Nonparametric Audio Signal Modeling

Joey Gadbois (016228847)

Set Up

Functions

```
load_libraries = function(){
  library(tuneR)
  library(tidyverse)
  library(broom)
  library(ggpubr)
  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)]
```

```r
  y1 = y[seq(1, length(y), 5)]
  df = tibble(n=n1, y=y1)
  df %>% print()
  return(df)
}

# plotting predictions and confidence bands
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')
}
```

Load Libraries and Audio

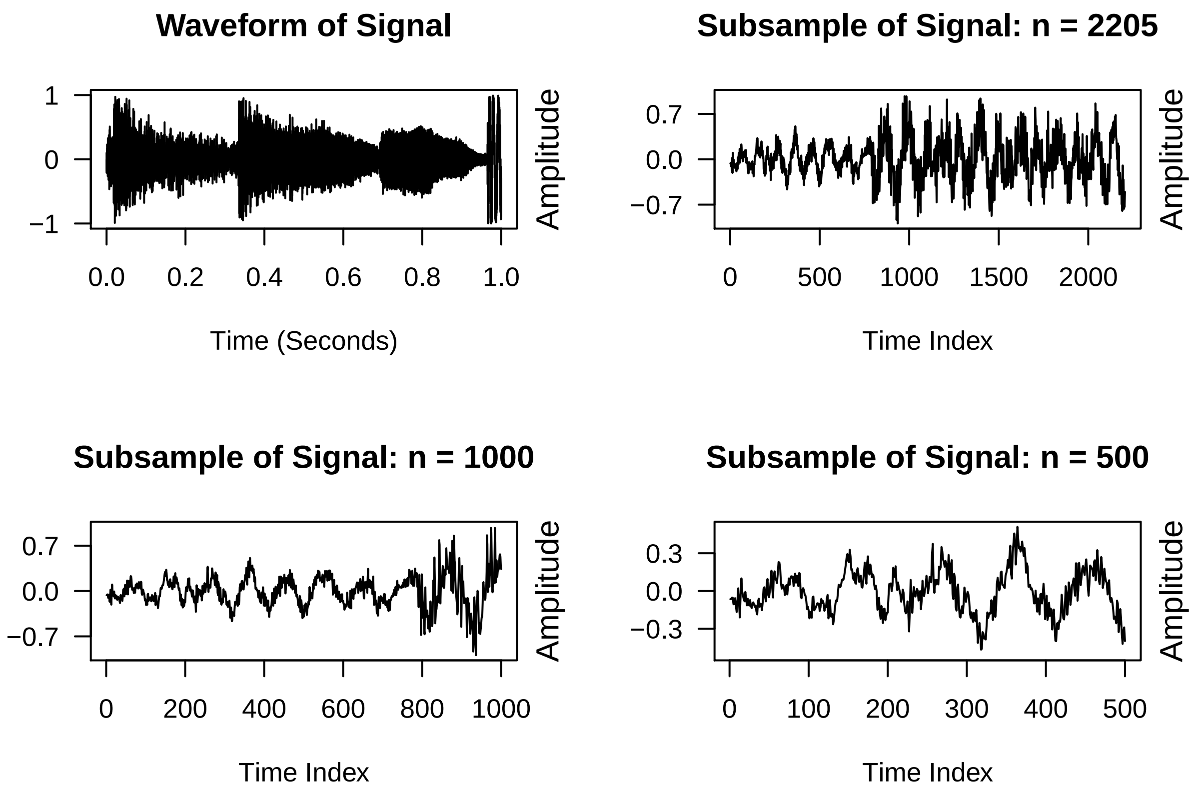
```
load_libraries()
xn = signal_load_process('Closer.wav', start = 70, stop = 71)
```

```
## Formal class 'Wave' [package "tuneR"] with 6 slots
##   ..@ left     : num [1:44100] -0.0648 -0.0642 -0.0533 -0.0532 -0.1051 ...
##   ..@ right    : num(0)
##   ..@ stereo   : logi FALSE
##   ..@ samp.rate: int 44100
##   ..@ bit      : num 32
##   ..@ pcm      : logi TRUE
```

Explore and Create Data

Explore Audio

```
par(mfrow=c(2,2))
plot(xn, main='Waveform of Signal', xlab='Time (Seconds)', ylab='Amplitude')
plot(extractWave(xn, from = 1, to = 2205), main = 'Subsample of Signal: n = 2205',
     xunit = 'samples', xlab = 'Time Index', ylab = 'Amplitude')
plot(extractWave(xn, from = 1, to = 1000), main = 'Subsample of Signal: n = 1000',
     xunit = 'samples', xlab = 'Time Index', ylab = 'Amplitude')
plot(extractWave(xn, from = 1, to = 500), main = 'Subsample of Signal: n = 500',
     xunit = 'samples', xlab = 'Time Index', ylab = 'Amplitude')
```




```r
par(mfrow=c(1,1))
```

Subsamples of Signal

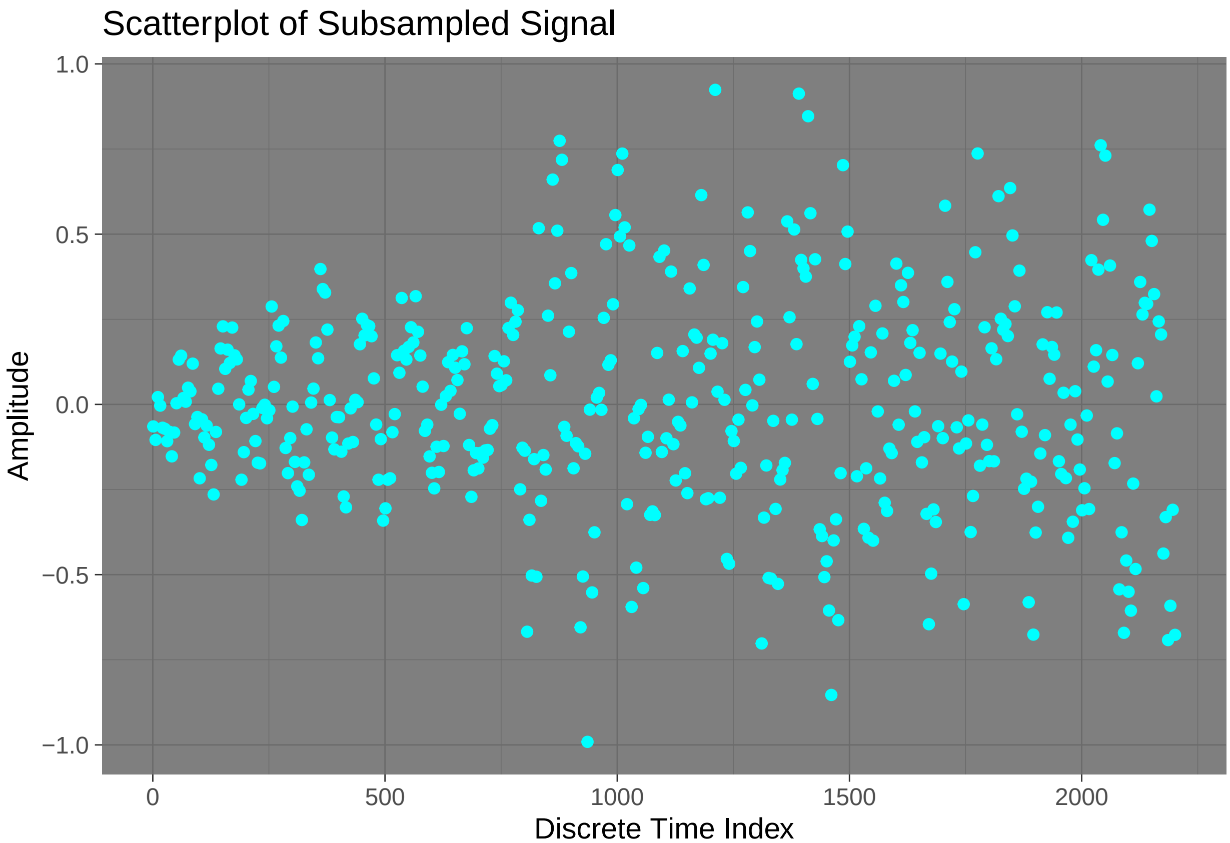
```r
xsamps = signal_subsamples(xn, 2205)
xdf_samps = data_signal(xsamps[[1]])
xdf = data_every_n(xdf_samps$n, xdf_samps$y)
```

```
## # A tibble: 441 x 2
##         n        y
##    <int>    <dbl>
##  1     1 -0.0648
##  2     6 -0.104
##  3    11  0.0215
##  4    16 -0.00360
##  5    21 -0.0685
##  6    26 -0.0716
##  7    31 -0.108
##  8    36 -0.0806
##  9    41 -0.153
## 10    46 -0.0826
## # ... with 431 more rows
```

Scatterplot

```r
xdf %>% ggplot(aes(x=n, y=y)) + geom_point(color='cyan') +
  ggtitle('Scatterplot of Subsampled Signal') +
  labs(x='Discrete Time Index', y='Amplitude') +
```

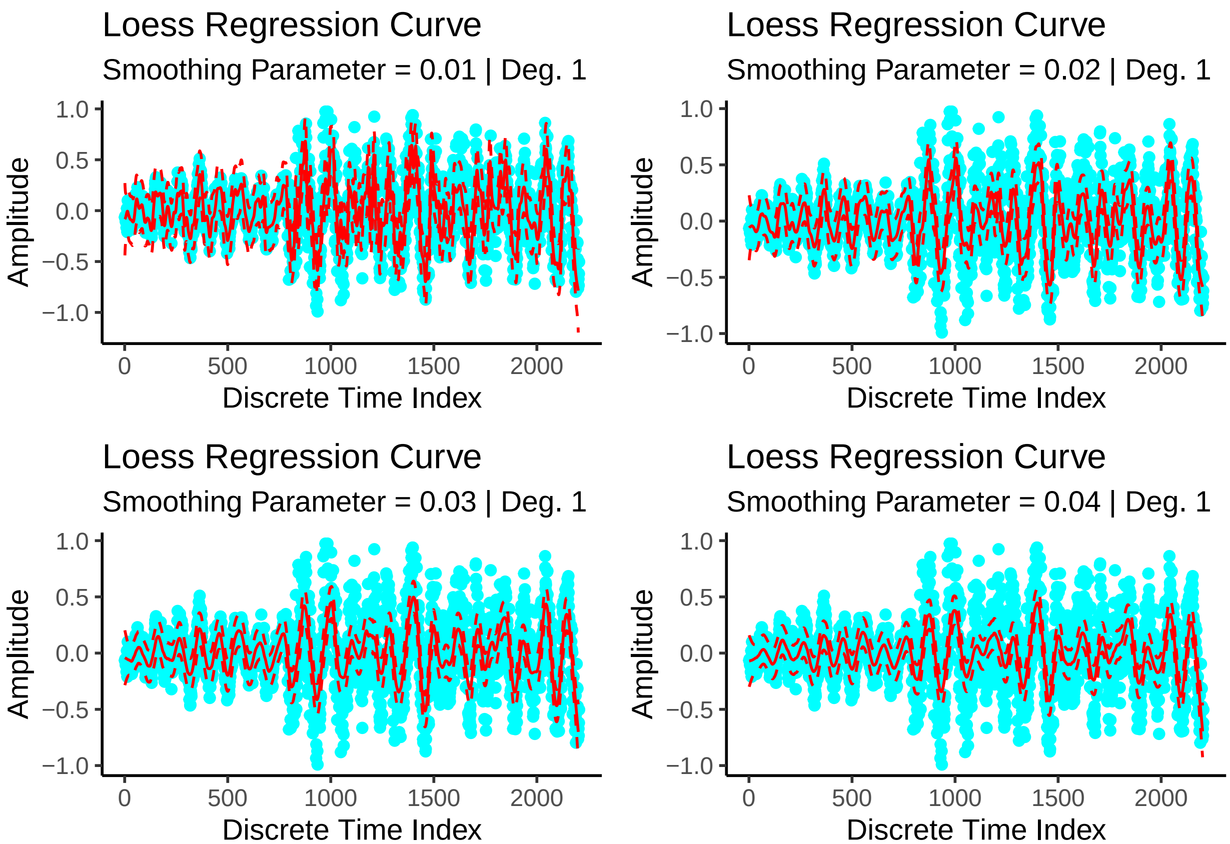
```
  theme_dark()
```



Loess Regression

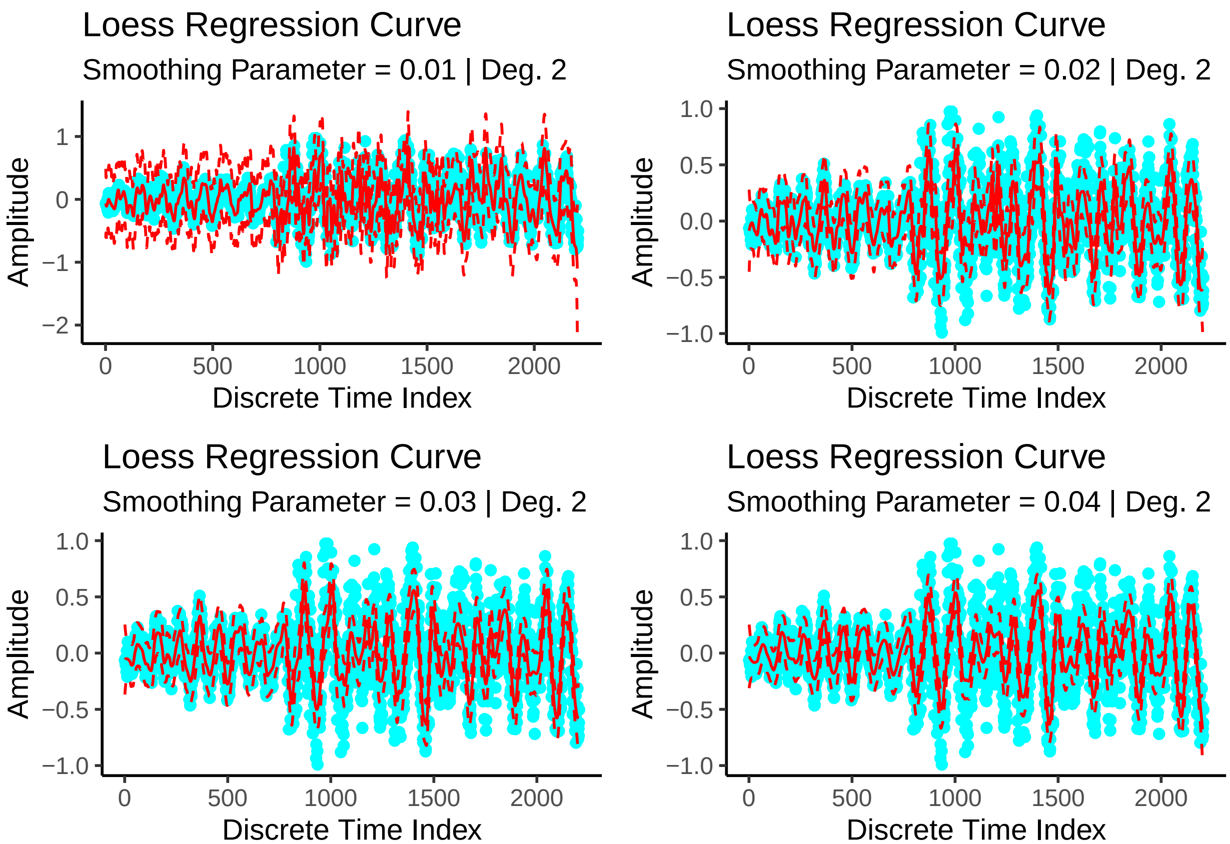
Degree 1

```
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)
```



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)
```

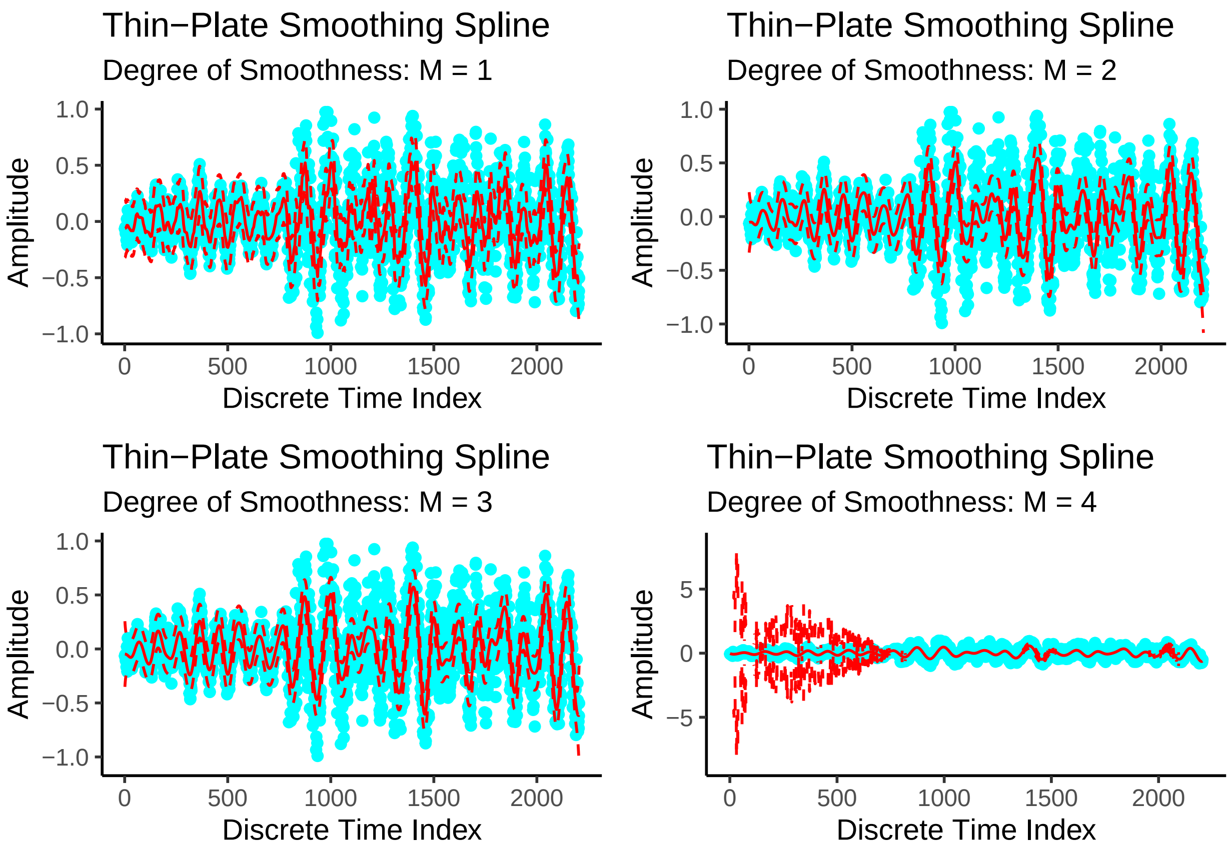


Thin-Plate Splines

```r
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))
}
```

```
## Warning:
## Grid searches over lambda (nugget and sill variances) with  minima at the endpoints:
##   (GCV) Generalized Cross-Validation
##    minimum at  right endpoint  lambda  =  8.881789e-16 (eff. df= 55.30496 )
```

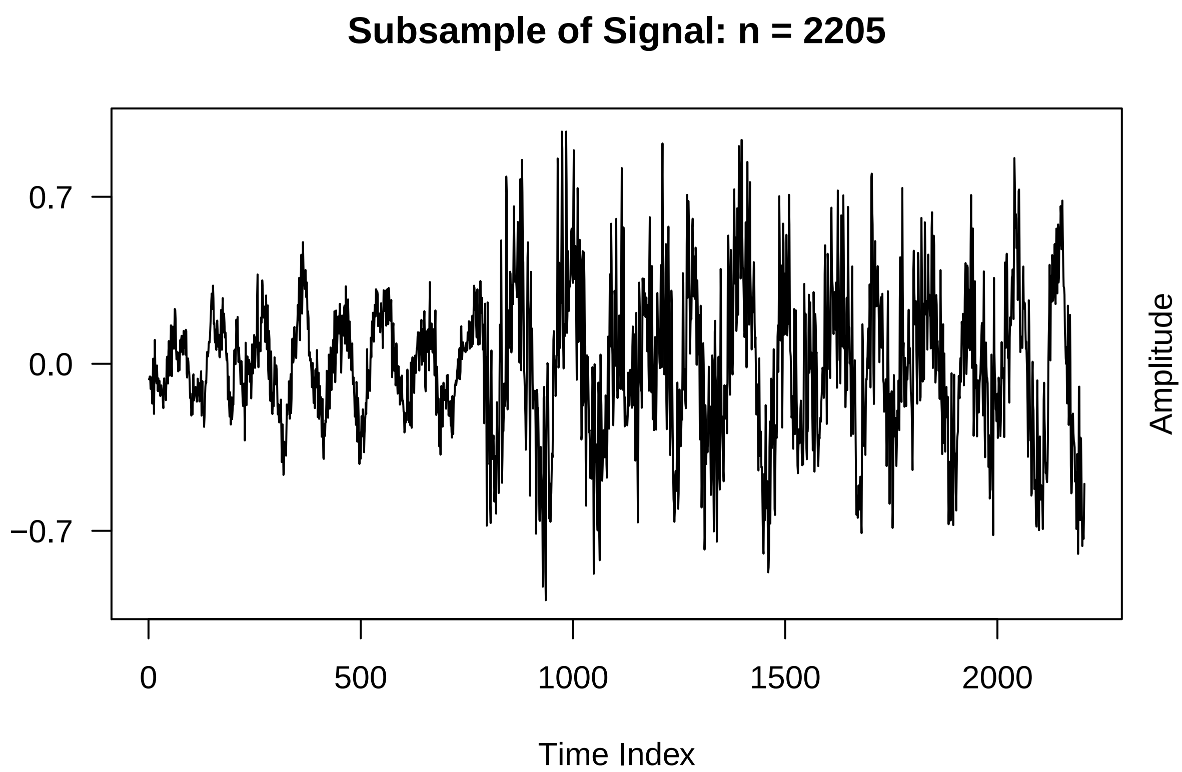
```r
ggarrange(p3[[1]], p3[[2]], p3[[3]], p3[[4]], nrow = 2, ncol = 2)
```



Comparisons

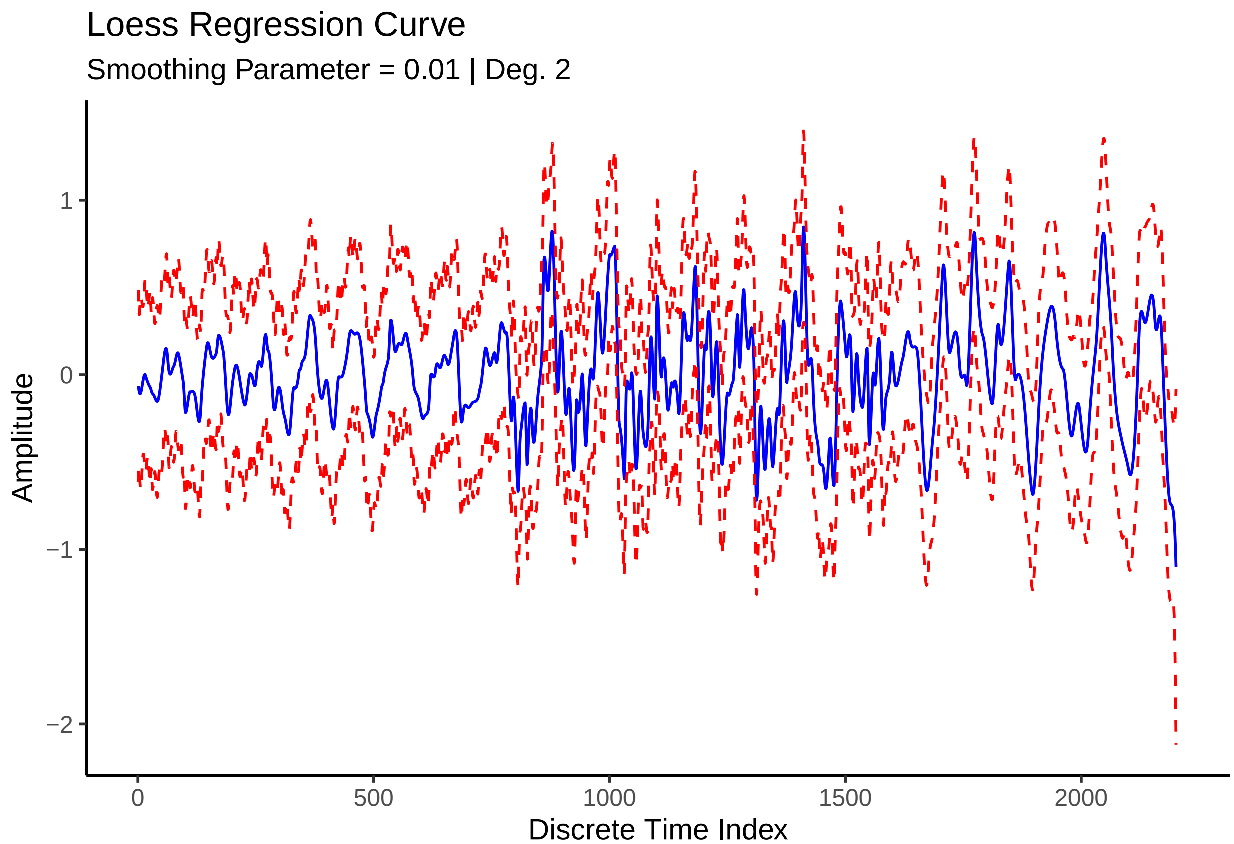
Real Subsample of Signal

```r
plot(extractWave(xn, from = 1, to = 2205), main = 'Subsample of Signal: n = 2205',
     xunit = 'samples', xlab = 'Time Index', ylab = 'Amplitude')
```



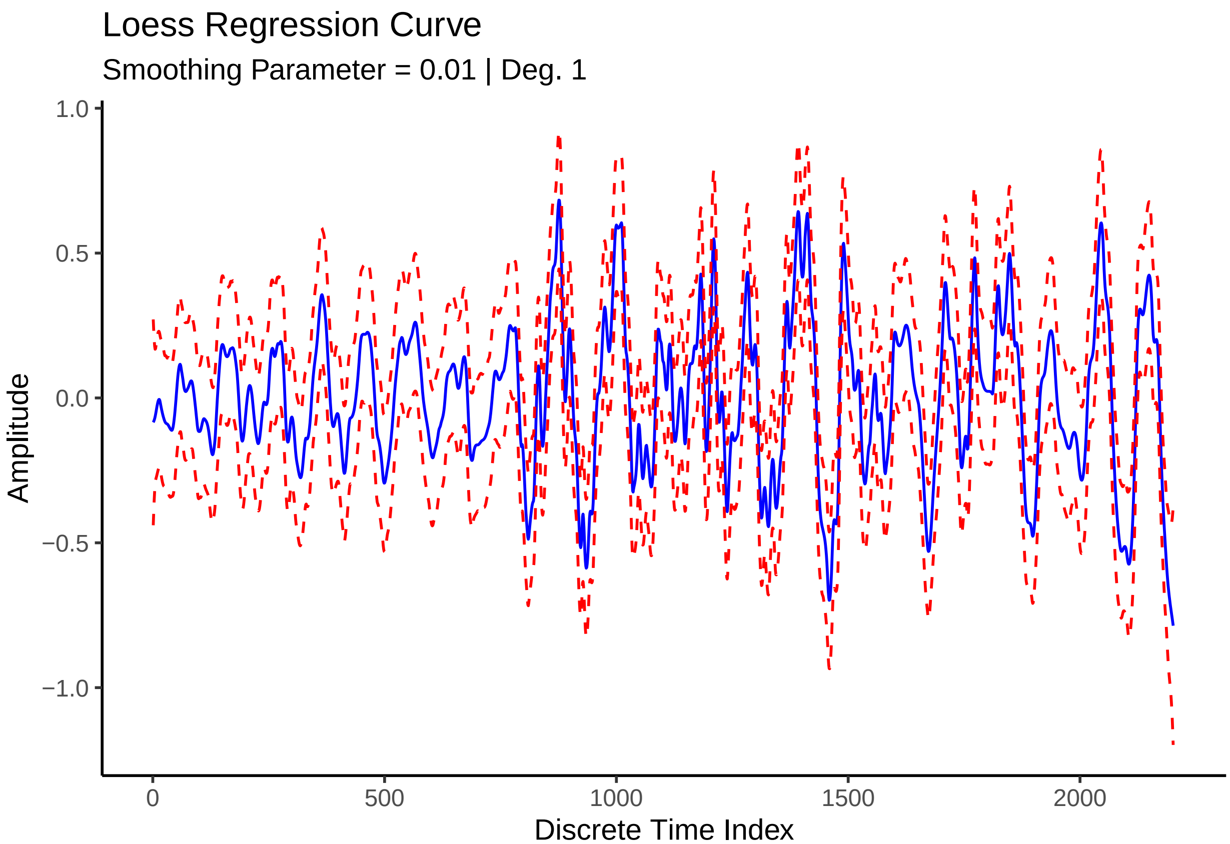
Loess Degree 2

```
curve_only_plot(dl2[[1]], type = 'loess', sub = subs2[[1]])
```



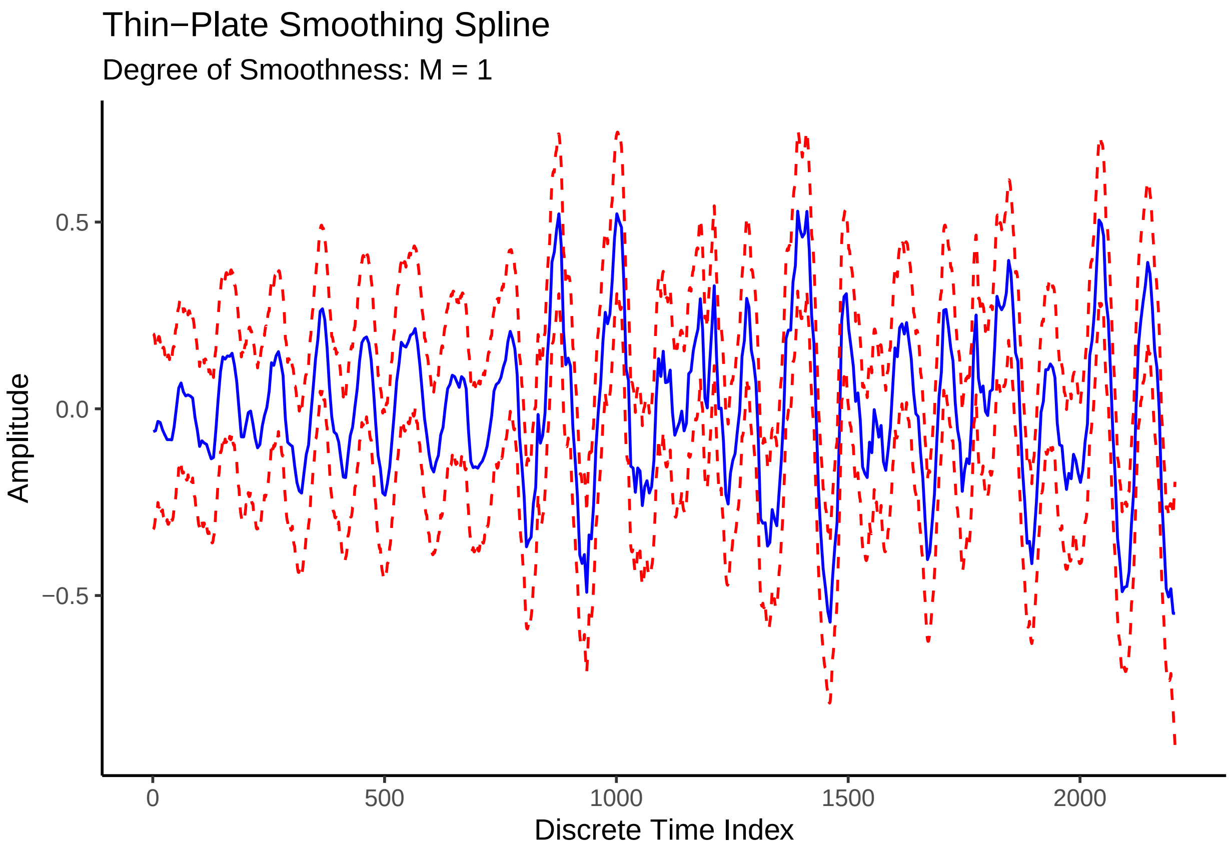
Loess Degree 1

```
curve_only_plot(dl1[[1]], type = 'loess', sub = subs1[[1]])
```



Thin-Plate

```
curve_only_plot(dtp[[1]], type = 'tp', sub = subs3[[1]])
```



Best Result

Based on the actual behavior of the signal, the quadratic loess regression curve with smoothing parameter 0.01 seemed to give the most accurate depiction of the actual signal.