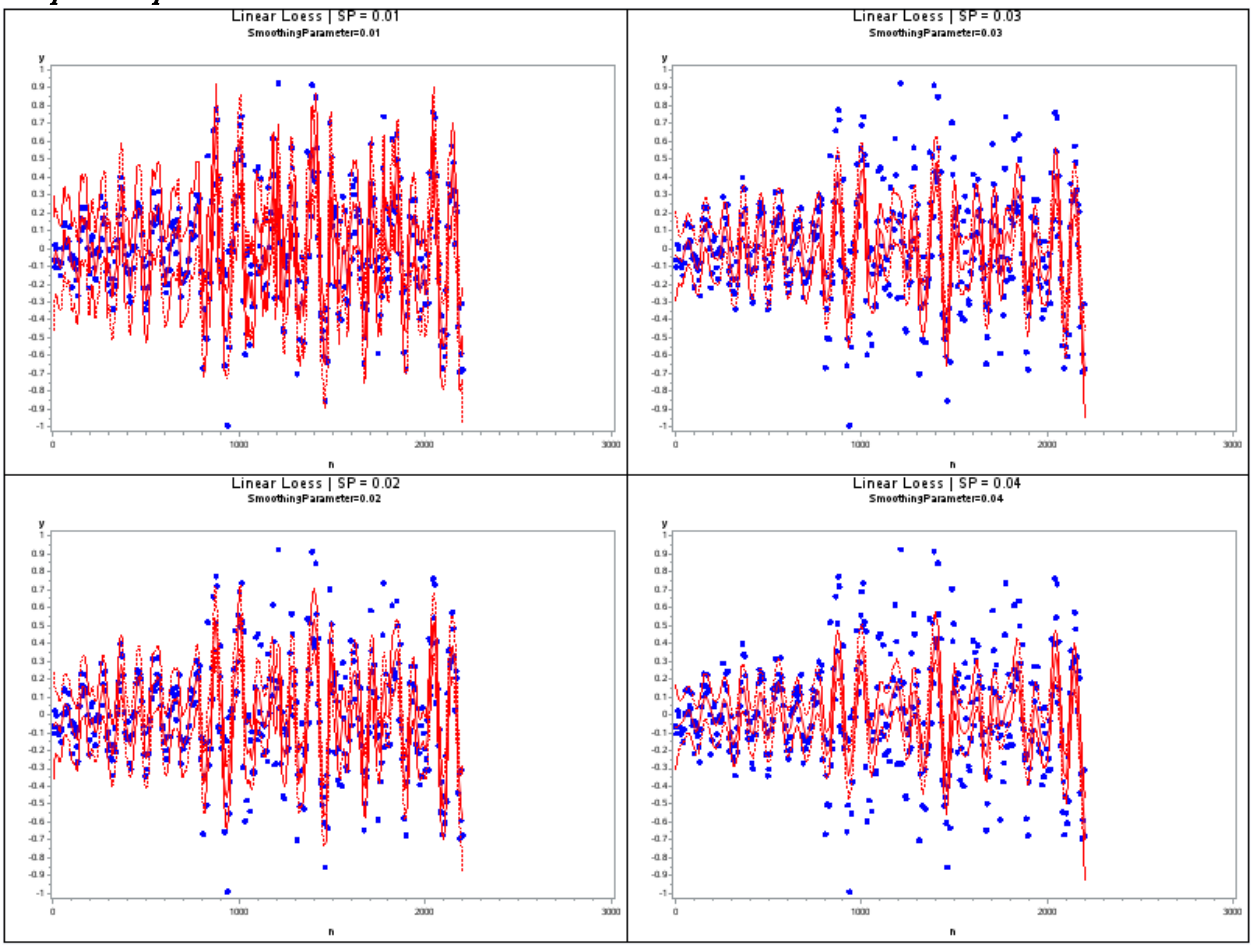

Thin-Plate Smoothing Spline
Degree of Smoothness: M = 1
Amplitude
Discrete Time Index

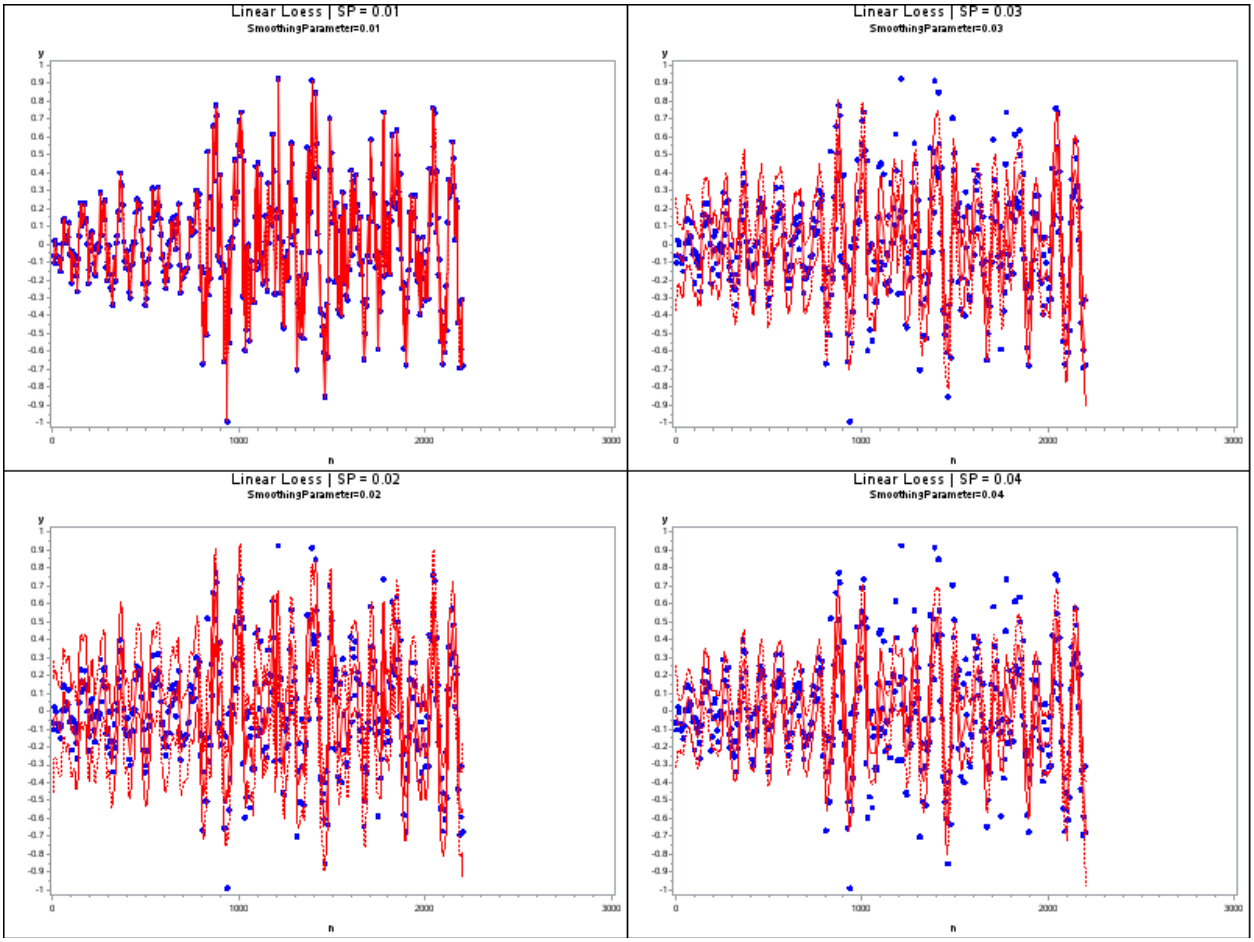

Thin-Plate Smoothing Spline
Degree of Smoothness: M = 2
Amplitude
Discrete Time Index

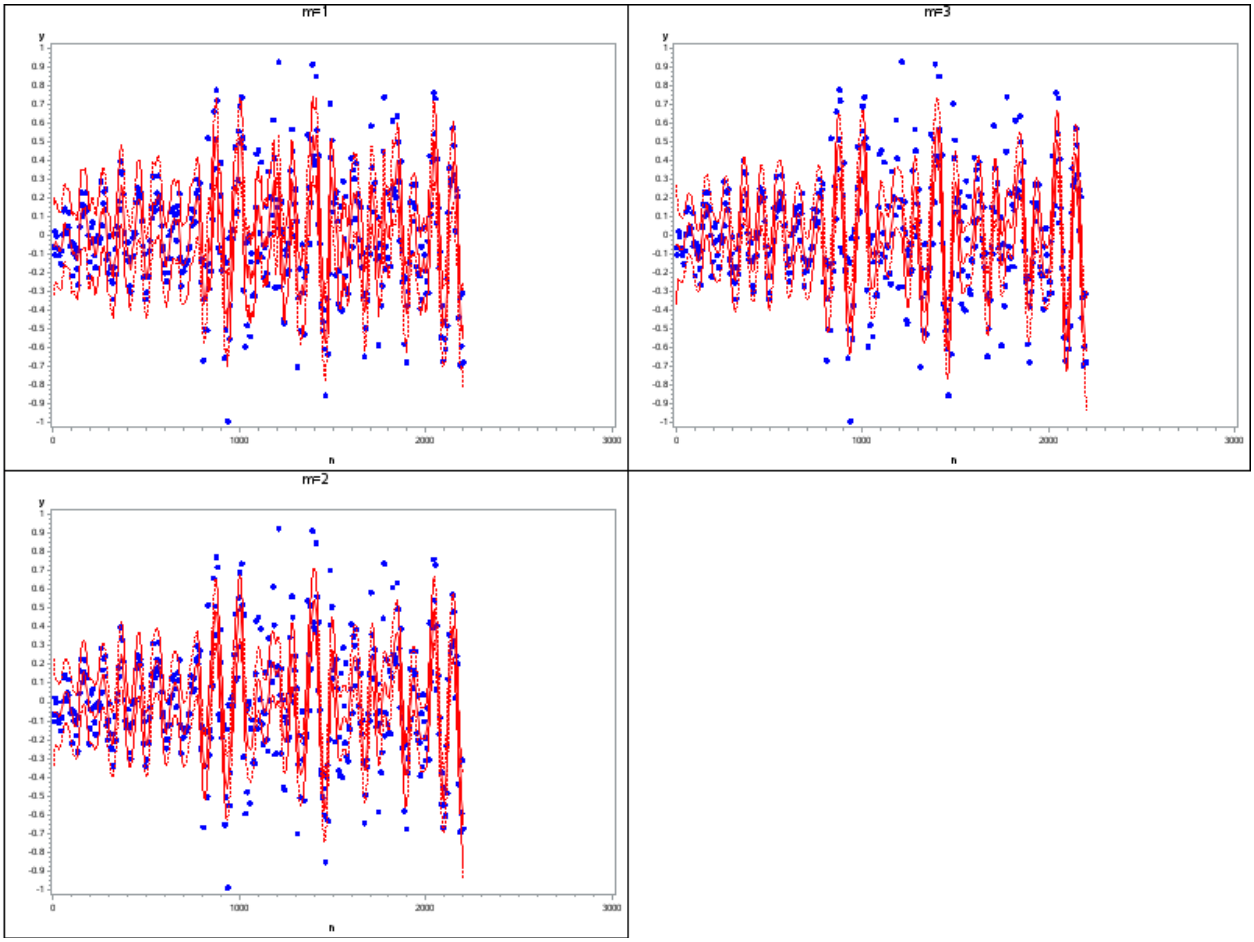

Thin-Plate Smoothing Spline
Degree of Smoothness: M = 3
Amplitude
Discrete Time Index


Thin-Plate Smoothing Spline
Degree of Smoothness: M = 4
Amplitude
Discrete Time Index

Output Graphs from SAS:




Linear Loess | SP = 0.01
SmoothingParameter=0.01
Linear Loess | SP = 0.03
SmoothingParameter=0.03
Linear Loess | SP = 0.02
SmoothingParameter=0.02
Linear Loess | SP = 0.04
SmoothingParameter=0.04
y
n


m=1
m=3
m=2
y
n
0
1000
2000
3000